Physicalc:
A Language for (simple) Scientific Computation

Brian Foo, bwf2101@columia.edu
Changlong Jiang, cj2214@columbia.edu
Ici Li, il2117@columbia.edu
Stuart Sierra, ss2806@columbia.edu

Final Report – December 18, 2007
COMS W4115 Program Languages and Translators
Columbia University, Fall 2007

Contents

Chapter 1

Introduction

Physicalc is a programming language for scientific computation, designed for students studying beginning and intermediate-level physics, chemistry, or other sciences.

Computer algebra systems are typically oriented towards higher mathematics, making them ill-suited to the sorts of calculations done by high-school and undergraduate science students. At the same time, some basic computer algebra features, such as symbolic computation, could be helpful to students. Physicalc presents itself initially as an intelligent calculator that understands physical units like "meters/second." For more advanced users, it supports real programming in an imperative style.

Physicalc is intended primarily as an educational tool, but may also be useful for exploratory data analysis in scientific fields.

1.1 Language Overview

1.1.1 Interpreter

Physicalc is an interpreted programming language. The interpreter is written in Java and executes a text file containing Physicalc source code.

1.1.2 Syntax

Physicalc syntax is as simple as possible, using mostly English words, more reminiscent of BASIC than C. Statements are separated by newlines. Statement blocks are enclosed in "do...done" pairs. Standard imperative-language features such as loops, if/then/else branching, and user-defined functions are provided. Standard mathematical operators are provided, with the addition of '^' for exponentiation.

1.2 Prior Art

- Units[16], a command-line program included in early UNIX systems and GNU/Linux, provides conversion factors between various units.

- Calchemy[1] is software for the Palm OS that combines a scientific calculator with unit conversion and dimensional analysis. It is based on an earlier Windows program called UNICALC.
- The Google Calculator[3] performs arithmetic on numbers including unit conversions.
- JScience[6] is a Java library for scientific computation including units.

Chapter 2

Tutorial

Physicalc is a language that is designed as a tool for students to learn physics, chemistry, and other sciences that involve the use of scientific units. Physicalc not only enables users to quickly convert between different metric systems, but allows for the automatic simplification of quantities of units that are multiplied or divided.

As an education tool, Physicalc is quite useful for a student or educator that has a solid grasp of algebra. Although existing software tools such as Mathematica and MATLAB[8] have algebraic functionality, they do not have tools that can handle physical constants and unit conversions. There are dozens of frameworks out there and most of them have been around much longer than Physicalc. Why should you care about yet another framework?

Physicalc is a simple language, and as such, requires a simple explanation for its mechanics. In creating Physicalc, we drew inspiration from the existing syntax for other languages, such as Basic and Perl. The basic constructs of the programming language are very similar to other languages. How does Physicalc achieve efficiency?

Part of the answer lies in Java, the language in which Physicalc is written. Java offers a wealth of resources and is particulary strong in the area of tokenizing and parsing. These tools allow Physicalc to quickly break down functions, units, and conditional statements very efficiently. Many things that are very simple to do in Physicalc are not even possible in most other languages. Physicalc takes full advantage of this. The rest of the answer is in two of Physicalc's guiding principles: less software and convention over configuration.

Less software means you write fewer lines of code to implement your physics or chemistry application. Keeping your code small means faster development and fewer bugs, which makes your code easier to understand, maintain, and enhance. Very shortly, you will see how Physicalc cuts your code burden.

Convention over configuration means an end to verbose XML header files or lengthy libraries. Instead of configuration files, a Physicalc program uses a few simple programming conventions that allow it to figure out everything through simplicity in the form of unit definitions, and function calls.

The following program prints "Hello world" followed by a line break. Using the `nprint()` command will print the same output without a line break.

```
print("Hello world")
```

In physicalc, every command is on its own line. The simple `set` command stores a numeric value into a variable. In the following example, the value 1 is stored into variables x and y. The next portion of the code performs operations under the condition that y remains less than or equal to 10. The product of z and y is stored into the variable z, overwriting the previous value, and the value of $y + 1$ is stored into variable y. In the loop, y converges towards 10, so the loop will not be infinite. The `done` statement indicates the loop's endpoint, and must be placed after every conditional statement and loop.

```
set y=1
set z=1
while y <= 10 do
  set z = z * y
  set y = y+1
done
```

The first line in the following program is a comment. Comments are not read by the computer, and are used for human reference only. The following program contains a function. A function is a subroutine that is called by the computer to perform a small task. A function takes in what is called a parameter. In this case, the parameter that is accepted by function `test` is the variable x. After the function is defined, `test` is called by the program, passing in a value of 6 as the x parameter. 6 is then fed into `test`, and is output by function `test`.

```
#this is a function
function test(x)
  print("x contains ", x)
  print(x)
done

print("use function to print x's contents")
test(6)
```

Perhaps what Physicalc is most useful for is defining physical units. Anyone who has taken an introductory Physics or Chemistry course understands the difficulties and complexities that are associated with units and physical constants. Physicalc allows you to define whatever constants you need to perform a certain calculation. Each such definition is signified by the `unit` command prefix. Starting with the most basic unit definitions first, consecutive unit definitions can add complexities to physics programs. In the following program, `second` is defined as a base unit, and a `minute` is defined as 60 seconds. When the `print()` function is called, Physicalc will print the contents of minute in simplest terms. In this case, `print` will display `60*second`.

```
# define the unit
unit second
unit minute = 60 * second

print(minute)
```

Chapter 3

Language Reference

3.1 Conventions in This Chapter

In this chapter, text in `monospace type` indicates a keyword or literal syntax. Text in *italic type* indicates a placeholder for some other piece of source code.

3.2 Program Sources

3.2.1 Encoding

Physicalc accepts source files encoded in plain 7-bit ASCII. High bit characters are not allowed. Line breaks may be encoded in DOS `\r\n`, UNIX `\n`, or Macintosh `\r` style. ASCII characters with decimal value less than 9, i.e. the control characters, are not allowed.

3.2.2 Comments

Comments begin with a hash character (`#`) and continue to the end of the line. Comments may be placed on the same line as source code. The line break is not considered part of the comment text.

3.2.3 Whitespace

Whitespace characters—spaces, tabs, and form feed characters—may be used to separate tokens in the input but are discarded during parsing.

3.2.4 Line Breaks and Semicolons

Line breaks are significant in Physicalc syntax. Line breaks serve as statement terminators. In the syntax rules that follow, all line breaks shown are mandatory.

To put multiple statements on a single line of source code, semicolons may be used in place of line breaks. Semicolons may be used anywhere a line break would normally be used, including between the parts of compound statements such as `if`/`elsif`/`else`.

Any number of consecutive line breaks and/or semicolons is read as a single terminator.

3.2.5 Identifiers

All identifiers begin with a letter or underscore, followed by zero or more letters, digits, and underscores. Identifiers are case-sensitive.

3.2.6 Reserved Words

The following words are reserved as keywords and may not be used as identifiers: `alias and break constant do done else elsif false for from function if in loop next not or return set step then to true unit while`

Additionally, Physicalc defines several built-in functions (Section 3.7) which may not be redefined.

3.3 Fundamental Types

3.3.1 Numbers

All numbers in Physicalc are treated as arbitrary-precision decimals. There is no distinction among integers, rationals, and reals. All arithmetical calculations are decimal-accurate to a reasonable degree of precision. Floating-point arithmetic is never used.

Literal numbers may be written in source code as integers or as decimals using C-style floating point number syntax. A number has three parts:

1. An integer part composed of digits;
2. A decimal part composed of a `.` (period) character followed by digits;
3. An exponent beginning with the letter `e` (or `E`), followed by an optional `+` or `-` sign, followed by digits.

All parts are optional, but either the integer or decimal part must be present.

3.3.2 Strings

Strings are sequences of ASCII characters. Literal strings may be written in source code by placing them between double-quotation (`"`) characters. A literal `"` character is written in a string as two consecutive `"` characters, so the string

The “big” bus

could be written as

```
"The ""big"" bus"
```

C-style character escapes (`\n`, `\t`, etc.) are not supported. Line breaks are permitted inside strings.

3.3.3 Lists

Lists are one-dimensional, variable-length, zero-indexed arrays of objects. Lists are heterogeneous—they may contain any combination of different object types. Lists are automatically resized.

A list literal may be written in source code by enclosing the entire list in square brackets (`[` and `]`) and separating individual elements with commas. Whitespace, but not line breaks, is permitted between list elements; it is ignored. Nested lists are allowed. The elements in a list literal may be expressions; those expressions are evaluated and their results are stored in the list.

3.3.4 Booleans

Boolean values are the literal identifiers `true` and `false`. These are global built-in constants and may not be redefined. In statements that use boolean expressions, any value that is not exactly equal to `false` is considered true. For example, the empty list and the number zero are both "true" in a boolean context.

3.4 Definitions

A typical Physicalc program consists primarily of definitions. There are four types of definitions: units, constants, functions, and aliases.

A definition permanently associates some identifier with an object. All definitions must occur at the top level of program scope; they may not be nested. Defintions are present in the running program from the point at which they are defined until the program terminates. Defined identifiers have global scope, and may not be overridden by local variables with the same name. An identifier, once defined, may be redefined.

3.4.1 Note on Definitions Using Expressions

Some of the definition types below take the form *identifier*`=`*expression*. In these cases, the `=` is not an operator; it is part of the syntax. The *expression* that follows the `=` symbol is evaluated once, at the time the definition is read, and its result is stored in the *identifier*.

3.4.2 Units

A unit is a concrete standard of measurement for some physical quantity.

Base Units

Base units are units for base quantities. Examples of base units in the SI system are meters for length, seconds for time, and kilograms for mass.[14] Base units are defined simply by giving them a name.

Syntax: `unit` *identifier*

where *identifier* is the name of the new unit.

Derived Units

Derived units are defined in terms of mathematical relationships to other units. An example of a derived unit in the SI system is the Coulomb, defined as seconds multiplied by Amperes.[15] Derived units are defined in Physicalc with expressions involving other units.

Syntax: `unit` *identifier* `=` *expression*

where *expression* may only consist of previously defined units, numbers, parentheses, and the arithmetical operators `+`, `-`, `*`, `/`, and `^`.

A derived unit may also be defined as a conversion factor from another unit. Typically, this type of derived unit will have a definition *expression* consisting of a base unit multipied by some constant number. In this way, conversion factors between different systems of measurement may be defined. Those conversion factors may be used for automatic unit conversion with the `in` operator (Section 3.5.7).

An identifier that has been defined as a base unit may not be subsequently redefined as a derived unit; to do so is an error.

3.4.3 Constants

Constants are static identifiers that hold any type of value. They have global scope and may not be changed by assignment. Constants may be changed by redefining them, but this produces a warning.

Syntax: `constant` *identifier* `=` *expression*

where *expression* is any expression.

3.4.4 Functions

Functions are named subroutines which receive input and return output. All function parameters are passed by value; i.e. a copy of the parameter is made and the function operates on the copy. Functions cannot modify any objects in their calling environment, nor can they define new global objects.

The first line of a function definition consists of the keyword `function`, an identifier, and a parameter list enclosed in parentheses. The body of the function, a sequence of statements, follows. The function definition ends with the keyword `done`.

Syntax:

```
function identifier ( parameter list )
        statements
done
```

The indentation of *statements* is for easier reading and has no significance in the syntax. The parameter list consists of zero or more identifiers, separated by commas. The parentheses around the parameter list are mandatory, even if the parameter list is empty. Whitespace, but not line breaks, is allowed in the parameter list. Functions taking a variable number of parameters are not supported, but this can be achieved in practice by passing a list as one of the parameters.

Within the body of a function, a `return` statement (Section 3.6.3) terminates the function and returns its argument as the value of the function. The body of a function may refer to identifiers

Operator	Description	Associativity
`( )`	Grouping	N/A
identifier `[ ]`	List subscript	left
identifier `( )`	Function call	left
`[ ]`	List Literal	N/A
`-`	Unary minus	right
`^`	Exponentiation	right
`* /`	Multiplication/Division	left
`+ -`	Addition/Subtraction	left
`> < >= <=`	Relational Comparison	left
`= !=`	Equality Comparison	left
`not`	Logical NOT	right
`and`	Logical AND	left
`or`	Logical OR	left
`in`	Unit Conversion	left
`,`	Comma separating expressions	left

Highest precedence is on the top line of the table.

Figure 3.1: Operator Precedence

not yet defined, but those identifiers must be defined before the function is called or an error will result. See Section 3.5.4 for the syntax of function calls.

3.4.5 Aliases

To allow for multiple names for the same object, aliases may be defined. An alias is an identifier that may be used in place of another identifier. Aliases are alternate names for an object rather than true references. Aliases are subject to the same redefinition constraints as other definitions.

Syntax: `alias` *identifier*$_1$ `for` *identifier*$_2$

defines *identifier*$_1$ as a new alias for *identifier*$_2$, a previously-defined identifier. Subsequent uses of *identifier*$_1$ will be read as if they were *identifier*$_2$. It is an error if *identifier*$_2$ is not already defined.

3.5 Expressions

Expressions consist of operators, function calls, literals, and identifiers. An expression, when evaluated, returns a value. Operator precedence is summarized in Figure 3.1. The types of expressions are described below. Expressions may contain whitespace, which is ignored, but they may not contain line breaks.

3.5.1 Arithmetical Expressions

Arithmetical expressions consist of the unary negation operator (`-`), parentheses `()`, and binary operators for addition (`+`), subtraction (`-`), multiplication (`*`), division (`/`), and exponentiation (`^`)

		Right Operand			
		Number	Unit	String	List
Left Operand	Number	+ - * / ^	* /		
	Unit	* / ^	* /		
	String			+	
	List				+

Figure 3.2: Permitted Binary Operations by Operand Type

The unary negation operator takes one argument on the right, and returns its opposite. There is no unary plus operator, because it serves no purpose that the authors can imagine.

Binary operators take one argument on the left and one argument on the right. The caret operator (`^`) performs exponentiation, raising its left argument to the value of its right argument. The unary `-` has highest precedence, followed by `^`, followed by `*` and `/`, followed by binary `+` and `-`.

Parentheses are used for grouping expressions and specifying precedence explicitly. An expression inside parentheses is always evaluated before other expressions. Parentheses may be nested to any level (within the bounds of computer memory) and the inner-most parenthetical expression will be evaluated first.

Arithmetic may be performed on numbers and units. All arithmetical operators are supported when the operands are both numbers. Units may be multiplied and divided with other units and numbers, but may not be added or subtracted (see Section 3.5.2). Units may be raised to a numerical exponent. Finally, concatenation is supported for strings and lists using the binary `+` operator. Figure 3.2 summarizes the supported binary operations.

3.5.2 Combining Numbers and Units

Mathematically, numbers with units are said to be "multiplied" by a symbol representing the unit. In Physicalc, this is taken literally. Units are identifiers, and a number with units is simply an expression of the form "*number*∗*identifier*" where *identifier* has been defined as a unit. Units are preserved in calculations and results. Limited handling of units as algebraic expressions is supported, so an expression such as "three meters per second multiplied by ten seconds" could be written

```
3 * meter / second * 10 * second
```

and would return the correct result of thirty meters as `30*meter`. Calculations requiring unit conversions might not always return the desired units in the result; the `in` operator (Section 3.5.7) forces conversion to the correct units.

3.5.3 List Member Access

Once a list has been stored in a variable, its elements may be accessed using bracketed indexes.

Syntax: *identifier* `[` *expression* `]` where *expression* must evaluate to an integer, which is used as an index into the list stored in *identifier*. An attempt to access an index beyond the end of the list produces an error.

Bracketed indexes are only permitted on identifiers, not on literal lists nor on expressions that return a list. Items in nested lists may be accessed with multiple consecutive bracket expressions, so if the variable `x` contained the list

```
[ a, b, [ c, d ], e ]
```

the element `d` could be referenced as `x[2][1]`.

3.5.4 Function Calls

Built-in or user-defined functions are called with the name of the function, an opening parenthesis, an argument list, and a closing parenthesis. The parentheses are mandatory even if the argument list is empty.

Syntax: *identifier* `(` *argument list* `)`

The argument list is a sequence of expressions, separated by commas. The number of arguments in the argument list of the function call must match the number of arguments in the function definition. Some built-in functions, such as `print()` (Section 3.7), take any number of arguments, but user-defined functions always have a fixed number of arguments.

When a function is called, the expressions in the argument list are evaluated. A new local scope is created, and the values of the arguments are bound to the named parameters from the function definition.

3.5.5 Relational Operators

Numbers, and only numbers, may be compared with the standard relational operators `>`, `<`, `>=`, and `<=`. These operators all return a boolean value.

Any two objects may be compared with the equality operator, `=`, which returns true if its left operand is of the same type and has the same value as its right operand, and false otherwise. Two units are equal only if they are the same unit.

The not-equals operator, `!=`, returns true if its operands are not equal under the definition of equality used for `=`, and false otherwise.

3.5.6 Logical Operators

Logical operators work on boolean values and expresions.

Syntax: `not` *expression*

returns true if *expression* is false and returns false if *expression* is true.

Syntax: $expression_1$ `and` $expression_2$

returns true if both $expression_1$ and $expression_2$ are true. This operator "short-circuits"—if $expression_1$ is false, it returns false without evaluating $expression_2$.

Syntax: $expression_1$ `or` $expression_2$

returns true if $expression_1$ is true, $expression_2$ is true, or both are true. This operator "short-circuits"—if $expression_1$ is true, it returns true without evaluating $expression_2$.

3.5.7 Unit Conversion

The special binary `in` operator is used to convert values from one set of units to another.

Syntax: $expression_1$ `in` $expression_2$

where $expression_1$ is an expression that evaluates to a number with units, and $expression_2$ evaluates to units. The `in` operator searches through the set of defined relationships among units and quantities to find the correct conversion factor, applies that conversion, and returns the result number in the requested units. It is an error if a valid conversion factor between the units of $expression_1$ and the units of $expression_2$ cannot be found.

3.6 Statements

Statements are source code constructs which do not return a value. An expression may be used as a statement by itself; its return value is discarded.

3.6.1 Loading Source Files

The special `load` statement reads in additional source files.

Syntax: `load "`*filename*`"`

The *filename* is interpreted as a path on the local filesystem, relative to the current working directory of the interpreter process, to a file containing Physicalc source code. That file is read and its contents are executed as if they had been included in the current program at the point of the `load` statement.

The loaded file is evaluated in the same global context as the program that called `load`—any definitions in the loaded file will become part of the running program. However, top-level variables created with `set` statements in the loaded program will *not* be visible to the main program.

If the file cannot be found or cannot be read, an error results. `load` is only allowed at the top-level of a program source file; it may not appear inside functions or other statements.

3.6.2 Assignment

Syntax: `set` *identifier* `=` *expression*

An assignment statement evaluates *expression*, then binds its value to the local variable named *identifier*. If *identifier* is currently undefined, a new local variable is created with scope corresponding to the current function body. It is an error if *identifier* is already defined as a global object, i.e. a unit, function, or constant.

Assignment statements may be used outside of a function body, but doing so does not create a global variable. Global variables are not supported, only global constants. The top-level of a Physicalc program has its own scope for local variables, as if it were in the body of a function.

3.6.3 Return

Syntax: `return` *expression*

A `return` statement may only appear inside the body of a function; a `return` statement found outside of a function body is an error. When a `return` statement is executed, *expression* is evaluated and its value is returned as the value of the function.

3.6.4 If/Then/Else

An if/then/else block begins with the keyword `if`, followed by a boolean expression, followed by the keyword `then` and a terminator (line break or semicolon). After `then` comes a sequence of one or more statements, then any number of `elsif` blocks, then an optional `else` block, then finally the keyword `done`.

Syntax:

```
if expression₁ then
        statements₁
elsif expression₂ then
        statements₂
... additional elsif blocks ...
else
        statements₃
done
```

The indentation of the statement blocks is for easier reading and is not significant in the syntax. First, *expression*$_1$ is evaluated. If it returns true, *statements*$_1$ are executed. After completing *statements*$_1$, control passes to the statement following the `done` keyword.

If *expression*$_1$ returns false, *expression*$_2$ is evaluated. If *expression*$_2$ returns true, *statements*$_2$ are executed, then control passes to the statement following the `done` keyword. Additional `elsif` blocks may specify additional test expressions and statements to execute. If all of the test expressions return false, and if the optional final `else` block is present, *statements*$_3$ are executed.

An if/then/else block might not execute any statements at all if there is no `else` block. An if/then/else block never executes more than one group of statements. Once the first test expression returns true, its associated statement block is executed and all other test expressions and statement blocks are skipped.

3.6.5 While Loops

Syntax:

```
while expression do
        statements
done
```

While loops evaluate a group of statements as long as a given conditional expression remains true. The conditional *expression* is evaluated before the statement body on every iteration of the loop. If it returns true, the *statements* are executed. The first time *expression* returns false, control passes to the statement following the `done` keyword.

3.6.6 For Loops

Syntax:

```
for identifier from expression1 to expression2 step expression3 do
      statements
done
```

At the beginning of a for loop, *expression*$_1$, *expression*$_2$, and *expression*$_3$ are all evaluated immediately. All three must evaluate to positive numbers, optionally including units; it is an error if they do not. The result of *expression*$_1$ is assigned to the local variable *identifier*. If *identifier* is undefined, a new local variable is created with scope corresponding to the current function body. It is an error if *identifier* is already defined as a global. The *statements* are executed, after which the value of *expression*$_3$ is added to the value in *identifier*, and that new value is stored in *identifier*. Then the value of *identifier* is compared to the value of *expression*$_2$. If *identifier* is greater than the value of *expression*$_2$, control passes to the statement following the `done` keyword. If *identifier* is less than or equal to the value of *expression*$_2$, *statements* are executed again. This process repeats until *identifier* is greater than the value of *expression*$_2$ or a `break` or `return` statement is executed.

3.6.7 Control Statements Within Loops

Within any loop structure there are two special statements which affect the execution of the loop. The `break` statement will immediately terminate the execution of the loop and transfer control to the statement following the loop's `done` keyword.

The `next` statement will immediately return control to the top of the loop. In the case of `while` loops, the loop test is applied as if the loop had reached the end of its statement block. In the case of `for` loops, the counter variable is incremented and the test is applied as if the loop had reached the end of its statement block.

Additionally, a loop used inside of a function body may contain a `return` statement, which will immediately break out of the loop and return from the function.

3.7 Built-In Functions

Physicalc provides a some built-in functions, which have the same syntax as normal function calls but carry out operations which could not be implemented with normal Physicalc code.

3.7.1 print()

The `print()` function takes any number of arguments, which may be of any types, and print them to the output stream, followed by a line break. Printing a string prints its contents without the enclosing `"` characters. Lists, units, and numbers are automatically converted to strings as with the `toString()` function before being printed.

3.7.2 nprint()

The `nprint()` function acts like `print()` but does not print a line break.

3.7.3 toString()

The `toString()` function converts its argument, which may be any object, to a string. If the argument is already a string, it is simply returned. Other types of arguments are converted to a string that matches their literal syntax.

3.7.4 getNumber()

The `getNumber()` function takes an argument of a number with units and removes all units, leaving just the bare number. If a bare unit without any numbers is passed to the function, it returns the number one.

3.7.5 getUnit()

The `getUnit()` function takes an argument of a number with units and removes the number, leaving just the units.

3.7.6 toInt()

The `toInt()` truncates the decimal part of its argument, leaving an integer.

3.7.7 exit()

The `exit()` function, which takes no arguments, immediately terminates the Physicalc program.

Chapter 4

Project Plan

4.1 Process

Physicalc was developed on a wiki hosted by Google Code, available at `http://code.google.com/p/bcis/`.

4.2 Programming Style

In general, Physicalc follows the Java coding standards published by Sun[2]. Details below.

4.2.1 Spacing

- Indents are 4 spaces
- Put a space between keywords (while, for) and parentheses
- No space between method names and parentheses
- Put a blank line between method definitions

4.2.2 Names

- Interface and class names are `MixedCase` and start with a capital letter
- Methods and variables are `mixedCase` and start with a lower-case letter
- Constants are `UPPER_CASE_WITH_UNDERSCORES`

4.2.3 Comments

- Use `/* C-style block comments */` for comments longer than one line
- Use `// single-line comments` to comment-out sections of code
- Use Javadoc[4]

4.2.4 Braces

- Open brace `{` goes on the same line as the declaration
- Closing brace `}` goes on a line by itself, indented to match the start of the declaration
- Always use braces for if/else statements

4.2.5 ANTLR Grammar Files

For ANTLR source files, use the conventions from class:

- Token names in the Lexer are ALLCAPS
- Nonterminal names in the Parser are lowercase
- Separate "`|`" branches in productions go on separate lines

4.3 Project Timeline

September 17	Project Selected
September 18	Project Wiki Created
September 25	Proposal Submitted
October 17	Grammar and Parser Completed
October 18	Reference Manual Submitted
December 1	Tree Walker Completed
December 8	Interpreter Completed
December 15	Testing Completed
December 17	Presentation
December 18	Final Report Submitted

4.4 Roles and Responsibilities

Brian Foo	Data Classes, Unit System, Function Definitions
Changlong Jiang	Lexer, Statement Nodes, Example Programs, Testing
Ici Li	Tutorial, Expression Nodes, Built-in Functions
Stuart Sierra	Team Leader, Design, Proposal, Reference Manual, Parser, Tree Walker

4.5 Tools

- Sun Java 1.6 Development Kit
- ANTLR 2.7.7
- Subversion, hosted by Google Code
- GNU Make
- JUnit
- bash scripts (for testing)

4.6 Project Log

The complete project log, generated from Subversion, is included in the source file listing at the end of this report as `Changelog`. This log encompasses changes to both the wiki and the source code repository. Team members can be identified in the log by their user names as follows:

Brian Foo	brianwfoo
Changlong Jiang	ChadJiang
Ici Li	digitalfobulous
Stuart Sierra	the.stuart.sierra or ssierr@law.columbia.edu

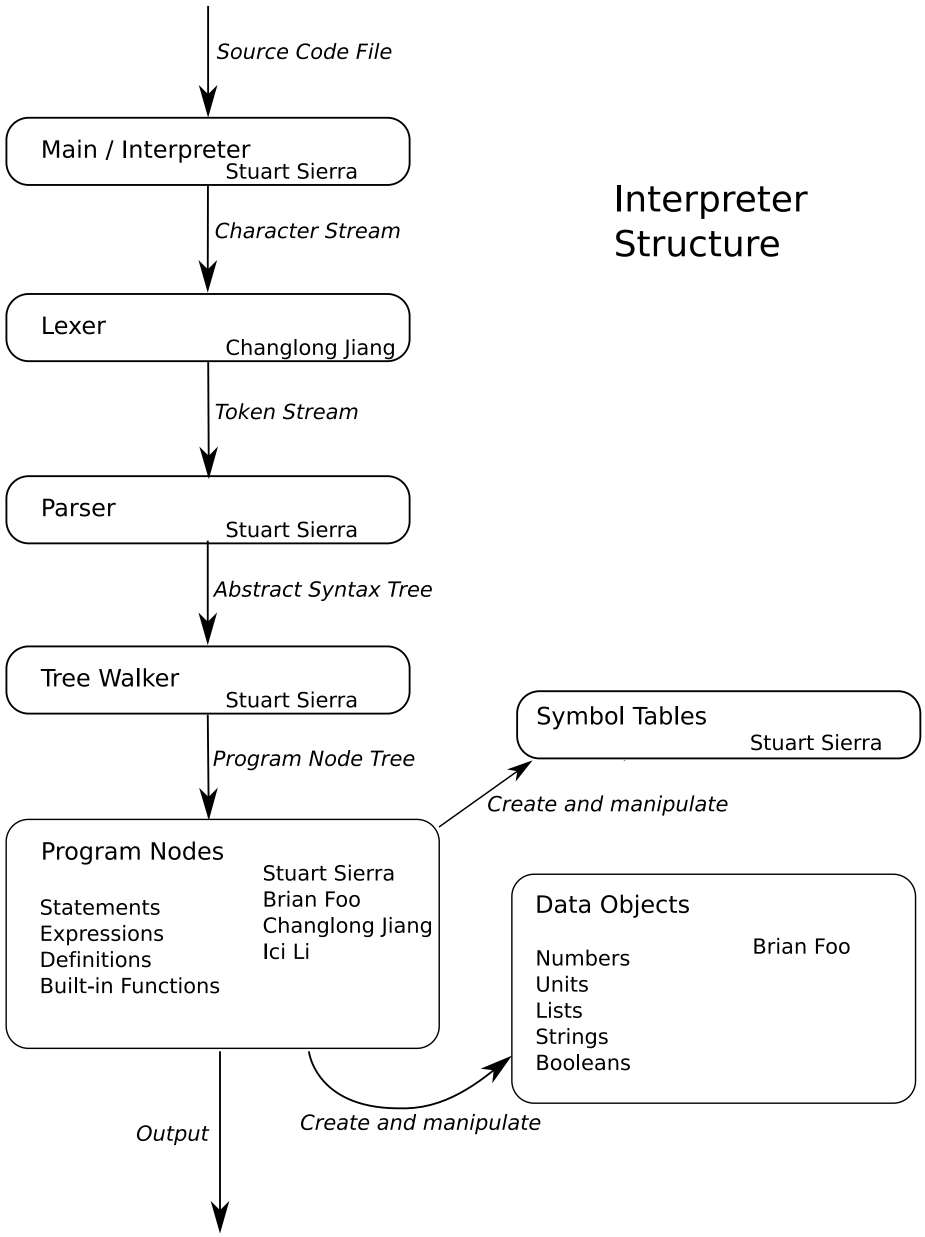
Chapter 5

Architectural Design

Physicalc is an interpreted programming language. The diagram on page 22 shows the general structure of the interpreter. The lexer and parser produce an abstract syntax tree, which is transformed by a tree walker into a tree of program nodes. The program nodes are all sub-classes of the Node class, each node represents a single program structure such as a statement or a function call. Every node class provides an "eval" method which is responsible for executing the behavior of that node.

The root of the tree is an instance of the Program class. The interpreter calls "eval" on the Program, which calls "eval" on its sub-nodes, and so on, recursively, so the node tree executes itself. The symbol tables are carried through the tree as arguments to "eval." Because Physicalc does not support nested scopes, there are never more than two symbol tables, one global and one local, in effect at any given time. The node structure was designed by Stuart Sierra, and the sub-classes were implemented by Brian Foo, Changlong Jiang, and Ici Li.

The data objects manipulated by a Physicalc program—numbers, units, lists, and strings—are all sub-classes of the Datum class. Datum defines virtual methods for all the arithmetical operators, which are overridden in sub-classes. Calling an operator on two types for which it is not defined, e.g. addition between two units, results in a exception of class TypeError. Brian Foo implemented the Datum sub-classes.


Source Code File
Main / Interpreter
Stuart Sierra
Interpreter Structure
Character Stream
Lexer
Changlong Jiang
Token Stream
Parser
Stuart Sierra
Abstract Syntax Tree
Tree Walker
Stuart Sierra
Symbol Tables
Stuart Sierra
Program Node Tree
Create and manipulate
Program Nodes
Statements
Expressions
Definitions
Built-in Functions
Stuart Sierra
Brian Foo
Changlong Jiang
Ici Li
Data Objects
Numbers
Units
Lists
Strings
Booleans
Brian Foo
Output
Create and manipulate

Chapter 6

Example Programs

6.1 Example 1: Calculating Factorials

This program uses `for`, `while`, and `if` statements to calculate factorials.

6.1.1 Program Source

```
# Calculate Factorial
# This program is testing Looping function
# Test Program written by Changlong Jiang cj2214@columbia.edu
# Date 12/15/2007

# use For Loops
set y=1
for x from 1 to 10 step 1 do
set y = y*x
done

print("use For Loop")
print(y)

# use While Loops
set y=1
set z=1
while y <= 10 do
set z = z * y
set y = y+1
done

print("use While Loop")
print(z)
```

```
#use IF
print("use While Loop and If")
set y=1
set z=1
while y <= 10 do
set z = z * y
  if y>9 then
    print("y=",y," ","greater than 9, stop")
    return y;
  elsif y>6 then
    print("y=",y," ","greater than 6, continue")
    print("result=",z)
  else
    print("y=",y," ","less and equal than 6,continue")
    print("result=",z)
  done
set y = y+1
done
```

6.1.2 Output

```
use For Loop
3628800.0

use While Loop
3628800.0

use While Loop and If
y=1.0 less and equal than 6,continue
result=1.0
y=2.0 less and equal than 6,continue
result=2.0
y=3.0 less and equal than 6,continue
result=6.0
y=4.0 less and equal than 6,continue
result=24.0
y=5.0 less and equal than 6,continue
result=120.0
y=6.0 less and equal than 6,continue
result=720.0
y=7.0 greater than 6, continue
result=5040.0
y=8.0 greater than 6, continue
result=40320.0
y=9.0 greater than 6, continue
result=362880.0
```

```
y=10.0 greater than 9, stop
```

6.2 Example 2: Factorials and Logical Comparisons

6.2.1 Program Source

```
# Calculate Factorial
# This program is testing function and logical operation
# Test Program written by Changlong Jiang cj2214@columbia.edu
# Date 12/15/2007

#this is function for factorial number
function factorial(x)
  print("x=",x)
  set y=1
  set z=1
  while y <= x do
    set z = z * y
    set y = y+1
  done
  nprint(x,"!=")
  print(z)
done

print("use function to calculate factorial")
factorial(6)

# this is function to find the biggest number
function findbiggest(x,y,z)
   print("x=",x," ","y=",y," ","z=",z)
   if x>=y and y>=z then
     print("x is biggest")
   done
   if x>=y or y>=z then
     print("x is not smallest")
   done
   if not(y>=x) then
     print("y is smaller than x")
   done
done

set x = [7,5,3]
findbiggest(x[0],x[1],x[2])
```

6.2.2 Output

```
use function to calculate factorial
x=6.0
6.0!=720.0
x=7.0 y=5.0 z=3.0
x is biggest
x is not smallest
y is smaller than x
```

6.3 Example 3: Calculating the Mass of the Sun

This physics program was published by the University of Oregon[11]: "Estimate the mass of the sun given the Earth's distance from the sun $r = 1.50 \times 10^{11} m$. Assume the Earth follows a circular orbit instead of an elliptical one. $G = 6.67 \times 10^{-11} Nm^2/kg^2$."

With the solution:

$$
\begin{aligned}
F_g &= F_c = ma_c = m_{earth} r\omega^2 = Gm_{earth}m_{sun}/r^2 \\
m_{sun} &= r^3\omega^2/G \\
365days &= 3.15 \times 10^7 s \\
1\text{rotation} &= 2\pi rad \\
\omega &= 2\pi rad/3.15 \times 10^7 s = 1.99 \times 10^{-7} rad/s \\
m_{sun} &= (1.50 \times 10^{11} m)^3 (1.99 \times 10^{-7} rad/s)^2 / 6.67 \times 10^{-11} Nm^2/kg^2 \\
m_{sun} &= 2.01 \times 10^{30} kg
\end{aligned}
$$

6.3.1 Program Source

```
# This is for Sun Mass Calculation
# Estimate the mass of the sun given the Earth's distance from the sun
# r=1.50*10^11 meter
# Assume the Earch follows a circular orbit
# Universal Gravitational consatant G=6.67*10^(-11)*Newton*meter^2/kilogram^2
# source from http://zebu.uoregon.edu/~probs/mech/grav
# test program written by Changlong Jiang : cj2214@columbia.edu
# Date 12/15/2007

# define the unit
unit second
unit minute = 60 * second
unit hour = 60 * minute
unit day = 24 * hour
unit year = 365 * day
unit meter
alias m for meter
unit kilogram
```

```
unit newton = m * kilogram / second ^ 2

# define the variable and calculate
set x = 1 * year
set Pi = 3.1415926
set omiga = 2 * Pi/x
set G = 6.67E-11 * newton * (1*m ^2) / (1 *kilogram ^ 2)
set r = 1.50E11 * m
set mass = (1*r^3) * (1*omiga^2)/G

#print result
print(mass)
```

6.3.2 Output

```
2.0086045922465554E30*kilogram
```

6.4 Example 4: Calculating the Radius of the Moon's Orbit

This physics problem was published by the Unversity of Oregon[12]: "The orbital period (T) of the Moon around the Earth is 29.53 days. Calculate the radius of orbit of the Moon assuming the orbit is circular. You are given the Universal Gravitation Constant, $G = 6.67 \times 10^{-11} Nm^2/kg^2$, and the mass of the Earth, $M_e = 5.98 \times 10^{24} kg$."

The solution is just lengthy enough to make typing it out in LaTeX a pain, so take our word (or the University of Oregon's) for it that it works out to $r = 4.04 \times 10^8 m = 404,000km$.

6.4.1 Program Source

```
#This is for Calculate the radius of orbit of the Moon
#Universal Gravitational consatant G=6.67*10^(-11)*Newton*meter^2/kilogram^2
#Earth Mass is 5.98E24 * kilogram
#Source from http://zebu.uoregon.edu/~probs/mech/grav/distmoon
#Test Program written by Changlong Jiang : cj2214@columbia.edu
#Date 12/15/2007

#load the pre-defined unit
load "si.phy"

#set variable
set x = 29.53 * day
nprint("seconds:",x)
print()

#print(x in hour)
set y = 29.53 * 24 * 3600*second
print("Number is:", getNumber(y))
```

```
print("Unit is:",getUnit(y))
print("hours:",y in hour)

set Pi = 3.1415926
set G = 6.67E-11 * newton * (1*meter ^ 2) /(1*kilogram^2)
set masse = 5.98E24 * kilogram
set r = ((x*(1*G*(1*masse))^(1/2))/(2*Pi))^(2/3)
print(r)
print("Number is:", getNumber(r))
print("Unit is:",getUnit(r)))
```

6.4.2 Output

```
seconds:2551392.0*second
Number is:2551392.0
Unit is:second
hours:708.72*hour
4.036521081066972E8*meter^1.0
Number is:4.036521081066972E8
Unit is:meter^1.0
```

Chapter 7

Tests

The Physicalc test suite consists of three parts: 1) interactive programs used during development; 2) unit tests implemented with JUnit; and 3) integration tests that test the whole interpreter, implemented with Bash shell scripts.

7.1 Interactive Testing During Development

Several executable Java programs assisted in developing and debugging the interpreter. These files remain in the source tree as `src/Try*.java`.

TryLexer runs the Physicalc lexer on a string, given as a command line argument, and prints out the tokens, one per line.

TryParser runs the lexer and parser on a string, given as a command line argument, and prints out the abstract syntax tree in a Lisp-like syntax.

ParseFile acts like TryParser, but takes a file name as an argument and parses the file.

TryDatum executes various arithmetical operations on the Datum sub-classes and prints the results.

TryDatum was written by Brian Foo; TryLexer, TryParser, and ParseFile were written by Stuart Sierra.

7.2 Unit Tests

To test the operation of simple expressions in the interpreter, tests defined using JUnit[7] are defined in `test/InterpreterTest.java`. This class defines an "assertPrints" method which calls the interpreter on a string of source code and checks that it produces a certain output. Due to Java's lack of support for multi-line strings in source code, this mode of testing is awkward for longer programs. The Makefile target `test` compiles the interpreter and runs the unit tests. The unit tests framework was written by Stuart Sierra and tests were added by Changlong Jiang.

7.3 Integration Tests

Tests of the complete behavior of the interpreter are implemented with the `runexamples` shell script. This script looks for files named `*.in` and `*.out` in the directory `test/examples`. The "in" files are Physicalc source code containing one or more "print" statements. The corresponding "out" files contain the text that the program should print. The `runexamples` script runs the interpreter on each "in" file and compares its output with its "out" file, reporting how they differ. The `runexamples` script was written by Stuart Sierra; test programs were contributed by all team members. The example files are included in the source code section at the end of this report.

Chapter 8

Lessons Learned

8.1 Brian Foo

The lessons learned from this project were things you can't really learn enough of—clear communication with team members, project management, and work flow. More specifically, I learned the importance of setting clear goals and responsibilities of each team member in order to reach a common end. On a more technical note, I learned the importance of taking my time to solve a design problem before diving into it head-first. This would be exponentially rewarded in the future when you are too deep into the project to go back.

8.2 Changlong Jiang

1. For the lexer part: I should fully understand the language grammar. The good easy-to-understand, easy-to-use grammar is the important thing for the language design. During the project, we modified our grammar several times to make the whole language concise and easy to use.

2. For the node implementation: I am not good at the Java languagae. I only finished several simple classes. I should improve my skills in Java.

3. For testing: I spent a lot of time in testing, but still some scopes I did not test. Since the relationship between each node is more complex, I should leave more time to do testing. For every bug found in each test I always gave out the detailed informaton and notified other team members. If I can understand other members' work, I will be able to fix them during my testing, which will save the whole team's time.

4. Need to add detailed comments in every source file, which will be easier for others to understand.

8.3 Ici Li

Having a very strong team is an integral part of every large software project. In particular, it is absolutely essential to have a capable team leader. Stuart Sierra has been an extraordinary team leader for this project, and I really enjoyed working with him, as well as the rest of my teammates this semester. This was a very humbling experience, as I realized how much I did not know coming into the class. It was interesting seeing how concepts that were taught in class were applicable to a real-world situation. Problems such as scoping and Normal vs Applicative order became very real to me as I slowly began to understand the intricacies of parsing and enabling the compiler to understand input. Overall, I had a rewarding experience, and enjoyed this course.

8.4 Stuart Sierra

Managing coders is much harder than writing code. It's a challenge trying to figure out what people's strengths are and how to use them effectively. On the other hand, having to explain my design so that others could implement it forced me to think more carefully about the implementation ahead of time, and probably resulted in a better design than I would have come up with had I just started banging out code on my own. Perhaps I should have enforced coding standards and unit testing more aggressively—this might have left us with fewer bugs to fix—but I was reluctant to play a dictator, even a benevolent one, with a group I had just met. I understand now why "successful" programming languages usually have one stubborn individual or large coporation calling the shots.

I discovered why so few languages use significant whitespace—it's really annoying to implement. Even though our language only used line breaks, not spaces or tabs, as significant characters, it was still tricky to get all the parser rules to work correctly. I had to punt on the issue of multi-line expressions by forbidding them altogether.

I made a choice early on that Physicalc would perform decimal-accurate arithmetic, without considering the implications. Java's BigDecimal class does the job, but the results were not always what I intended. Decimal-accurate arithmetic is a nice idea, but it requires decisions about how to handle precision, rounding, and exponents. We never had a clear plan for dealing with these issues, so the final implementation represents a series of compromises rather than any sound mathematical theory.

Bibliography

[1] Calchemy, http://www.calchemy.com/

[2] Code Conventions for the Java Programming Language, Sun Developer Network, http://java.sun.com/docs/codeconv/

[3] Google Calculator, http://www.google.com/help/calculator.html

[4] How to Write Doc Comments for the Javadoc Tool, Sun Developer Network, http://java.sun.com/j2se/javadoc/writingdoccomments

[5] "International System of Units." Wikipedia, http://en.wikipedia.org/wiki/Si_units

[6] JScience, http://jscience.org/

[7] JUnit, http://www.junit.org/

[8] MATLAB, http://www.mathworks.com/products/matlab/

[9] Mathematica, http://www.wolfram.com/products/mathematica/

[10] Maxima, http://maxima.sourceforge.net/

[11] Physics Problem: Calculating the Mass of the Sun. University of Oregon, http://zebu.uoregon.edu/~probs/mech/grav/Gravity2/Gravity2.html

[12] Physics Problem: Calculating the Radius of the Moon's Orbit. University of Oregon, http://zebu.uoregon.edu/~probs/mech/grav/distmoon/distmoon.html

[13] "Rounding numbers." Wikipedia, http://en.wikipedia.org/wiki/Rounding_numbers

[14] "SI base unit." Wikipedia, http://en.wikipedia.org/wiki/SI_base_unit

[15] "SI derived unit." Wikipedia, http://en.wikipedia.org/wiki/SI_derived_unit

[16] Units, GNU Project, http://www.gnu.org/software/units/units.html

Chapter 9

Source Code

Copies of all the source files are attached to this report.

```
------------------------------------------------------------------------
r349 | the.stuart.sierra | 2007-12-18 20:17:36 -0500 (Tue, 18 Dec 2007) | 3 line
s
Changed paths:
   M /trunk/Makefile
   M /trunk/report/bibliography.tex
   M /trunk/report/plan.tex
   M /trunk/report/tests.tex
   M /trunk/report/tutorial.tex
   M /trunk/runexamples
   M /trunk/si.phy
   M /trunk/src/ParseFile.java
   M /trunk/src/TryLexer.java
   M /trunk/src/TryParser.java
   M /trunk/src/physicalc/Access.java
   M /trunk/src/physicalc/AliasDef.java
   M /trunk/src/physicalc/And.java
   M /trunk/src/physicalc/Block.java
   M /trunk/src/physicalc/BoundsError.java
   M /trunk/src/physicalc/Break.java
   M /trunk/src/physicalc/BreakSignal.java
   M /trunk/src/physicalc/Constant.java
   M /trunk/src/physicalc/ConstantDef.java
   M /trunk/src/physicalc/ControlSignal.java
   M /trunk/src/physicalc/Datum.java
   M /trunk/src/physicalc/Def.java
   M /trunk/src/physicalc/ExitFunction.java
   M /trunk/src/physicalc/Expr.java
   M /trunk/src/physicalc/ExprList.java
   M /trunk/src/physicalc/FunCall.java
   M /trunk/src/physicalc/Function.java
   M /trunk/src/physicalc/FunctionDef.java
   M /trunk/src/physicalc/GetNumberFunction.java
   M /trunk/src/physicalc/GetUnitFunction.java
   M /trunk/src/physicalc/Id.java
   M /trunk/src/physicalc/In.java
   M /trunk/src/physicalc/Interpreter.java
   M /trunk/src/physicalc/InterpreterError.java
   M /trunk/src/physicalc/LValue.java
   M /trunk/src/physicalc/Literal.java
   M /trunk/src/physicalc/Load.java
   M /trunk/src/physicalc/Logical.java
   M /trunk/src/physicalc/Main.java
   M /trunk/src/physicalc/NPrintFunction.java
   M /trunk/src/physicalc/Next.java
   M /trunk/src/physicalc/NextSignal.java
   M /trunk/src/physicalc/Node.java
   M /trunk/src/physicalc/Op.java
   M /trunk/src/physicalc/Or.java
   M /trunk/src/physicalc/PBoolean.java
   M /trunk/src/physicalc/PList.java
   M /trunk/src/physicalc/PNumber.java
   M /trunk/src/physicalc/PString.java
   M /trunk/src/physicalc/PUnit.java
   M /trunk/src/physicalc/PUnitPair.java
   M /trunk/src/physicalc/ParamList.java
   M /trunk/src/physicalc/PrintFunction.java
   M /trunk/src/physicalc/Program.java
   M /trunk/src/physicalc/Rel.java
   M /trunk/src/physicalc/Return.java
   M /trunk/src/physicalc/ReturnSignal.java
   M /trunk/src/physicalc/RuntimeObject.java
```



```
   M /trunk/src/physicalc/Set.java
   M /trunk/src/physicalc/Stmt.java
   M /trunk/src/physicalc/SymbolTable.java
   M /trunk/src/physicalc/ToIntFunction.java
   M /trunk/src/physicalc/ToStringFunction.java
   M /trunk/src/physicalc/TypeError.java
   M /trunk/src/physicalc/Unary.java
   M /trunk/src/physicalc/UndefinedError.java
   M /trunk/src/physicalc/Unit.java
   M /trunk/src/physicalc/UnitDef.java
   M /trunk/src/physicalc/Variable.java

* Added author names to source files.
* Updated final report.

------------------------------------------------------------------------
r348 | the.stuart.sierra | 2007-12-17 21:48:15 -0500 (Mon, 17 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/report/lessons.tex

Added "lessons learned" from Ici.

------------------------------------------------------------------------
r347 | ChadJiang | 2007-12-16 22:43:11 -0500 (Sun, 16 Dec 2007) | 1 line
Changed paths:
   M /trunk/rundemocode

modify rundemocode
------------------------------------------------------------------------
r346 | the.stuart.sierra | 2007-12-16 21:54:34 -0500 (Sun, 16 Dec 2007) | 3 line
s
Changed paths:
   M /trunk/test/examples/UnitTime.in
   M /trunk/test/examples/UnitTime.out
   M /trunk/test/examples/alias.out
   M /trunk/test/examples/const1.out
   M /trunk/test/examples/exit.out
   M /trunk/test/examples/for1.out
   M /trunk/test/examples/for2.out
   M /trunk/test/examples/funcTest.out
   M /trunk/test/examples/getNumber.out
   M /trunk/test/examples/in.out
   M /trunk/test/examples/mathexample1.out
   M /trunk/test/examples/phy1.out
   M /trunk/test/examples/phy2.out
   M /trunk/test/examples/phy3.out
   M /trunk/test/examples/phy4.out
   M /trunk/test/examples/set1.out
   M /trunk/test/examples/toInt.out
   M /trunk/test/examples/toString.out
   M /trunk/test/examples/unary.out
   M /trunk/test/examples/unit1.out
   M /trunk/test/examples/unit2.out
   M /trunk/test/examples/unit3.out
   M /trunk/test/examples/unit7.out
   M /trunk/test/examples/while2.out
   M /trunk/test/examples/while3.out
   M /trunk/test/examples/while4.out

Modified test output files to reflect change to printing all
numbers as doubles.
```

```
------------------------------------------------------------------------
r345 | the.stuart.sierra | 2007-12-16 15:59:52 -0500 (Sun, 16 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/report/bibliography.tex
   M /trunk/report/design.tex
   A /trunk/report/examples.tex
   M /trunk/report/finalreport.tex
   M /trunk/report/intro.tex
   M /trunk/report/lessons.tex
   M /trunk/report/plan.tex
   M /trunk/report/refman.tex

Updated final report files for draft 3.

------------------------------------------------------------------------
r344 | ChadJiang | 2007-12-16 12:31:05 -0500 (Sun, 16 Dec 2007) | 1 line
Changed paths:
   M /trunk/test/examples/phy1.in
   M /trunk/test/examples/phy2.in
   M /trunk/test/examples/phy4.in

update some typo
------------------------------------------------------------------------
r343 | brianwfoo | 2007-12-16 11:27:27 -0500 (Sun, 16 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/GetUnitFunction.java

Added a file remotely
------------------------------------------------------------------------
r342 | brianwfoo | 2007-12-16 11:27:09 -0500 (Sun, 16 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/GetUnitFunction.java

Removed file/folder
------------------------------------------------------------------------
r341 | brianwfoo | 2007-12-16 11:26:54 -0500 (Sun, 16 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r340 | brianwfoo | 2007-12-16 11:26:33 -0500 (Sun, 16 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r339 | ChadJiang | 2007-12-15 23:53:43 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   M /trunk/test/examples/phy2.in

add functioncall
------------------------------------------------------------------------
r338 | ChadJiang | 2007-12-15 23:40:07 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   M /trunk/test/examples/phy2.in
   M /trunk/test/examples/phy2.out

add functioncall
------------------------------------------------------------------------
```



```
r337 | the.stuart.sierra | 2007-12-15 22:54:47 -0500 (Sat, 15 Dec 2007) | 3 line
s
Changed paths:
   M /trunk/Makefile
   M /trunk/report/bibliography.tex
   A /trunk/report/design.tex
   M /trunk/report/finalreport.tex
   A /trunk/report/lessons.tex
   A /trunk/report/plan.tex
   M /trunk/report/refman.tex
   A /trunk/report/source.tex
   A /trunk/report/structure.svg
   A /trunk/report/tests.tex
   A /trunk/report/tutorial.tex

* Committed all files for draft 2 of the final report.
* Added Makefile rules to generate final report.

------------------------------------------------------------------------
r336 | ChadJiang | 2007-12-15 22:34:26 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   M /trunk/test/examples/phy4.in

add in function
------------------------------------------------------------------------
r335 | ChadJiang | 2007-12-15 22:31:04 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   M /trunk/test/examples/phy4.in

add in function
------------------------------------------------------------------------
r334 | ChadJiang | 2007-12-15 22:05:05 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   M /trunk/test/examples/phy3.in
   M /trunk/test/examples/phy4.in

add alias,nprint, getNumber, getUnit
------------------------------------------------------------------------
r333 | the.stuart.sierra | 2007-12-15 17:01:57 -0500 (Sat, 15 Dec 2007) | 2 line
s
Changed paths:
   A /trunk/src/physicalc/ControlSignal.java

* Reinstated ControlSignal.java

------------------------------------------------------------------------
r332 | ChadJiang | 2007-12-15 15:38:08 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/rundemocode

only run examples/phy*.in
------------------------------------------------------------------------
r331 | ChadJiang | 2007-12-15 15:00:09 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   R /trunk/test/examples/phy1.in
   R /trunk/test/examples/phy1.out

factorial loop test
------------------------------------------------------------------------
r330 | ChadJiang | 2007-12-15 14:59:24 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   M /trunk/test/examples/phy1.out
```

```
factorial loop test
------------------------------------------------------------------------
r329 | ChadJiang | 2007-12-15 13:55:31 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/test/examples/phy3.in
   A /trunk/test/examples/phy3.out

replace phy1
------------------------------------------------------------------------
r328 | ChadJiang | 2007-12-15 12:53:22 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   R /trunk/test/examples/phy1.in
   R /trunk/test/examples/phy4.in

Unit test program
------------------------------------------------------------------------
r327 | ChadJiang | 2007-12-15 12:44:47 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   M /trunk/si.phy

change some unit to make sure program can run
------------------------------------------------------------------------
r326 | brianwfoo | 2007-12-15 12:03:57 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PNumber.java

Added a file remotely
------------------------------------------------------------------------
r325 | brianwfoo | 2007-12-15 12:03:29 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r324 | brianwfoo | 2007-12-15 12:02:53 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/ControlSignal.java
   D /trunk/src/physicalc/PNumber.java
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r323 | brianwfoo | 2007-12-15 11:36:21 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PNumber.java

Added a file remotely
------------------------------------------------------------------------
r322 | brianwfoo | 2007-12-15 11:35:59 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PNumber.java

Removed file/folder
------------------------------------------------------------------------
r321 | ChadJiang | 2007-12-15 11:34:38 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   M /trunk/test/examples/phy1.in

Pi / x and Pi * x^-1
------------------------------------------------------------------------
r320 | ChadJiang | 2007-12-15 11:22:55 -0500 (Sat, 15 Dec 2007) | 1 line
```



```
Changed paths:
   R /trunk/test/examples/phy4.in

G use multiply
------------------------------------------------------------------------
r319 | brianwfoo | 2007-12-15 11:11:36 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r318 | brianwfoo | 2007-12-15 11:11:14 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r317 | ChadJiang | 2007-12-15 10:27:36 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/test/examples/phy2.in
   A /trunk/test/examples/phy2.out

for unit ^ test
------------------------------------------------------------------------
r316 | brianwfoo | 2007-12-15 09:25:13 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/In.java

Added a file remotely
------------------------------------------------------------------------
r315 | brianwfoo | 2007-12-15 09:24:56 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PNumber.java

Added a file remotely
------------------------------------------------------------------------
r314 | brianwfoo | 2007-12-15 09:24:36 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r313 | brianwfoo | 2007-12-15 09:24:12 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnitPair.java

Added a file remotely
------------------------------------------------------------------------
r312 | brianwfoo | 2007-12-15 09:23:45 -0500 (Sat, 15 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/In.java
   D /trunk/src/physicalc/PNumber.java
   D /trunk/src/physicalc/PUnit.java
   D /trunk/src/physicalc/PUnitPair.java

Removed file/folder
------------------------------------------------------------------------
r311 | ChadJiang | 2007-12-14 23:04:10 -0500 (Fri, 14 Dec 2007) | 1 line
Changed paths:
   A /trunk/test/examples/phy4.in
   A /trunk/test/examples/phy4.out
```

```
caret ^ not support for complex unit
------------------------------------------------------------------------
r310 | ChadJiang | 2007-12-14 17:37:54 -0500 (Fri, 14 Dec 2007) | 1 line
Changed paths:
   M /trunk/test/examples/phy1.in

Sun mass caculation Sample
------------------------------------------------------------------------
r309 | the.stuart.sierra | 2007-12-14 01:11:03 -0500 (Fri, 14 Dec 2007) | 3 line
s
Changed paths:
   M /trunk/Makefile

Added more Makefile rules for generating final report,
including using a2ps for source files.

------------------------------------------------------------------------
r308 | ChadJiang | 2007-12-13 00:26:39 -0500 (Thu, 13 Dec 2007) | 1 line
Changed paths:
   A /trunk/test/examples/phy1.in
   A /trunk/test/examples/phy1.out

 sample program1
------------------------------------------------------------------------
r307 | ChadJiang | 2007-12-13 00:02:15 -0500 (Thu, 13 Dec 2007) | 1 line
Changed paths:
   R /trunk/test/examples/unit7.in

 unit test
------------------------------------------------------------------------
r306 | ChadJiang | 2007-12-12 23:52:34 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   A /trunk/test/examples/unit7.in
   A /trunk/test/examples/unit7.out

 unit test
------------------------------------------------------------------------
r305 | ChadJiang | 2007-12-12 22:44:18 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   M /trunk/src/physicalc/Arith.java
   M /trunk/src/physicalc/For.java
   M /trunk/src/physicalc/If.java
   M /trunk/src/physicalc/Not.java
   M /trunk/src/physicalc/While.java

add author
------------------------------------------------------------------------
r304 | the.stuart.sierra | 2007-12-12 21:00:33 -0500 (Wed, 12 Dec 2007) | 2 line
s
Changed paths:
   A /trunk/test/examples/unit6.in
   A /trunk/test/examples/unit6.out

Added example test using units.

------------------------------------------------------------------------
r303 | brianwfoo | 2007-12-12 19:29:02 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
```



```
r302 | brianwfoo | 2007-12-12 19:28:35 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r301 | brianwfoo | 2007-12-12 19:21:14 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r300 | brianwfoo | 2007-12-12 19:20:50 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r299 | brianwfoo | 2007-12-12 19:20:03 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r298 | brianwfoo | 2007-12-12 19:19:39 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r297 | brianwfoo | 2007-12-12 19:08:34 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PNumber.java

Added a file remotely
------------------------------------------------------------------------
r296 | brianwfoo | 2007-12-12 19:08:15 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r295 | brianwfoo | 2007-12-12 19:07:53 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PNumber.java
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r294 | ChadJiang | 2007-12-12 14:12:18 -0500 (Wed, 12 Dec 2007) | 1 line
Changed paths:
   M /trunk/Makefile

 add Exitfunction
------------------------------------------------------------------------
r293 | ChadJiang | 2007-12-11 15:08:51 -0500 (Tue, 11 Dec 2007) | 1 line
Changed paths:
   A /trunk/test/examples/UnitTime.in
   A /trunk/test/examples/UnitTime.out

simple test unit multiply unit
------------------------------------------------------------------------
```

```
r292 | ChadJiang | 2007-12-11 15:07:31 -0500 (Tue, 11 Dec 2007) | 1 line
Changed paths:
   A /trunk/test/examples/mathexample1.in
   A /trunk/test/examples/mathexample1.out

example in usermanual
------------------------------------------------------------------------
r291 | the.stuart.sierra | 2007-12-10 22:06:36 -0500 (Mon, 10 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/Makefile

Added Makefile rules for running pdflatex on final report.

------------------------------------------------------------------------
r290 | the.stuart.sierra | 2007-12-10 21:56:33 -0500 (Mon, 10 Dec 2007) | 2 line
s
Changed paths:
   A /trunk/report
   A /trunk/report/bibliography.tex
   A /trunk/report/finalreport.tex
   A /trunk/report/functions.tex
   A /trunk/report/intro.tex
   A /trunk/report/refman.tex

Added beginning of the final report, in LaTeX.

------------------------------------------------------------------------
r289 | the.stuart.sierra | 2007-12-09 17:32:28 -0500 (Sun, 09 Dec 2007) | 2 line
s
Changed paths:
   A /trunk/test/examples/unit5.in
   A /trunk/test/examples/unit5.out

Added test for units like meter/second^2

------------------------------------------------------------------------
r288 | brianwfoo | 2007-12-08 16:23:37 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/In.java

Added a file remotely
------------------------------------------------------------------------
r287 | brianwfoo | 2007-12-08 16:23:11 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/UnitDef.java

Added a file remotely
------------------------------------------------------------------------
r286 | brianwfoo | 2007-12-08 16:22:53 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Unit.java

Added a file remotely
------------------------------------------------------------------------
r285 | brianwfoo | 2007-12-08 16:22:29 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/In.java
   D /trunk/src/physicalc/Unit.java
   D /trunk/src/physicalc/UnitDef.java

Removed file/folder
```



```
------------------------------------------------------------------------
r284 | ChadJiang | 2007-12-08 15:26:52 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   M /trunk/src/physicalc/ExitFunction.java

fix bug
------------------------------------------------------------------------
r283 | the.stuart.sierra | 2007-12-08 15:07:02 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   A /trunk/test/examples/unit3.in
   A /trunk/test/examples/unit3.out
   A /trunk/test/examples/unit4.in
   A /trunk/test/examples/unit4.out

More unit tests.

------------------------------------------------------------------------
r282 | brianwfoo | 2007-12-08 15:01:24 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/In.java

Added a file remotely
------------------------------------------------------------------------
r281 | brianwfoo | 2007-12-08 15:01:07 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/In.java

Removed file/folder
------------------------------------------------------------------------
r280 | the.stuart.sierra | 2007-12-08 14:38:52 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/Access.java
   M /trunk/src/physicalc/Block.java
   M /trunk/src/physicalc/Datum.java
   M /trunk/src/physicalc/ExitFunction.java
   M /trunk/src/physicalc/ExprList.java
   M /trunk/src/physicalc/FunCall.java
   M /trunk/src/physicalc/GetNumberFunction.java
   M /trunk/src/physicalc/GetUnitFunction.java
   M /trunk/src/physicalc/Id.java
   M /trunk/src/physicalc/If.java
   M /trunk/src/physicalc/Interpreter.java
   M /trunk/src/physicalc/Literal.java
   M /trunk/src/physicalc/Load.java
   M /trunk/src/physicalc/Main.java
   M /trunk/src/physicalc/NPrintFunction.java
   M /trunk/src/physicalc/PList.java
   M /trunk/src/physicalc/ParamList.java
   M /trunk/src/physicalc/PrintFunction.java
   M /trunk/src/physicalc/Program.java
   M /trunk/src/physicalc/Unit.java
   M /trunk/src/physicalc/UnitDef.java

Commented out debugging System.err.println lines in all sources.

------------------------------------------------------------------------
r279 | the.stuart.sierra | 2007-12-08 14:36:03 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   A /trunk/otherunits.phy
```

```
Added other units from Ici.

------------------------------------------------------------------------
r278 | the.stuart.sierra | 2007-12-08 14:20:07 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/si.phy

Added lots more SI units, generated from a script.

------------------------------------------------------------------------
r277 | digitalfobulous | 2007-12-08 14:01:30 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/test/examples/funcTest.in
   A /trunk/test/examples/funcTest.out

test
------------------------------------------------------------------------
r276 | ChadJiang | 2007-12-08 13:53:57 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/test/examples/exit.in
   A /trunk/test/examples/exit.out

exit function test
------------------------------------------------------------------------
r275 | ChadJiang | 2007-12-08 13:52:29 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/ExitFunction.java

exit function
------------------------------------------------------------------------
r274 | brianwfoo | 2007-12-08 13:47:29 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnitPair.java

Added a file remotely
------------------------------------------------------------------------
r273 | the.stuart.sierra | 2007-12-08 13:47:21 -0500 (Sat, 08 Dec 2007) | 3 line
s
Changed paths:
   M /trunk/src/physicalc/AliasDef.java
   M /trunk/src/physicalc/Id.java
   M /trunk/src/physicalc/UnitDef.java
   A /trunk/test/examples/alias.in
   A /trunk/test/examples/alias.out

* Added test for aliases.
* Fixed UnitDef, AliasDef, Id

------------------------------------------------------------------------
r272 | brianwfoo | 2007-12-08 13:47:10 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnitPair.java

Removed file/folder
------------------------------------------------------------------------
r271 | digitalfobulous | 2007-12-08 13:38:45 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   M /trunk/src/physicalc/AliasDef.java
   M /trunk/src/physicalc/ConstantDef.java
```



```
variable and constant
------------------------------------------------------------------------
r270 | brianwfoo | 2007-12-08 13:28:55 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r269 | brianwfoo | 2007-12-08 13:28:32 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r268 | digitalfobulous | 2007-12-08 13:25:26 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   M /trunk/src/physicalc/AliasDef.java

variable and constant
------------------------------------------------------------------------
r267 | the.stuart.sierra | 2007-12-08 13:17:33 -0500 (Sat, 08 Dec 2007) | 3 line
s
Changed paths:
   M /trunk/src/physicalc/Load.java
   A /trunk/test/examples/const1.in
   A /trunk/test/examples/const1.out

Tests for Constant.
Load.java implemented.

------------------------------------------------------------------------
r266 | brianwfoo | 2007-12-08 13:16:37 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/GetUnitFunction.java

Added a file remotely
------------------------------------------------------------------------
r265 | brianwfoo | 2007-12-08 13:16:19 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/GetUnitFunction.java

Removed file/folder
------------------------------------------------------------------------
r264 | brianwfoo | 2007-12-08 13:15:58 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/GetNumberFunction.java

Added a file remotely
------------------------------------------------------------------------
r263 | brianwfoo | 2007-12-08 13:15:39 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/GetNumberFunction.java

Removed file/folder
------------------------------------------------------------------------
r262 | brianwfoo | 2007-12-08 13:02:05 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/GetUnitFunction.java

Added a file remotely
------------------------------------------------------------------------
r261 | brianwfoo | 2007-12-08 13:01:48 -0500 (Sat, 08 Dec 2007) | 1 line
```

```
Changed paths:
   A /trunk/src/physicalc/GetNumberFunction.java

Added a file remotely
------------------------------------------------------------------------
r260 | brianwfoo | 2007-12-08 13:01:29 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/GetNumberFunction.java
   D /trunk/src/physicalc/GetUnitFunction.java

Removed file/folder
------------------------------------------------------------------------
r259 | the.stuart.sierra | 2007-12-08 12:50:23 -0500 (Sat, 08 Dec 2007) | 3 line
s
Changed paths:
   M /trunk/src/physicalc/Interpreter.java
   A /trunk/test/examples/for2.in
   A /trunk/test/examples/for2.out
   A /trunk/test/examples/getNumber.in
   A /trunk/test/examples/getNumber.out
   A /trunk/test/examples/getUnit.in
   A /trunk/test/examples/getUnit.out

Tests for for, getNumber, getUnit.
Updated builtin symbols in Interpreter.

------------------------------------------------------------------------
r258 | brianwfoo | 2007-12-08 12:42:40 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/UnitDef.java

Added a file remotely
------------------------------------------------------------------------
r257 | brianwfoo | 2007-12-08 12:42:23 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Unit.java

Added a file remotely
------------------------------------------------------------------------
r256 | brianwfoo | 2007-12-08 12:41:43 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/Unit.java
   D /trunk/src/physicalc/UnitDef.java

Removed file/folder
------------------------------------------------------------------------
r255 | the.stuart.sierra | 2007-12-08 12:41:07 -0500 (Sat, 08 Dec 2007) | 3 line
s
Changed paths:
   M /trunk/src/physicalc/For.java
   A /trunk/test/examples/for1.in
   A /trunk/test/examples/for1.out
   A /trunk/test/examples/in.in
   A /trunk/test/examples/in.out

* More example tests.
* Fixed For.java so it compiles.

------------------------------------------------------------------------
r254 | ChadJiang | 2007-12-08 12:36:45 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   M /trunk/src/physicalc/For.java
```



```

For class
------------------------------------------------------------------------
r253 | the.stuart.sierra | 2007-12-08 12:35:40 -0500 (Sat, 08 Dec 2007) | 4 line
s
Changed paths:
   M /trunk/Makefile
   M /trunk/src/physicalc/Interpreter.java
   M /trunk/src/physicalc/ToStringFunction.java
   A /trunk/test/examples/nprint.in
   A /trunk/test/examples/nprint.out
   A /trunk/test/examples/toString.in
   A /trunk/test/examples/toString.out
   A /trunk/test/examples/unary.in
   A /trunk/test/examples/unary.out
   A /trunk/test/examples/unit2.in
   A /trunk/test/examples/unit2.out

* More example tests
* Fixed ToStringFunction
* Updated Makefile

------------------------------------------------------------------------
r252 | digitalfobulous | 2007-12-08 12:21:54 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/NPrintFunction.java

variable and constant
------------------------------------------------------------------------
r251 | digitalfobulous | 2007-12-08 12:05:16 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   M /trunk/src/physicalc/ToStringFunction.java

variable and constant
------------------------------------------------------------------------
r250 | the.stuart.sierra | 2007-12-08 12:04:04 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/Interpreter.java
   M /trunk/src/physicalc/ToIntFunction.java
   A /trunk/test/examples/set1.in
   A /trunk/test/examples/set1.out
   A /trunk/test/examples/toInt.in
   A /trunk/test/examples/toInt.out
   A /trunk/test/examples/while2.in
   A /trunk/test/examples/while2.out
   A /trunk/test/examples/while3.in
   A /trunk/test/examples/while3.out
   A /trunk/test/examples/while4.in
   A /trunk/test/examples/while4.out

More example tests; fixed ToIntFunction.java

------------------------------------------------------------------------
r249 | digitalfobulous | 2007-12-08 12:03:09 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/ToStringFunction.java

variable and constant
------------------------------------------------------------------------
r248 | digitalfobulous | 2007-12-08 11:40:21 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
```

```
   M /trunk/src/physicalc/Constant.java
   M /trunk/src/physicalc/Variable.java

variable and constant
------------------------------------------------------------------------
r247 | the.stuart.sierra | 2007-12-08 11:38:58 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/Makefile

Added Makefile line to delete exampes/*.actual on 'make clean'

------------------------------------------------------------------------
r246 | the.stuart.sierra | 2007-12-08 11:37:06 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/Set.java

Implemented Set.java.

------------------------------------------------------------------------
r245 | the.stuart.sierra | 2007-12-08 11:36:37 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/Id.java

Fixed Id.java so it compiles.

------------------------------------------------------------------------
r244 | digitalfobulous | 2007-12-08 11:24:38 -0500 (Sat, 08 Dec 2007) | 5 lines
Changed paths:
   M /trunk/src/physicalc/Id.java


dssfgghhf-This line, and those below, will be ignored--

M    physicalc/Id.java

------------------------------------------------------------------------
r243 | the.stuart.sierra | 2007-12-08 10:49:50 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   A /trunk/test/examples/while1.in
   A /trunk/test/examples/while1.out

Added simple while1 example test for while loop with a break.

------------------------------------------------------------------------
r242 | the.stuart.sierra | 2007-12-08 10:47:31 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/Makefile

Added Constant.java to Makefile.

------------------------------------------------------------------------
r241 | the.stuart.sierra | 2007-12-08 10:39:26 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   A /trunk/runexample

Added runexample script to run a single example test file.
```



```
------------------------------------------------------------------------
r240 | the.stuart.sierra | 2007-12-08 10:31:40 -0500 (Sat, 08 Dec 2007) | 2 line
s
Changed paths:
   A /trunk/test/examples/unit1.in
   A /trunk/test/examples/unit1.out

Added test for defining and printing base unit.

------------------------------------------------------------------------
r239 | brianwfoo | 2007-12-08 08:43:50 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/UnitDef.java

Added a file remotely
------------------------------------------------------------------------
r238 | brianwfoo | 2007-12-08 08:43:33 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Unit.java

Added a file remotely
------------------------------------------------------------------------
r237 | brianwfoo | 2007-12-08 08:43:18 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnitPair.java

Added a file remotely
------------------------------------------------------------------------
r236 | brianwfoo | 2007-12-08 08:42:56 -0500 (Sat, 08 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnitPair.java
   D /trunk/src/physicalc/Unit.java
   D /trunk/src/physicalc/UnitDef.java

Removed file/folder
------------------------------------------------------------------------
r235 | ChadJiang | 2007-12-04 22:48:26 -0500 (Tue, 04 Dec 2007) | 1 line
Changed paths:
   M /trunk/src/physicalc/While.java

new version
------------------------------------------------------------------------
r234 | the.stuart.sierra | 2007-12-04 21:37:18 -0500 (Tue, 04 Dec 2007) | 4 line
s
Changed paths:
   A /trunk/runexamples
   A /trunk/test/examples
   A /trunk/test/examples/print1.in
   A /trunk/test/examples/print1.out

Added test/examples.
Added test/examples/print1.in and print1.out.
Added runexamples script.

------------------------------------------------------------------------
r233 | the.stuart.sierra | 2007-12-04 20:55:19 -0500 (Tue, 04 Dec 2007) | 3 line
s
Changed paths:
   M /trunk/test/InterpreterTest.java

Added tests for While, based on meeting w/ Chad 2007-12-04.
```

```
Need Set implementation to finish testing While.

------------------------------------------------------------------------
r232 | ChadJiang | 2007-12-04 11:22:43 -0500 (Tue, 04 Dec 2007) | 1 line
Changed paths:
   M /trunk/src/physicalc/While.java

it can pass compile, but not sure for testing
------------------------------------------------------------------------
r231 | the.stuart.sierra | 2007-12-03 09:03:00 -0500 (Mon, 03 Dec 2007) | 3 line
s
Changed paths:
   M /trunk/src/grammar.g
   M /trunk/src/physicalc/UnitDef.java

* Fixed parser & tree walker to support base & derived units.
* Added constructor to UnitDef for base units.

------------------------------------------------------------------------
r230 | the.stuart.sierra | 2007-12-03 09:02:06 -0500 (Mon, 03 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/test/InterpreterTest.java

* Committed tests on Unary minus operator.

------------------------------------------------------------------------
r229 | the.stuart.sierra | 2007-12-03 08:42:08 -0500 (Mon, 03 Dec 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/Unary.java

Added Unary.java from Ici, corrected one line.

------------------------------------------------------------------------
r228 | brianwfoo | 2007-12-03 03:40:50 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/UnitDef.java

Added a file remotely
------------------------------------------------------------------------
r227 | brianwfoo | 2007-12-03 03:40:26 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Unit.java

Added a file remotely
------------------------------------------------------------------------
r226 | brianwfoo | 2007-12-03 03:39:53 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/Unit.java
   D /trunk/src/physicalc/UnitDef.java

Removed file/folder
------------------------------------------------------------------------
r225 | ChadJiang | 2007-12-03 03:23:08 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   M /trunk/src/physicalc/While.java

not sure correct
------------------------------------------------------------------------
r224 | brianwfoo | 2007-12-03 03:21:44 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
```



```
   A /trunk/src/physicalc/In.java

Added a file remotely
------------------------------------------------------------------------
r223 | brianwfoo | 2007-12-03 03:21:22 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r222 | brianwfoo | 2007-12-03 03:20:52 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/In.java
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r221 | brianwfoo | 2007-12-03 01:24:03 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/GetUnitFunction.java

Added a file remotely
------------------------------------------------------------------------
r220 | brianwfoo | 2007-12-03 01:23:41 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/GetNumberFunction.java

Added a file remotely
------------------------------------------------------------------------
r219 | brianwfoo | 2007-12-03 01:23:22 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/ToIntFunction.java

Added a file remotely
------------------------------------------------------------------------
r218 | brianwfoo | 2007-12-03 01:22:59 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Unit.java

Added a file remotely
------------------------------------------------------------------------
r217 | brianwfoo | 2007-12-03 01:19:26 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   A /trunk/Makefile

Added a file remotely
------------------------------------------------------------------------
r216 | brianwfoo | 2007-12-03 01:19:08 -0500 (Mon, 03 Dec 2007) | 1 line
Changed paths:
   D /trunk/Makefile

Removed file/folder
------------------------------------------------------------------------
r215 | the.stuart.sierra | 2007-12-01 20:58:13 -0500 (Sat, 01 Dec 2007) | 8 line
s
Changed paths:
   M /trunk/Makefile
   M /trunk/src/grammar.g
   A /trunk/src/physicalc/Access.java
   A /trunk/src/physicalc/Constant.java
   M /trunk/src/physicalc/Id.java
   A /trunk/src/physicalc/LValue.java
```

```
   M /trunk/src/physicalc/PList.java
   M /trunk/src/physicalc/PNumber.java
   M /trunk/src/physicalc/Set.java
   A /trunk/src/physicalc/Unary.java
   M /trunk/test/InterpreterTest.java

* Implemented Access completely.
* Implmeneted Id as it relates to Set.
* Modified PList and PNumber to support Access.
* Templated Set, Constant, Id, and Unary.
* Finished tree walker.
* Added a bunch of interpreter tests.
* Updated Makefile

------------------------------------------------------------------------
r214 | the.stuart.sierra | 2007-12-01 10:10:59 -0500 (Sat, 01 Dec 2007) | 3 lines
Changed paths:
   M /trunk/test/InterpreterTest.java

Added tests for simple arithmetic, relational operators, and
the "if" statement.

------------------------------------------------------------------------
r213 | brianwfoo | 2007-11-30 18:21:38 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/FunctionDef.java

Added a file remotely
------------------------------------------------------------------------
r212 | brianwfoo | 2007-11-30 18:21:17 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/FunctionDef.java

Removed file/folder
------------------------------------------------------------------------
r211 | brianwfoo | 2007-11-30 18:13:43 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Return.java

Added a file remotely
------------------------------------------------------------------------
r210 | brianwfoo | 2007-11-30 18:13:21 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Function.java

Added a file remotely
------------------------------------------------------------------------
r209 | brianwfoo | 2007-11-30 18:13:03 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/FunCall.java

Added a file remotely
------------------------------------------------------------------------
r208 | brianwfoo | 2007-11-30 18:12:37 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/FunCall.java
   D /trunk/src/physicalc/Function.java
   D /trunk/src/physicalc/Return.java

Removed file/folder
------------------------------------------------------------------------
```



```
r207 | the.stuart.sierra | 2007-11-30 18:03:37 -0500 (Fri, 30 Nov 2007) | 2 lines
Changed paths:
   M /trunk/test/InterpreterTest.java

Enabled "Hello, World!" test in InterpreterTest.

------------------------------------------------------------------------
r206 | the.stuart.sierra | 2007-11-30 17:36:02 -0500 (Fri, 30 Nov 2007) | 3 lines
Changed paths:
   M /trunk/src/grammar.g
   M /trunk/src/physicalc/Block.java
   M /trunk/src/physicalc/FunCall.java
   M /trunk/src/physicalc/If.java
   M /trunk/src/physicalc/Literal.java
   M /trunk/src/physicalc/PrintFunction.java

grammar.g: fixed bug in how tree walker handles nodes inside a BLOCK
Added debugging statements to various eval() methods.

------------------------------------------------------------------------
r205 | the.stuart.sierra | 2007-11-30 17:21:26 -0500 (Fri, 30 Nov 2007) | 5 lines
Changed paths:
   M /trunk/Makefile
   M /trunk/src/physicalc/Function.java
   M /trunk/src/physicalc/Interpreter.java
   A /trunk/src/physicalc/PrintFunction.java

* Added PrintFunction to implement built-in "print()"
* Added protected constructor to Function for base classes.
* Added builtin "print()" to Interpreter global symbol table.
* Updated Makefile.

------------------------------------------------------------------------
r204 | the.stuart.sierra | 2007-11-30 17:06:58 -0500 (Fri, 30 Nov 2007) | 3 lines
Changed paths:
   M /trunk/src/physicalc/FunCall.java
   M /trunk/src/physicalc/Function.java

Fixed FunCall and Function to use current local symbol table
when evaluating arguments.

------------------------------------------------------------------------
r203 | the.stuart.sierra | 2007-11-30 16:52:33 -0500 (Fri, 30 Nov 2007) | 2 lines
Changed paths:
   M /trunk/src/physicalc/ReturnSignal.java

Added documentation to ReturnSignal.java

------------------------------------------------------------------------
r202 | brianwfoo | 2007-11-30 02:04:44 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Return.java

Added a file remotely
------------------------------------------------------------------------
r201 | brianwfoo | 2007-11-30 02:04:10 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
```

```
   D /trunk/src/physicalc/Return.java

Removed file/folder
------------------------------------------------------------------------
r200 | brianwfoo | 2007-11-30 02:03:43 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Function.java

Added a file remotely
------------------------------------------------------------------------
r199 | brianwfoo | 2007-11-30 02:03:10 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/FunCall.java

Added a file remotely
------------------------------------------------------------------------
r198 | brianwfoo | 2007-11-30 02:02:41 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/Function.java

Removed file/folder
------------------------------------------------------------------------
r197 | brianwfoo | 2007-11-30 02:02:30 -0500 (Fri, 30 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/FunCall.java

Removed file/folder
------------------------------------------------------------------------
r196 | the.stuart.sierra | 2007-11-29 20:12:57 -0500 (Thu, 29 Nov 2007) | 4 line
s
Changed paths:
   M /trunk/Makefile
   M /trunk/src/physicalc/Interpreter.java
   M /trunk/test/InterpreterTest.java

* Updated InterpreterTest and Interpreter to correctly
  redirect standard output.
* Updated Makefile to compile tests correctly.

------------------------------------------------------------------------
r195 | the.stuart.sierra | 2007-11-29 19:13:53 -0500 (Thu, 29 Nov 2007) | 3 line
s
Changed paths:
   M /trunk/Makefile
   M /trunk/src/grammar.g
   A /trunk/src/physicalc/In.java

* Added skeleton In class for "in" operator.
* Updated grammar and Makefile.

------------------------------------------------------------------------
r194 | the.stuart.sierra | 2007-11-29 19:05:01 -0500 (Thu, 29 Nov 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/Break.java
   M /trunk/src/physicalc/Next.java

Implemented Next and Break statements.

------------------------------------------------------------------------
r193 | ChadJiang | 2007-11-29 09:39:11 -0500 (Thu, 29 Nov 2007) | 1 line
Changed paths:
```



```
   M /trunk/src/physicalc/While.java

While class
------------------------------------------------------------------------
r192 | the.stuart.sierra | 2007-11-26 19:35:26 -0500 (Mon, 26 Nov 2007) | 4 line
s
Changed paths:
   M /trunk/Makefile
   M /trunk/src/grammar.g
   A /trunk/src/physicalc/AliasDef.java
   A /trunk/src/physicalc/ConstantDef.java
   A /trunk/src/physicalc/FunctionDef.java
   M /trunk/src/physicalc/ParamList.java
   A /trunk/src/physicalc/UnitDef.java

* Added skeleton classes for all definitions.
* Added definitions to tree walker.
* Updated Makefile.

------------------------------------------------------------------------
r191 | the.stuart.sierra | 2007-11-26 19:02:58 -0500 (Mon, 26 Nov 2007) | 3 line
s
Changed paths:
   M /trunk/Makefile
   M /trunk/src/grammar.g

* Modifications to grammar.g for correct handling of if/elsif/else.
* Put TryParser and ParseFile back in Makefile.

------------------------------------------------------------------------
r190 | the.stuart.sierra | 2007-11-26 18:32:30 -0500 (Mon, 26 Nov 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/If.java

* Fixed missing brace in If.java

------------------------------------------------------------------------
r189 | ChadJiang | 2007-11-25 22:52:33 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   M /trunk/src/physicalc/If.java

modify If.java
------------------------------------------------------------------------
r188 | the.stuart.sierra | 2007-11-25 22:14:26 -0500 (Sun, 25 Nov 2007) | 4 line
s
Changed paths:
   M /trunk/Makefile
   A /trunk/src/physicalc/BreakSignal.java
   A /trunk/src/physicalc/ControlSignal.java
   A /trunk/src/physicalc/NextSignal.java
   D /trunk/src/physicalc/PVector.java
   A /trunk/src/physicalc/ReturnSignal.java

* Created skeleton ControlSignal classes.
* Removed PVector.
* Updated Makefile.

------------------------------------------------------------------------
r187 | the.stuart.sierra | 2007-11-25 21:22:55 -0500 (Sun, 25 Nov 2007) | 4 line
s
Changed paths:
```

```
   M /trunk/Makefile
   M /trunk/src/grammar.g
   A /trunk/src/physicalc/Break.java
   A /trunk/src/physicalc/For.java
   A /trunk/src/physicalc/If.java
   M /trunk/src/physicalc/Interpreter.java
   A /trunk/src/physicalc/Load.java
   A /trunk/src/physicalc/Next.java
   A /trunk/src/physicalc/Return.java
   A /trunk/src/physicalc/Set.java
   A /trunk/src/physicalc/While.java

* Added skeleton classes for all statement nodes.
* Adjusted interpreter to print "null" on null pointer returned.
* Updated Makefile.

------------------------------------------------------------------------
r186 | the.stuart.sierra | 2007-11-25 19:11:55 -0500 (Sun, 25 Nov 2007) | 5 line
s
Changed paths:
   M /trunk/Makefile
   M /trunk/src/grammar.g
   M /trunk/src/physicalc/Interpreter.java
   A /trunk/src/physicalc/Program.java

* Added Program class; changed Interpreter to use it.
* Added program rule to grammar.g, paving the way for
  definitions and statements.
* Updated Makefile.

------------------------------------------------------------------------
r185 | the.stuart.sierra | 2007-11-25 18:57:45 -0500 (Sun, 25 Nov 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/SymbolTable.java

SymbolTable.java: added documentation.

------------------------------------------------------------------------
r184 | the.stuart.sierra | 2007-11-25 18:44:46 -0500 (Sun, 25 Nov 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/PList.java

PList.java: line 23: added type cast to make it compile.

------------------------------------------------------------------------
r183 | brianwfoo | 2007-11-25 17:06:50 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/TypeError.java

Added a file remotely
------------------------------------------------------------------------
r182 | brianwfoo | 2007-11-25 17:06:33 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnitPair.java

Added a file remotely
------------------------------------------------------------------------
r181 | brianwfoo | 2007-11-25 17:06:17 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java
```



```

Added a file remotely
------------------------------------------------------------------------
r180 | brianwfoo | 2007-11-25 17:05:56 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/TypeError.java

Removed file/folder
------------------------------------------------------------------------
r179 | brianwfoo | 2007-11-25 17:05:42 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnitPair.java

Removed file/folder
------------------------------------------------------------------------
r178 | brianwfoo | 2007-11-25 17:05:31 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r177 | brianwfoo | 2007-11-25 17:05:13 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PString.java

Added a file remotely
------------------------------------------------------------------------
r176 | brianwfoo | 2007-11-25 17:04:54 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PNumber.java

Added a file remotely
------------------------------------------------------------------------
r175 | brianwfoo | 2007-11-25 17:04:37 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PList.java

Added a file remotely
------------------------------------------------------------------------
r174 | brianwfoo | 2007-11-25 17:04:17 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PString.java

Removed file/folder
------------------------------------------------------------------------
r173 | brianwfoo | 2007-11-25 17:04:01 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PNumber.java

Removed file/folder
------------------------------------------------------------------------
r172 | brianwfoo | 2007-11-25 17:03:47 -0500 (Sun, 25 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PList.java

Removed file/folder
------------------------------------------------------------------------
r171 | the.stuart.sierra | 2007-11-25 12:33:39 -0500 (Sun, 25 Nov 2007) | 5 line
s
Changed paths:
   M /trunk/Makefile
   M /trunk/src/grammar.g
```

```
   A /trunk/src/physicalc/Block.java
   M /trunk/src/physicalc/Datum.java
   M /trunk/src/physicalc/ExprList.java
   A /trunk/src/physicalc/FunCall.java
   M /trunk/src/physicalc/Id.java
   M /trunk/src/physicalc/Variable.java

* More skeleton classes for FunCall, Variable, Block, and Id.
* More rules in tree walker (most expressions implemented).
* Makefile updated.
* Using System.err for all debugging output.

------------------------------------------------------------------------
r170 | the.stuart.sierra | 2007-11-24 18:24:10 -0500 (Sat, 24 Nov 2007) | 2 line
s
Changed paths:
   A /trunk/src/physicalc/Not.java

Added Not node class (from Ici).

------------------------------------------------------------------------
r169 | the.stuart.sierra | 2007-11-24 11:33:54 -0500 (Sat, 24 Nov 2007) | 2 line
s
Changed paths:
   A /trunk/src/physicalc/ExprList.java
   A /trunk/src/physicalc/Function.java
   A /trunk/src/physicalc/Id.java
   A /trunk/src/physicalc/ParamList.java
   A /trunk/src/physicalc/Variable.java

Added templates for Id, Function, and Variable.

------------------------------------------------------------------------
r168 | the.stuart.sierra | 2007-11-23 22:46:45 -0500 (Fri, 23 Nov 2007) | 2 line
s
Changed paths:
   M /trunk/Makefile
   A /trunk/src/physicalc/Or.java

Added Or.java (from Ici) and adjusted Makefile.

------------------------------------------------------------------------
r167 | brianwfoo | 2007-11-22 01:50:36 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnitPair.java

Added a file remotely
------------------------------------------------------------------------
r166 | brianwfoo | 2007-11-22 01:50:15 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnitPair.java

Removed file/folder
------------------------------------------------------------------------
r165 | brianwfoo | 2007-11-22 01:50:02 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r164 | brianwfoo | 2007-11-22 01:49:46 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
```



```
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
r163 | brianwfoo | 2007-11-22 01:49:30 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PString.java

Added a file remotely
------------------------------------------------------------------------
r162 | brianwfoo | 2007-11-22 01:49:10 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PString.java

Removed file/folder
------------------------------------------------------------------------
r161 | brianwfoo | 2007-11-22 01:48:55 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PNumber.java

Added a file remotely
------------------------------------------------------------------------
r160 | brianwfoo | 2007-11-22 01:48:38 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PNumber.java

Removed file/folder
------------------------------------------------------------------------
r159 | brianwfoo | 2007-11-22 01:47:10 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/TryDatum.java

Added a file remotely
------------------------------------------------------------------------
r158 | brianwfoo | 2007-11-22 01:46:52 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/TryDatum.java

Removed file/folder
------------------------------------------------------------------------
r157 | brianwfoo | 2007-11-22 01:46:27 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   A /trunk/Makefile

Added a file remotely
------------------------------------------------------------------------
r156 | brianwfoo | 2007-11-22 01:46:10 -0500 (Thu, 22 Nov 2007) | 1 line
Changed paths:
   D /trunk/Makefile

Removed file/folder
------------------------------------------------------------------------
r155 | the.stuart.sierra | 2007-11-19 23:17:23 -0500 (Mon, 19 Nov 2007) | 4 line
s
Changed paths:
   M /trunk/src/grammar.g
   M /trunk/src/physicalc/Datum.java
   M /trunk/src/physicalc/Interpreter.java
   M /trunk/src/physicalc/Main.java
   M /trunk/src/physicalc/PNumber.java

* Rough working Interpreter and Main classes.
```

```
* PNumber#add() modified as example.
* Early tree walker in grammar.g.

------------------------------------------------------------------------
r154 | the.stuart.sierra | 2007-11-19 22:35:40 -0500 (Mon, 19 Nov 2007) | 2 line
s
Changed paths:
   A /trunk/src/physicalc/Arith.java

Added Arith class, from Chad.

------------------------------------------------------------------------
r153 | the.stuart.sierra | 2007-11-19 22:08:10 -0500 (Mon, 19 Nov 2007) | 3 line
s
Changed paths:
   M /trunk/Makefile

Added Chad’s Arith class (slightly corrected) and
updated Makefile.

------------------------------------------------------------------------
r152 | the.stuart.sierra | 2007-11-19 22:04:16 -0500 (Mon, 19 Nov 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/Rel.java

Corrected string comparisons to use "equals" instead of "==".

------------------------------------------------------------------------
r151 | the.stuart.sierra | 2007-11-19 21:55:05 -0500 (Mon, 19 Nov 2007) | 2 line
s
Changed paths:
   M /trunk/Makefile
   A /trunk/src/physicalc/Literal.java

Added Literal node class and updated Makefile.

------------------------------------------------------------------------
r150 | brianwfoo | 2007-11-19 19:45:15 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnitPair.java

Added a file remotely
------------------------------------------------------------------------
r149 | brianwfoo | 2007-11-19 19:44:53 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnitPair.java

Removed file/folder
------------------------------------------------------------------------
r148 | brianwfoo | 2007-11-19 19:44:34 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r147 | brianwfoo | 2007-11-19 19:44:14 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PUnit.java

Removed file/folder
------------------------------------------------------------------------
```



```
r146 | brianwfoo | 2007-11-19 19:43:57 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PString.java

Added a file remotely
------------------------------------------------------------------------
r145 | brianwfoo | 2007-11-19 19:43:32 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PString.java

Removed file/folder
------------------------------------------------------------------------
r144 | brianwfoo | 2007-11-19 19:43:15 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PNumber.java

Added a file remotely
------------------------------------------------------------------------
r143 | brianwfoo | 2007-11-19 19:42:56 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PNumber.java

Removed file/folder
------------------------------------------------------------------------
r142 | brianwfoo | 2007-11-19 19:42:43 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PList.java

Added a file remotely
------------------------------------------------------------------------
r141 | brianwfoo | 2007-11-19 19:42:21 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PList.java

Removed file/folder
------------------------------------------------------------------------
r140 | brianwfoo | 2007-11-19 19:42:06 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PBoolean.java

Added a file remotely
------------------------------------------------------------------------
r139 | brianwfoo | 2007-11-19 19:41:41 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PBoolean.java

Removed file/folder
------------------------------------------------------------------------
r138 | brianwfoo | 2007-11-19 19:41:17 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Datum.java

Added a file remotely
------------------------------------------------------------------------
r137 | brianwfoo | 2007-11-19 19:40:56 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/Datum.java

Removed file/folder
------------------------------------------------------------------------
r136 | brianwfoo | 2007-11-19 19:31:55 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
```

```
   A /trunk/src/TryDatum.java

Added a file remotely
------------------------------------------------------------------------
r135 | brianwfoo | 2007-11-19 19:31:32 -0500 (Mon, 19 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/TryDatum.java

Removed file/folder
------------------------------------------------------------------------
r134 | the.stuart.sierra | 2007-11-18 21:36:59 -0500 (Sun, 18 Nov 2007) | 1 line
Changed paths:
   M /wiki/ClassList.wiki

Added ExprList.
------------------------------------------------------------------------
r133 | the.stuart.sierra | 2007-11-18 21:14:14 -0500 (Sun, 18 Nov 2007) | 3 line
s
Changed paths:
   M /trunk/Makefile
   A /trunk/src/physicalc/Rel.java

* Added Rel class for relational operator nodes.
* Updated Makefile with new class.

------------------------------------------------------------------------
r132 | the.stuart.sierra | 2007-11-18 20:38:32 -0500 (Sun, 18 Nov 2007) | 2 line
s
Changed paths:
   M /trunk/src/physicalc/And.java

Fixed And.java: should not be an abstract class.

------------------------------------------------------------------------
r131 | the.stuart.sierra | 2007-11-18 20:18:41 -0500 (Sun, 18 Nov 2007) | 2 line
s
Changed paths:
   M /trunk/src/grammar.g

Parser fix: moved "not" to higher precedence.

------------------------------------------------------------------------
r130 | the.stuart.sierra | 2007-11-17 14:10:02 -0500 (Sat, 17 Nov 2007) | 3 line
s
Changed paths:
   M /trunk/Makefile
   A /trunk/src/physicalc/And.java
   A /trunk/src/physicalc/Logical.java
   A /trunk/src/physicalc/Op.java

Added Logical and Op base classes, and "And" example class.
Updated Makefile.

------------------------------------------------------------------------
r129 | the.stuart.sierra | 2007-11-16 09:56:31 -0500 (Fri, 16 Nov 2007) | 3 line
s
Changed paths:
   M /trunk/Makefile
   A /trunk/src/physicalc/Def.java
   A /trunk/src/physicalc/Expr.java
   M /trunk/src/physicalc/InterpreterError.java
   A /trunk/src/physicalc/Node.java
```



```
   A /trunk/src/physicalc/Stmt.java

* Wrote base Node classes.
* Fixed some small Makefile bugs.

------------------------------------------------------------------------
r128 | brianwfoo | 2007-11-15 16:19:09 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   A /trunk/Makefile

Added a file remotely
------------------------------------------------------------------------
r127 | brianwfoo | 2007-11-15 16:18:49 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   D /trunk/Makefile

Removed file/folder
------------------------------------------------------------------------
r126 | brianwfoo | 2007-11-15 16:14:02 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/TypeError.java

Added a file remotely
------------------------------------------------------------------------
r125 | brianwfoo | 2007-11-15 16:13:13 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/TypeError.java

Removed file/folder
------------------------------------------------------------------------
r124 | brianwfoo | 2007-11-15 16:13:00 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/Datum.java

Added a file remotely
------------------------------------------------------------------------
r123 | brianwfoo | 2007-11-15 16:12:31 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/Datum.java

Removed file/folder
------------------------------------------------------------------------
r122 | brianwfoo | 2007-11-15 16:12:09 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PVector.java

Added a file remotely
------------------------------------------------------------------------
r121 | brianwfoo | 2007-11-15 16:11:09 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnitPair.java

Added a file remotely
------------------------------------------------------------------------
r120 | brianwfoo | 2007-11-15 16:10:48 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PString.java

Added a file remotely
------------------------------------------------------------------------
r119 | brianwfoo | 2007-11-15 16:10:26 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
```

```
   A /trunk/src/physicalc/PList.java

Added a file remotely
------------------------------------------------------------------------
r118 | brianwfoo | 2007-11-15 16:10:00 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PBoolean.java

Added a file remotely
------------------------------------------------------------------------
r117 | brianwfoo | 2007-11-15 16:09:42 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   D /trunk/src/physicalc/PBoolean.java

Removed file/folder
------------------------------------------------------------------------
r116 | brianwfoo | 2007-11-15 16:08:46 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PUnit.java

Added a file remotely
------------------------------------------------------------------------
r115 | brianwfoo | 2007-11-15 16:08:05 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/physicalc/PNumber.java

Added a file remotely
------------------------------------------------------------------------
r114 | brianwfoo | 2007-11-15 16:07:18 -0500 (Thu, 15 Nov 2007) | 1 line
Changed paths:
   A /trunk/src/TryDatum.java

Added a file remotely
------------------------------------------------------------------------
r113 | the.stuart.sierra | 2007-11-11 10:59:53 -0500 (Sun, 11 Nov 2007) | 1 line
Changed paths:
   M /wiki/ClassList.wiki

Rewrote with program node tree from meeting 11/9/2007.
------------------------------------------------------------------------
r112 | ssierr@law.columbia.edu | 2007-11-10 15:07:25 -0500 (Sat, 10 Nov 2007) |
2 lines
Changed paths:
   A /trunk/src/physicalc/RuntimeObject.java
   A /trunk/src/physicalc/SymbolTable.java
   A /trunk/src/physicalc/UndefinedError.java

Added SymbolTable and associated classes.

------------------------------------------------------------------------
r111 | ssierr@law.columbia.edu | 2007-11-09 16:06:43 -0500 (Fri, 09 Nov 2007) |
2 lines
Changed paths:
   A /trunk/test/UnitTest.java

* UnitTest.java: added test file for Units

------------------------------------------------------------------------
r110 | ssierr@law.columbia.edu | 2007-11-07 12:53:49 -0500 (Wed, 07 Nov 2007) |
2 lines
Changed paths:
   D /trunk/ bcis --username ChadJiang
```



```

* Removed extraneous directory.

------------------------------------------------------------------------
r109 | ChadJiang | 2007-11-07 01:32:59 -0500 (Wed, 07 Nov 2007) | 1 line
Changed paths:
   A /trunk/ bcis --username ChadJiang


------------------------------------------------------------------------
r108 | ssierr@law.columbia.edu | 2007-11-06 23:19:51 -0500 (Tue, 06 Nov 2007) |
2 lines
Changed paths:
   M /trunk/profile.sh

* added -sourcepath to "compile" alias

------------------------------------------------------------------------
r107 | ssierr@law.columbia.edu | 2007-11-04 17:37:32 -0500 (Sun, 04 Nov 2007) |
4 lines
Changed paths:
   A /trunk/test/NumberTest.java

* Added NumberTest for testing integer & decimal arithmetic.
  This is NOT yet included in the Makefile, because the
  Number class is not in place.

------------------------------------------------------------------------
r106 | ssierr@law.columbia.edu | 2007-11-04 17:24:50 -0500 (Sun, 04 Nov 2007) |
5 lines
Changed paths:
   M /trunk/profile.sh

Added two new aliases:
1. "test" for running a single JUnit test class on the command line.
2. "compile" for compiling a single source file with the same
   options as would be used in the Makefile.

------------------------------------------------------------------------
r105 | ssierr@law.columbia.edu | 2007-11-03 13:38:55 -0400 (Sat, 03 Nov 2007) |
3 lines
Changed paths:
   M /trunk/Makefile
   M /trunk/src/grammar.g

* Added PhysiWalker (tree walker) to the end of grammar.g.
* Added Makefile rules to generate & compile the tree walker.

------------------------------------------------------------------------
r104 | ssierr@law.columbia.edu | 2007-11-02 19:29:13 -0400 (Fri, 02 Nov 2007) |
10 lines
Changed paths:
   M /trunk/Makefile
   M /trunk/profile.sh
   M /trunk/si.phy
   M /trunk/src/grammar.g
   A /trunk/src/physicalc/BoundsError.java
   A /trunk/src/physicalc/Datum.java
   A /trunk/src/physicalc/InterpreterError.java
   A /trunk/src/physicalc/PBoolean.java
   A /trunk/src/physicalc/TypeError.java
```

```
* Added abstract base class Datum.
* Added skeleton error classes InterpreterError, TypeError,
  and BoundsError.
* Removed Quantities from the grammar.
* Updated the Makefile for new classes; added -sourcepath so
  javac can find all the classes.
* Removed quantities from the library file si.phy.
* Added current directory to CLASSPATH in profile.sh for
  convenience.

------------------------------------------------------------------------
r103 | ssierr@law.columbia.edu | 2007-11-01 21:47:40 -0400 (Thu, 01 Nov 2007) |
2 lines
Changed paths:
   A /trunk/si.phy

* si.phy: added first draft of standard library

------------------------------------------------------------------------
r102 | ssierr@law.columbia.edu | 2007-11-01 21:47:20 -0400 (Thu, 01 Nov 2007) |
3 lines
Changed paths:
   M /trunk/Makefile
   A /trunk/src/ParseFile.java

* src/ParseFile.java: added new testing program
* Makefile: added rules for ParseFile

------------------------------------------------------------------------
r101 | ssierr@law.columbia.edu | 2007-11-01 21:45:58 -0400 (Thu, 01 Nov 2007) |
4 lines
Changed paths:
   M /trunk/src/grammar.g

grammar.g:
* added '!' to TERMINATOR in statements
* added rule to allow empty statements, to make comments work

------------------------------------------------------------------------
r100 | ssierr@law.columbia.edu | 2007-10-26 22:29:26 -0400 (Fri, 26 Oct 2007) |
2 lines
Changed paths:
   A /trunk/profile.sh

* added profile.sh, to set up CLASSPATH and other env vars.

------------------------------------------------------------------------
r99 | ssierr@law.columbia.edu | 2007-10-25 16:55:44 -0400 (Thu, 25 Oct 2007) | 2
 lines
Changed paths:
   M /trunk/src/grammar.g

* Added 'lvalue' for assigning to list elements.

------------------------------------------------------------------------
r98 | ssierr@law.columbia.edu | 2007-10-20 23:45:09 -0400 (Sat, 20 Oct 2007) | 6
 lines
Changed paths:
   M /trunk/Makefile
   A /trunk/src/physicalc/Interpreter.java
   A /trunk/test
   A /trunk/test/InterpreterTest.java
```



```
   A /trunk/test/PhysicalcSuite.java

* Added skeleton Interpreter class with I/O streams.
* Added test directory.
* Added InterpreterTest class with a "Hello, world!" test.
* Added PhysicalcSuite class to hold all test classes.
* Added 'test' target to Makefile which runs the suite.

------------------------------------------------------------------------
r97 | ssierr@law.columbia.edu | 2007-10-17 17:41:42 -0400 (Wed, 17 Oct 2007) | 2
 lines
Changed paths:
   M /trunk/src/grammar.g

* Complete grammar, including definitions and statements.

------------------------------------------------------------------------
r96 | ssierr@law.columbia.edu | 2007-10-17 13:40:56 -0400 (Wed, 17 Oct 2007) | 2
 lines
Changed paths:
   M /trunk/src/grammar.g

* Small corrections discovered while writing Reference Manual draft 4.

------------------------------------------------------------------------
r95 | ssierr@law.columbia.edu | 2007-10-17 00:13:32 -0400 (Wed, 17 Oct 2007) | 2
 lines
Changed paths:
   M /trunk/src/grammar.g

* Small corrections to grammar.

------------------------------------------------------------------------
r94 | ssierr@law.columbia.edu | 2007-10-16 22:42:36 -0400 (Tue, 16 Oct 2007) | 8
 lines
Changed paths:
   M /trunk/src/grammar.g

* Small corrections to expression grammar.
* Added function calls.
* Added not-equals expressions.
* Added AST node for unary minus.
* Added placeholder rules for definitions and statements.
* Renamed SEPARATOR token to TERMINATOR.
* Reordered rules.

------------------------------------------------------------------------
r93 | ssierr@law.columbia.edu | 2007-10-15 22:05:09 -0400 (Mon, 15 Oct 2007) | 4
 lines
Changed paths:
   M /trunk/src/TryParser.java
   M /trunk/src/grammar.g

* Wrote most of the grammar for expressions.
* Altered TryParser to use "program" as the starting rule
  when parsing.

------------------------------------------------------------------------
r92 | the.stuart.sierra | 2007-10-11 08:26:50 -0400 (Thu, 11 Oct 2007) | 1 line
Changed paths:
   A /wiki/BuiltinFunctions.wiki
```

```
Created page with 'print' and 'read'.
------------------------------------------------------------------------
r91 | the.stuart.sierra | 2007-10-11 08:22:57 -0400 (Thu, 11 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxReference.wiki

Added BuiltinFunctions.
------------------------------------------------------------------------
r90 | the.stuart.sierra | 2007-10-11 08:22:27 -0400 (Thu, 11 Oct 2007) | 1 line
Changed paths:
   M /wiki/ReservedWords.wiki

Edited wiki page through web user interface.
------------------------------------------------------------------------
r89 | the.stuart.sierra | 2007-10-11 08:21:57 -0400 (Thu, 11 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Changed to allow expressions as statements; removed "print".
------------------------------------------------------------------------
r88 | the.stuart.sierra | 2007-10-10 17:18:27 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   A /wiki/FundamentalTypes.wiki

Created page with basic first- and second-class types.
------------------------------------------------------------------------
r87 | the.stuart.sierra | 2007-10-10 17:13:11 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxReference.wiki

Added semantic section and FundamentalTypes link.
------------------------------------------------------------------------
r86 | the.stuart.sierra | 2007-10-10 17:12:03 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxLiterals.wiki

Added 'true' and 'false'.
------------------------------------------------------------------------
r85 | the.stuart.sierra | 2007-10-10 17:11:03 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/ReservedWords.wiki

Added 'true' and 'false'.
------------------------------------------------------------------------
r84 | the.stuart.sierra | 2007-10-10 17:05:20 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxExpressions.wiki

Added explanations about 'in' operator and roots.
------------------------------------------------------------------------
r83 | the.stuart.sierra | 2007-10-10 17:03:16 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxExpressions.wiki

Crossed out find...given operator.
------------------------------------------------------------------------
r82 | the.stuart.sierra | 2007-10-10 17:02:28 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxReference.wiki

Added link to ReservedWords.
------------------------------------------------------------------------
```



```
r81 | the.stuart.sierra | 2007-10-10 17:01:57 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxIdentifiers.wiki

Added link to ReservedWords.
------------------------------------------------------------------------
r80 | the.stuart.sierra | 2007-10-10 17:01:28 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   A /wiki/ReservedWords.wiki

Created initial list in alphabetical order.
------------------------------------------------------------------------
r79 | the.stuart.sierra | 2007-10-10 16:55:58 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Added sentence about expressions and statements.
------------------------------------------------------------------------
r78 | the.stuart.sierra | 2007-10-10 16:54:45 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Fixed wiki formatting of Assignment.
------------------------------------------------------------------------
r77 | the.stuart.sierra | 2007-10-10 16:54:18 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Clarified definition of for loop, reformatted "print".
------------------------------------------------------------------------
r76 | the.stuart.sierra | 2007-10-10 16:48:19 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Removed "read" (which can be a builtin function).
------------------------------------------------------------------------
r75 | the.stuart.sierra | 2007-10-10 16:47:11 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Minor edits & clarifications.
------------------------------------------------------------------------
r74 | the.stuart.sierra | 2007-10-10 16:46:30 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Minor edits.
------------------------------------------------------------------------
r73 | the.stuart.sierra | 2007-10-10 16:44:16 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Added sentence about creating local variables.
------------------------------------------------------------------------
r72 | the.stuart.sierra | 2007-10-10 16:43:11 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxDefinitions.wiki

Replaced "check" with "assert".
------------------------------------------------------------------------
r71 | the.stuart.sierra | 2007-10-10 16:41:53 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
```

```
   M /wiki/SyntaxDefinitions.wiki

Fixed wiki rendering of '*' operator.
------------------------------------------------------------------------
r70 | the.stuart.sierra | 2007-10-10 16:37:25 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxLiterals.wiki

Added Vectors and Lists.
------------------------------------------------------------------------
r69 | the.stuart.sierra | 2007-10-10 15:29:10 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/ClassList.wiki

Renamed "ClassList" link to "ClassForList" to avoid clash with this page.
------------------------------------------------------------------------
r68 | the.stuart.sierra | 2007-10-10 15:28:27 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/ClassList.wiki

Reordered and renamed some classes, added "ClassInterpreter"
------------------------------------------------------------------------
r67 | the.stuart.sierra | 2007-10-10 15:23:28 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxLiterals.wiki

Added sentence about units with numbers.
------------------------------------------------------------------------
r66 | the.stuart.sierra | 2007-10-10 15:21:53 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   D /wiki/Research.wiki

Deleting wiki page Research.
------------------------------------------------------------------------
r65 | the.stuart.sierra | 2007-10-10 15:21:41 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/UsefulLinks.wiki

Moved links from "Research" page.
------------------------------------------------------------------------
r64 | the.stuart.sierra | 2007-10-10 15:19:12 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/UsefulLinks.wiki

Removed "Inflector" link.
------------------------------------------------------------------------
r63 | the.stuart.sierra | 2007-10-10 15:18:35 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/Research.wiki

Removed all but "Physics" and "Computer Algebra"
------------------------------------------------------------------------
r62 | the.stuart.sierra | 2007-10-10 14:11:58 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   A /wiki/SyntaxLiterals.wiki

Created page with strings and numbers.
------------------------------------------------------------------------
r61 | the.stuart.sierra | 2007-10-10 13:58:44 -0400 (Wed, 10 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxReference.wiki
```



```
Added link to SyntaxLiterals.
------------------------------------------------------------------------
r60 | the.stuart.sierra | 2007-10-09 18:03:05 -0400 (Tue, 09 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxDefinitions.wiki

Added Constants and Functions.
------------------------------------------------------------------------
r59 | the.stuart.sierra | 2007-10-09 17:56:36 -0400 (Tue, 09 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Removed '$' from variable names.
------------------------------------------------------------------------
r58 | the.stuart.sierra | 2007-10-07 17:30:23 -0400 (Sun, 07 Oct 2007) | 1 line
Changed paths:
   A /wiki/SyntaxDefinitions.wiki

Created page with quantity and unit definitions.
------------------------------------------------------------------------
r57 | ssierr@law.columbia.edu | 2007-10-06 19:09:27 -0400 (Sat, 06 Oct 2007) | 4
 lines
Changed paths:
   M /trunk/Makefile
   A /trunk/src/TryLexer.java
   A /trunk/src/TryParser.java
   M /trunk/src/grammar.g

1. Started Lexer rules with whitespace, comments, and numbers.
2. Added TryParser and TryLexer classes for testing parser/lexer.
3. Added 'clean' and 'doc' (for JavaDoc) rules to the Makefile.

------------------------------------------------------------------------
r56 | the.stuart.sierra | 2007-10-04 16:25:15 -0400 (Thu, 04 Oct 2007) | 1 line
Changed paths:
   M /wiki/ClassList.wiki

Removed 'Featured' tag.
------------------------------------------------------------------------
r55 | the.stuart.sierra | 2007-10-04 16:24:06 -0400 (Thu, 04 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Added sentence about line break / semicolon.
------------------------------------------------------------------------
r54 | the.stuart.sierra | 2007-10-04 10:42:39 -0400 (Thu, 04 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxExpressions.wiki

Added note about removing Inferred Calculation.
------------------------------------------------------------------------
r53 | the.stuart.sierra | 2007-10-04 10:41:43 -0400 (Thu, 04 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Replaced if/do with more conventional if/then.
------------------------------------------------------------------------
r52 | the.stuart.sierra | 2007-10-04 10:41:09 -0400 (Thu, 04 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxIdentifiers.wiki

Replaced global/local separation with single C-like description.
```

```
------------------------------------------------------------------------
r51 | the.stuart.sierra | 2007-10-03 12:26:12 -0400 (Wed, 03 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxIdentifiers.wiki

Rewrote as local and global identifiers.
------------------------------------------------------------------------
r50 | the.stuart.sierra | 2007-10-03 12:21:43 -0400 (Wed, 03 Oct 2007) | 1 line
Changed paths:
   M /wiki/SyntaxExpressions.wiki

Rewrote page with operator list.
------------------------------------------------------------------------
r49 | ssierr@law.columbia.edu | 2007-10-01 12:07:26 -0400 (Mon, 01 Oct 2007) | 3
 lines
Changed paths:
   A /trunk/Makefile

Added beginning Makefile with rules for ANTLR and compilation.
This Makefile requires adding a new rule for each new class.

------------------------------------------------------------------------
r48 | ssierr@law.columbia.edu | 2007-10-01 12:06:49 -0400 (Mon, 01 Oct 2007) | 3
 lines
Changed paths:
   A /trunk/src/physicalc
   A /trunk/src/physicalc/Main.java

Added directory for .java source files in the 'physicalc' package
and a skeleton Main.java class.

------------------------------------------------------------------------
r47 | ssierr@law.columbia.edu | 2007-10-01 12:05:47 -0400 (Mon, 01 Oct 2007) | 2
 lines
Changed paths:
   A /trunk/src/grammar.g

Added skeleton grammar file for ANTLR.

------------------------------------------------------------------------
r46 | ssierr@law.columbia.edu | 2007-10-01 12:05:02 -0400 (Mon, 01 Oct 2007) | 2
 lines
Changed paths:
   M /trunk/class/physicalc

Set svn:ignore for *.class files in class/physicalc.

------------------------------------------------------------------------
r45 | ssierr@law.columbia.edu | 2007-10-01 12:01:56 -0400 (Mon, 01 Oct 2007) | 2
 lines
Changed paths:
   A /trunk/lib/junit-4.4.jar

Added lib/junit-4.4.jar, needed to run JUnit tests.

------------------------------------------------------------------------
r44 | ssierr@law.columbia.edu | 2007-10-01 12:01:18 -0400 (Mon, 01 Oct 2007) | 2
 lines
Changed paths:
   A /trunk/class
   A /trunk/class/physicalc
```



```
Added class/physicalc dir for compiled java .class files.

------------------------------------------------------------------------
r43 | ssierr@law.columbia.edu | 2007-10-01 11:39:13 -0400 (Mon, 01 Oct 2007) | 1
 line
Changed paths:
   M /wiki/SyntaxStatements.wiki

Added assignment.
------------------------------------------------------------------------
r42 | ssierr@law.columbia.edu | 2007-10-01 11:37:17 -0400 (Mon, 01 Oct 2007) | 1
 line
Changed paths:
   A /wiki/SyntaxExpressions.wiki

Created page with basic list of expressions.
------------------------------------------------------------------------
r41 | ssierr@law.columbia.edu | 2007-10-01 11:33:09 -0400 (Mon, 01 Oct 2007) | 1
 line
Changed paths:
   A /wiki/SyntaxStatements.wiki

Created basic I/O, if/then/else, for/while loops.
------------------------------------------------------------------------
r40 | ssierr@law.columbia.edu | 2007-10-01 10:48:08 -0400 (Mon, 01 Oct 2007) | 1
 line
Changed paths:
   A /wiki/ExampleSunMass.wiki

Created incomplete code example.
------------------------------------------------------------------------
r39 | ssierr@law.columbia.edu | 2007-10-01 10:44:24 -0400 (Mon, 01 Oct 2007) | 1
 line
Changed paths:
   A /wiki/ExamplePrograms.wiki

Created page with link to first example.
------------------------------------------------------------------------
r38 | ssierr@law.columbia.edu | 2007-10-01 10:36:47 -0400 (Mon, 01 Oct 2007) | 1
 line
Changed paths:
   M /wiki/CodingStyle.wiki

Added ANTLR code style.
------------------------------------------------------------------------
r37 | ssierr@law.columbia.edu | 2007-09-29 19:36:12 -0400 (Sat, 29 Sep 2007) | 2
 lines
Changed paths:
   A /trunk/lib/antlr.jar

lib/antlr.jar: Added ANTLR .jar file version 2.7.7

------------------------------------------------------------------------
r36 | ssierr@law.columbia.edu | 2007-09-29 19:32:00 -0400 (Sat, 29 Sep 2007) | 2
 lines
Changed paths:
   A /trunk/doc
   A /trunk/lib
   A /trunk/src

Added skeleton src/lib/doc dirs.
```

```
------------------------------------------------------------------------
r35 | ssierr@law.columbia.edu | 2007-09-29 09:49:48 -0400 (Sat, 29 Sep 2007) | 1
 line
Changed paths:
   A /wiki/CodingStyle.wiki

Created wiki page with summary of Sun coding standards.
------------------------------------------------------------------------
r34 | ssierr@law.columbia.edu | 2007-09-26 21:25:56 -0400 (Wed, 26 Sep 2007) | 1
 line
Changed paths:
   A /wiki/SyntaxIdentifiers.wiki

Created page based on meeting of 9/26/2007.
------------------------------------------------------------------------
r33 | ssierr@law.columbia.edu | 2007-09-26 21:19:38 -0400 (Wed, 26 Sep 2007) | 1
 line
Changed paths:
   M /wiki/SyntaxReference.wiki

Removed original content, replace with list of links to sub-pages.
------------------------------------------------------------------------
r32 | ssierr@law.columbia.edu | 2007-09-26 17:24:09 -0400 (Wed, 26 Sep 2007) | 1
 line
Changed paths:
   M /wiki/SyntaxReference.wiki

Added 'Featured' tag.
------------------------------------------------------------------------
r31 | ssierr@law.columbia.edu | 2007-09-26 17:20:13 -0400 (Wed, 26 Sep 2007) | 1
 line
Changed paths:
   A /wiki/SyntaxReference.wiki

Created early draft page.
------------------------------------------------------------------------
r30 | ssierr@law.columbia.edu | 2007-09-26 17:11:55 -0400 (Wed, 26 Sep 2007) | 1
 line
Changed paths:
   A /wiki/ClassList.wiki

Created initial class list.
------------------------------------------------------------------------
r29 | ssierr@law.columbia.edu | 2007-09-26 16:24:35 -0400 (Wed, 26 Sep 2007) | 1
 line
Changed paths:
   M /wiki/Proposal.wiki

Deleted draft Proposal page.
------------------------------------------------------------------------
r28 | ssierr@law.columbia.edu | 2007-09-21 16:57:30 -0400 (Fri, 21 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Proposal.wiki

Added paragraph about JScience.

------------------------------------------------------------------------
r27 | ssierr@law.columbia.edu | 2007-09-21 09:42:38 -0400 (Fri, 21 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/UsefulLinks.wiki
```



```
Added eclipse/ANTLR link

------------------------------------------------------------------------
r26 | ssierr@law.columbia.edu | 2007-09-20 17:17:47 -0400 (Thu, 20 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Research.wiki

Added link to JAS.

------------------------------------------------------------------------
r25 | ssierr@law.columbia.edu | 2007-09-20 17:08:28 -0400 (Thu, 20 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Research.wiki

Added link to JScience.

------------------------------------------------------------------------
r24 | ssierr@law.columbia.edu | 2007-09-20 16:30:49 -0400 (Thu, 20 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Proposal.wiki

Fixed some wiki formatting.

------------------------------------------------------------------------
r23 | ssierr@law.columbia.edu | 2007-09-20 16:28:53 -0400 (Thu, 20 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Proposal.wiki

Said it's 'much much simpler' than CAS systems.

------------------------------------------------------------------------
r22 | ssierr@law.columbia.edu | 2007-09-20 14:11:22 -0400 (Thu, 20 Sep 2007) | 1
 line
Changed paths:
   M /wiki/Research.wiki

Added link to unit calculation in CAS page.
------------------------------------------------------------------------
r21 | ssierr@law.columbia.edu | 2007-09-20 14:08:25 -0400 (Thu, 20 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Research.wiki

Added sympy link

------------------------------------------------------------------------
r20 | ssierr@law.columbia.edu | 2007-09-20 13:53:06 -0400 (Thu, 20 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Proposal.wiki

Proposal rewritten with more features, no examples.

------------------------------------------------------------------------
r19 | ssierr@law.columbia.edu | 2007-09-20 13:22:11 -0400 (Thu, 20 Sep 2007) | 1
 line
Changed paths:
```

```
   M /wiki/Research.wiki

added apfloat link
------------------------------------------------------------------------
r18 | ssierr@law.columbia.edu | 2007-09-19 17:46:48 -0400 (Wed, 19 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Research.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r17 | ssierr@law.columbia.edu | 2007-09-19 16:41:53 -0400 (Wed, 19 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Research.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r16 | ssierr@law.columbia.edu | 2007-09-19 16:40:57 -0400 (Wed, 19 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/UsefulLinks.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r15 | ssierr@law.columbia.edu | 2007-09-19 15:02:42 -0400 (Wed, 19 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/UsefulLinks.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r14 | ssierr@law.columbia.edu | 2007-09-18 11:22:46 -0400 (Tue, 18 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Proposal.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r13 | ssierr@law.columbia.edu | 2007-09-18 11:14:27 -0400 (Tue, 18 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Proposal.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r12 | ssierr@law.columbia.edu | 2007-09-18 11:09:09 -0400 (Tue, 18 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Proposal.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r11 | ssierr@law.columbia.edu | 2007-09-18 11:06:06 -0400 (Tue, 18 Sep 2007) | 2
 lines
```



```
Changed paths:
   M /wiki/Proposal.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r10 | ssierr@law.columbia.edu | 2007-09-18 11:00:46 -0400 (Tue, 18 Sep 2007) | 2
 lines
Changed paths:
   M /wiki/Proposal.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r9 | ssierr@law.columbia.edu | 2007-09-18 10:47:51 -0400 (Tue, 18 Sep 2007) | 3
lines
Changed paths:
   M /wiki/Proposal.wiki

  Edited wiki page through web user interface.


------------------------------------------------------------------------
r8 | ssierr@law.columbia.edu | 2007-09-18 10:32:19 -0400 (Tue, 18 Sep 2007) | 2
lines
Changed paths:
   M /wiki/Research.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r7 | ssierr@law.columbia.edu | 2007-09-18 10:06:46 -0400 (Tue, 18 Sep 2007) | 2
lines
Changed paths:
   M /wiki/Research.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r6 | ssierr@law.columbia.edu | 2007-09-18 09:52:27 -0400 (Tue, 18 Sep 2007) | 2
lines
Changed paths:
   A /wiki/Research.wiki

 Created wiki page through web user interface.

------------------------------------------------------------------------
r5 | ssierr@law.columbia.edu | 2007-09-18 09:51:11 -0400 (Tue, 18 Sep 2007) | 2
lines
Changed paths:
   M /wiki/UsefulLinks.wiki

 Edited wiki page through web user interface.

------------------------------------------------------------------------
r4 | ssierr@law.columbia.edu | 2007-09-18 09:50:50 -0400 (Tue, 18 Sep 2007) | 2
lines
Changed paths:
   M /wiki/UsefulLinks.wiki

 Edited wiki page through web user interface.
```

```
------------------------------------------------------------------------
r3 | ssierr@law.columbia.edu | 2007-09-18 09:48:06 -0400 (Tue, 18 Sep 2007) | 2
lines
Changed paths:
   A /wiki/UsefulLinks.wiki

 Created wiki page through web user interface.

------------------------------------------------------------------------
r2 | ssierr@law.columbia.edu | 2007-09-18 09:34:31 -0400 (Tue, 18 Sep 2007) | 5
lines
Changed paths:
   A /wiki
   A /wiki/Proposal.wiki

    Created wiki page through web user interface.




------------------------------------------------------------------------
r1 | (no author) | 2007-09-18 08:48:00 -0400 (Tue, 18 Sep 2007) | 1 line
Changed paths:
   A /branches
   A /tags
   A /trunk

Initial directory structure.
------------------------------------------------------------------------
```



```
# Makefile for Physicalc
# By Stuart Sierra, ss2806@columbia.edu

# Path to the Java runtime interpreter
JAVA = java

# Path to the Java compiler
JAVAC = javac

# Path to the JavaDoc tool
JAVADOC = javadoc

# The absolute path where this Makefile is located.
# Default is current working directory.
PROJECT = $(PWD)

# Directory where source .java file go.
SOURCE = $(PROJECT)/src

# Directory where compiled .class files will go.
CLASS = $(PROJECT)/class

# Directory for JavaDoc-generated documentation.
APIDOC = $(PROJECT)/doc/api

# Directory for .java source files for unit tests.
TEST = $(PROJECT)/test

# Directory for the .tex files for the final report.
REPORT = $(PROJECT)/report

# Paths and .jar files to search for Java .class files
# (colon separated)
CLASSPATH = $(CLASS):$(PROJECT)/lib/antlr.jar:$(PROJECT)/lib/junit-4.4.jar

# Flags for the Java compiler (-g includes debugging info)
JAVACFLAGS = -g -cp $(CLASSPATH) -sourcepath $(SOURCE) -d $(CLASS)

# Flags for the Java interpreter
JFLAGS = -cp $(CLASSPATH)

# Command line to compile a java class file:
JC = $(JAVAC) $(JAVACFLAGS)

# Directories for sources and class files within the 'physicalc'
# package.
SDIR = $(SOURCE)/physicalc
CDIR = $(CLASS)/physicalc

# Command line for ANTLR
ANTLR = $(JAVA) $(JFLAGS) antlr.Tool

# List of all project class files.  New classes should be added here
# and in the PER-CLASS COMPILATION RULES, below.
CLASSES = \
    $(CDIR)/PhysiLexer.class \
    $(CDIR)/PhysiLexerTokenTypes.class \
    $(CDIR)/PhysiParser.class \
    $(CDIR)/PhysiWalker.class \
    $(CLASS)/TryDatum.class \
    $(CLASS)/TryLexer.class \
    $(CLASS)/TryParser.class \
```

```
    $(CLASS)/ParseFile.class \
    $(CDIR)/Access.class \
    $(CDIR)/And.class \
    $(CDIR)/Arith.class \
    $(CDIR)/Block.class \
    $(CDIR)/BoundsError.class \
    $(CDIR)/Break.class \
    $(CDIR)/BreakSignal.class \
    $(CDIR)/ControlSignal.class \
    $(CDIR)/Datum.class \
    $(CDIR)/Def.class \
    $(CDIR)/Expr.class \
    $(CDIR)/ExprList.class \
    $(CDIR)/ExitFunction.class \
    $(CDIR)/For.class \
    $(CDIR)/FunCall.class \
    $(CDIR)/Function.class \
    $(CDIR)/GetNumberFunction.class \
    $(CDIR)/GetUnitFunction.class \
    $(CDIR)/Id.class \
    $(CDIR)/If.class \
    $(CDIR)/In.class \
    $(CDIR)/InterpreterError.class \
    $(CDIR)/Interpreter.class \
    $(CDIR)/Literal.class \
    $(CDIR)/Load.class \
    $(CDIR)/Logical.class \
    $(CDIR)/LValue.class \
    $(CDIR)/Main.class \
    $(CDIR)/Next.class \
    $(CDIR)/NextSignal.class \
    $(CDIR)/Node.class \
    $(CDIR)/Not.class \
    $(CDIR)/Op.class \
    $(CDIR)/Or.class \
    $(CDIR)/ParamList.class \
    $(CDIR)/PrintFunction.class \
    $(CDIR)/PBoolean.class \
    $(CDIR)/PList.class \
    $(CDIR)/PNumber.class \
    $(CDIR)/Program.class \
    $(CDIR)/PString.class \
    $(CDIR)/PUnit.class \
    $(CDIR)/PUnitPair.class \
    $(CDIR)/Rel.class \
    $(CDIR)/Return.class \
    $(CDIR)/ReturnSignal.class \
    $(CDIR)/RuntimeObject.class \
    $(CDIR)/Set.class \
    $(CDIR)/Stmt.class \
    $(CDIR)/SymbolTable.class \
    $(CDIR)/ToIntFunction.class \
    $(CDIR)/ToStringFunction.class \
    $(CDIR)/NPrintFunction.class \
    $(CDIR)/TypeError.class \
    $(CDIR)/Unary.class \
    $(CDIR)/UndefinedError.class \
    $(CDIR)/Variable.class \
    $(CDIR)/While.class \
    $(CDIR)/Constant.class \
    $(CDIR)/ConstantDef.class \
    $(CDIR)/Unit.class \
```



```
    $(CDIR)/UnitDef.class \
    $(CDIR)/FunctionDef.class \
    $(CDIR)/AliasDef.class

# List of all test class files.  New tests should be added here and in
# the PER-CLASS COMPILATION RULES, below.
TESTCLASSES = $(CDIR)/InterpreterTest.class \
    $(CDIR)/PhysicalcSuite.class

# Default target: compile all project classes (not tests)
all: $(CLASSES)

# 'run' target: run the "Main" class
run: $(CLASSES)
        $(JAVA) $(JFLAGS) physicalc.Main

# 'test' target: compile and run all unit tests
test: $(CLASSES) $(TESTCLASSES)
        $(JAVA) $(JFLAGS) org.junit.runner.JUnitCore physicalc.PhysicalcSuite

# The files generated by running ANTLR on the grammar file.
ANTLR_OUTPUT = $(SDIR)/PhysiLexer.java \
    $(SDIR)/PhysiParser.java \
    $(SDIR)/PhysiLexerTokenTypes.java \
    $(SDIR)/PhysiLexer.smap \
    $(SDIR)/PhysiLexerTokenTypes.txt \
    $(SDIR)/PhysiParser.smap \
    $(SDIR)/PhysiWalker.java \
    $(SDIR)/PhysiWalker.smap

# 'doc' target: make the Javadocs
doc: $(CLASSES)
        mkdir -p $(APIDOC)
        $(JAVADOC) -sourcepath $(SOURCE) \
                -private -d $(APIDOC) physicalc

report: $(REPORT)/physicalc-report.pdf

$(REPORT)/physicalc-report.pdf: $(REPORT)/finalreport.pdf $(REPORT)/sources.pdf
        (cd $(REPORT); pdftk finalreport.pdf sources.pdf \
    cat output physicalc-report.pdf )

$(REPORT)/finalreport.pdf: $(REPORT)/finalreport.tex \
    $(REPORT)/bibliography.tex \
    $(REPORT)/functions.tex \
    $(REPORT)/intro.tex \
    $(REPORT)/refman.tex
        (cd $(REPORT); pdflatex finalreport; pdflatex finalreport; pdflatex fina
lreport)

$(REPORT)/sources.pdf: $(REPORT)/sources.ps
        ps2pdf $(REPORT)/sources.ps $(REPORT)/sources.pdf

$(REPORT)/sources.ps: Changelog
        a2ps -A fill -o $(REPORT)/sources.ps \
    Changelog \
    Makefile \
    profile.sh si.phy otherunits.phy \
    runexamples runexample \
    src/grammar.g \
    src/*.java \
    src/physicalc/*.java \
```

```
    test/*.java \
    test/examples/*

Changelog:
	svn log -v http://bcis.googlecode.com/svn/ > Changelog


# Rules for generating the lexer & parser sources from the
# ANTLR grammar.
$(ANTLR_OUTPUT): $(SOURCE)/grammar.g
	$(ANTLR) -o $(SDIR) $(SOURCE)/grammar.g

# 'clean' target: remove all generated files
clean:
	rm -f $(ANTLR_OUTPUT) $(CLASSES) $(TESTCLASSES)
	rm -rf $(APIDOC)
	rm -f $(TEST)/examples/*.actual
	rm -f $(REPORT)/*.toc $(REPORT)/*.aux $(REPORT)/*.log
	rm -f $(REPORT)/*.pdf $(REPORT)/*.ps
	find $(PROJECT) -name '*~' -exec rm '{}' \;


### PER-CLASS COMPILATION RULES

# Compilation rules for each class file.  We need one rule
# for every class file because the .java sources and the
# compiled .class files go in different directories.

$(CLASS)/TryParser.class: $(SOURCE)/TryParser.java
	$(JC) $(SOURCE)/TryParser.java

$(CLASS)/TryLexer.class: $(SOURCE)/TryLexer.java
	$(JC) $(SOURCE)/TryLexer.java

$(CLASS)/TryDatum.class: $(SOURCE)/TryDatum.java
	$(JC) $(SOURCE)/TryDatum.java

$(CLASS)/ParseFile.class: $(SOURCE)/ParseFile.java
	$(JC) $(SOURCE)/ParseFile.java



$(CDIR)/PhysiLexer.class: $(SDIR)/PhysiLexer.java
	$(JC) $(SDIR)/PhysiLexer.java

$(CDIR)/PhysiLexerTokenTypes.class: $(SDIR)/PhysiLexerTokenTypes.java
	$(JC) $(SDIR)/PhysiLexerTokenTypes.java

$(CDIR)/PhysiParser.class: $(SDIR)/PhysiParser.java
	$(JC) $(SDIR)/PhysiParser.java

$(CDIR)/PhysiWalker.class: $(SDIR)/PhysiWalker.java
	$(JC) $(SDIR)/PhysiWalker.java



$(CDIR)/Access.class: $(SDIR)/Access.java
	$(JC) $(SDIR)/Access.java

$(CDIR)/And.class: $(SDIR)/And.java
	$(JC) $(SDIR)/And.java
```



```
$(CDIR)/Arith.class: $(SDIR)/Arith.java
	$(JC) $(SDIR)/Arith.java

$(CDIR)/Block.class: $(SDIR)/Block.java
	$(JC) $(SDIR)/Block.java

$(CDIR)/BoundsError.class: $(SDIR)/BoundsError.java
	$(JC) $(SDIR)/BoundsError.java

$(CDIR)/Break.class: $(SDIR)/Break.java
	$(JC) $(SDIR)/Break.java

$(CDIR)/BreakSignal.class: $(SDIR)/BreakSignal.java
	$(JC) $(SDIR)/BreakSignal.java

$(CDIR)/ControlSignal.class: $(SDIR)/ControlSignal.java
	$(JC) $(SDIR)/ControlSignal.java

$(CDIR)/Datum.class: $(SDIR)/Datum.java
	$(JC) $(SDIR)/Datum.java

$(CDIR)/Def.class: $(SDIR)/Def.java
	$(JC) $(SDIR)/Def.java

$(CDIR)/ExitFunction.class: $(SDIR)/ExitFunction.java
	$(JC) $(SDIR)/ExitFunction.java

$(CDIR)/Expr.class: $(SDIR)/Expr.java
	$(JC) $(SDIR)/Expr.java

$(CDIR)/ExprList.class: $(SDIR)/ExprList.java
	$(JC) $(SDIR)/ExprList.java

$(CDIR)/For.class: $(SDIR)/For.java
	$(JC) $(SDIR)/For.java

$(CDIR)/FunCall.class: $(SDIR)/FunCall.java
	$(JC) $(SDIR)/FunCall.java

$(CDIR)/Function.class: $(SDIR)/Function.java
	$(JC) $(SDIR)/Function.java

$(CDIR)/GetNumberFunction.class: $(SDIR)/GetNumberFunction.java
	$(JC) $(SDIR)/GetNumberFunction.java

$(CDIR)/GetUnitFunction.class: $(SDIR)/GetUnitFunction.java
	$(JC) $(SDIR)/GetUnitFunction.java

$(CDIR)/Id.class: $(SDIR)/Id.java
	$(JC) $(SDIR)/Id.java

$(CDIR)/If.class: $(SDIR)/If.java
	$(JC) $(SDIR)/If.java

$(CDIR)/In.class: $(SDIR)/In.java
	$(JC) $(SDIR)/In.java

$(CDIR)/InterpreterError.class: $(SDIR)/InterpreterError.java
	$(JC) $(SDIR)/InterpreterError.java

$(CDIR)/Interpreter.class: $(SDIR)/Interpreter.java
	$(JC) $(SDIR)/Interpreter.java
```

```
$(CDIR)/Literal.class: $(SDIR)/Literal.java
	$(JC) $(SDIR)/Literal.java

$(CDIR)/Load.class: $(SDIR)/Load.java
	$(JC) $(SDIR)/Load.java

$(CDIR)/Logical.class: $(SDIR)/Logical.java
	$(JC) $(SDIR)/Logical.java

$(CDIR)/LValue.class: $(SDIR)/LValue.java
	$(JC) $(SDIR)/LValue.java

$(CDIR)/Main.class: $(SDIR)/Main.java
	$(JC) $(SDIR)/Main.java

$(CDIR)/Next.class: $(SDIR)/Next.java
	$(JC) $(SDIR)/Next.java

$(CDIR)/NextSignal.class: $(SDIR)/NextSignal.java
	$(JC) $(SDIR)/NextSignal.java

$(CDIR)/Node.class: $(SDIR)/Node.java
	$(JC) $(SDIR)/Node.java

$(CDIR)/Not.class: $(SDIR)/Not.java
	$(JC) $(SDIR)/Not.java

$(CDIR)/Op.class: $(SDIR)/Op.java
	$(JC) $(SDIR)/Op.java

$(CDIR)/Or.class: $(SDIR)/Or.java
	$(JC) $(SDIR)/Or.java

$(CDIR)/ParamList.class: $(SDIR)/ParamList.java
	$(JC) $(SDIR)/ParamList.java

$(CDIR)/PBoolean.class: $(SDIR)/PBoolean.java
	$(JC) $(SDIR)/PBoolean.java

$(CDIR)/PList.class: $(SDIR)/PList.java
	$(JC) $(SDIR)/PList.java

$(CDIR)/PNumber.class: $(SDIR)/PNumber.java
	$(JC) $(SDIR)/PNumber.java

$(CDIR)/PrintFunction.class: $(SDIR)/PrintFunction.java
	$(JC) $(SDIR)/PrintFunction.java

$(CDIR)/Program.class: $(SDIR)/Program.java
	$(JC) $(SDIR)/Program.java

$(CDIR)/PString.class: $(SDIR)/PString.java
	$(JC) $(SDIR)/PString.java

$(CDIR)/PUnit.class: $(SDIR)/PUnit.java
	$(JC) $(SDIR)/PUnit.java

$(CDIR)/PUnitPair.class: $(SDIR)/PUnitPair.java
	$(JC) $(SDIR)/PUnitPair.java

$(CDIR)/Rel.class: $(SDIR)/Rel.java
```



```
	$(JC) $(SDIR)/Rel.java

$(CDIR)/Return.class: $(SDIR)/Return.java
	$(JC) $(SDIR)/Return.java

$(CDIR)/ReturnSignal.class: $(SDIR)/ReturnSignal.java
	$(JC) $(SDIR)/ReturnSignal.java

$(CDIR)/RuntimeObject.class: $(SDIR)/RuntimeObject.java
	$(JC) $(SDIR)/RuntimeObject.java

$(CDIR)/Set.class: $(SDIR)/Set.java
	$(JC) $(SDIR)/Set.java

$(CDIR)/Stmt.class: $(SDIR)/Stmt.java
	$(JC) $(SDIR)/Stmt.java

$(CDIR)/SymbolTable.class: $(SDIR)/SymbolTable.java
	$(JC) $(SDIR)/SymbolTable.java

$(CDIR)/ToIntFunction.class: $(SDIR)/ToIntFunction.java
	$(JC) $(SDIR)/ToIntFunction.java

$(CDIR)/ToStringFunction.class: $(SDIR)/ToStringFunction.java
	$(JC) $(SDIR)/ToStringFunction.java

$(CDIR)/NPrintFunction.class: $(SDIR)/NPrintFunction.java
	$(JC) $(SDIR)/NPrintFunction.java

$(CDIR)/TypeError.class: $(SDIR)/TypeError.java
	$(JC) $(SDIR)/TypeError.java

$(CDIR)/UndefinedError.class: $(SDIR)/UndefinedError.java
	$(JC) $(SDIR)/UndefinedError.java

$(CDIR)/Unary.class: $(SDIR)/Unary.java
	$(JC) $(SDIR)/Unary.java

$(CDIR)/Variable.class: $(SDIR)/Variable.java
	$(JC) $(SDIR)/Variable.java

$(CDIR)/While.class: $(SDIR)/While.java
	$(JC) $(SDIR)/While.java

$(CDIR)/ConstantDef.class: $(SDIR)/ConstantDef.java
	$(JC) $(SDIR)/ConstantDef.java

$(CDIR)/Constant.class: $(SDIR)/Constant.java
	$(JC) $(SDIR)/Constant.java

$(CDIR)/Unit.class: $(SDIR)/Unit.java
	$(JC) $(SDIR)/Unit.java

$(CDIR)/UnitDef.class: $(SDIR)/UnitDef.java
	$(JC) $(SDIR)/UnitDef.java

$(CDIR)/FunctionDef.class: $(SDIR)/FunctionDef.java
	$(JC) $(SDIR)/FunctionDef.java

$(CDIR)/AliasDef.class: $(SDIR)/AliasDef.java
	$(JC) $(SDIR)/AliasDef.java
```

```
$(CDIR)/InterpreterTest.class: $(TEST)/InterpreterTest.java
	$(JC) $(TEST)/InterpreterTest.java

$(CDIR)/PhysicalcSuite.class: $(TEST)/PhysicalcSuite.java
	$(JC) $(TEST)/PhysicalcSuite.java
```



```
#!/bin/bash

# profile.sh

# This file sets up the CLASSPATH and other needed environment
# variables to run the example programs and tests.
#
# Do not execute this file as a shell script; instead, "source" it at
# the shell command line like this:
#
#     source profile.sh
#
# This will only work if your shell is "bash".  It should work in any
# UNIX-like environment, including Linux and Cygwin.


# Java class search path: needs to include the project "class"
# directory and any .jar files.
export CLASSPATH=.:$PWD:$PWD/class:$PWD/lib/antlr.jar:$PWD/lib/junit-4.4.jar

# Java source search path:
export SOURCEPATH=.:$PWD:$PWD/src:$PWD/test

# Convenience alias for ANTLR.
alias antlr="java antlr.Tool -diagnostic"

# Convenience alias for compiling files without changing the Makefile:
alias compile="javac -g -d $PWD/class -sourcepath $SOURCEPATH"

# Convenience alias for running a single test class.  Should be
# followed by the name of a Test class, like "physicalc.NumberTest":
alias test="java org.junit.runner.JUnitCore"
```

```
# SI Units
# by Stuart Sierra

# SI Base Quantities & Base Units
# from http://en.wikipedia.org/wiki/SI_base_unit
unit meter
unit kilogram
unit second
unit ampere
unit kelvin
unit mole
unit candela

# SI Derived Units
# from http://en.wikipedia.org/wiki/SI_derived_unit

unit minute = 60 * second
unit hour = 60 * minute
unit day = 24 * hour
unit year = 365 * day
unit newton = meter * kilogram / second ^ 2

unit hertz = 1 * second ^-1
unit newton = meter * kilogram / second ^ 2
unit pascal = newton / meter ^ 2
unit joule = newton * meter
unit watt = joule / second
unit coulomb = second * ampere
unit volt = watt / ampere
unit farad = coulomb / volt
unit ohm = volt / ampere
unit siemens = 1 * ohm ^ -1
unit weber = joule / ampere
unit tesla = volt * second / meter ^ 2
unit henry = volt * second / ampere
unit lumen = candela
unit lux = lumen / meter ^ 2
unit becquerel = 1 *  second^-1
unit gray = joule / kilogram
unit sievert = joule / kilogram
unit katal = mole / second


unit gram = kilogram * 0.001

# Prefixed SI units
unit yottahertz = hertz * 10 ^ 24
unit zettahertz = hertz * 10 ^ 21
unit exahertz = hertz * 10 ^ 18
unit petahertz = hertz * 10 ^ 15
unit terahertz = hertz * 10 ^ 12
unit gigahertz = hertz * 10 ^ 9
unit megahertz = hertz * 10 ^ 6
unit kilohertz = hertz * 10 ^ 3
unit hectohertz = hertz * 10 ^ 2
unit decahertz = hertz * 10 ^ 1
unit decihertz = hertz * 10 ^ -1
unit centihertz = hertz * 10 ^ -2
unit millihertz = hertz * 10 ^ -3
unit microhertz = hertz * 10 ^ -6
unit nanohertz = hertz * 10 ^ -9
unit picohertz = hertz * 10 ^ -12
```



```
unit femtohertz = hertz * 10 ^ -15
unit attohertz = hertz * 10 ^ -18
unit zeptohertz = hertz * 10 ^ -21
unit yoctohertz = hertz * 10 ^ -24
unit yottanewton = newton * 10 ^ 24
unit zettanewton = newton * 10 ^ 21
unit exanewton = newton * 10 ^ 18
unit petanewton = newton * 10 ^ 15
unit teranewton = newton * 10 ^ 12
unit giganewton = newton * 10 ^ 9
unit meganewton = newton * 10 ^ 6
unit kilonewton = newton * 10 ^ 3
unit hectonewton = newton * 10 ^ 2
unit decanewton = newton * 10 ^ 1
unit decinewton = newton * 10 ^ -1
unit centinewton = newton * 10 ^ -2
unit millinewton = newton * 10 ^ -3
unit micronewton = newton * 10 ^ -6
unit nanonewton = newton * 10 ^ -9
unit piconewton = newton * 10 ^ -12
unit femtonewton = newton * 10 ^ -15
unit attonewton = newton * 10 ^ -18
unit zeptonewton = newton * 10 ^ -21
unit yoctonewton = newton * 10 ^ -24
unit yottapascal = pascal * 10 ^ 24
unit zettapascal = pascal * 10 ^ 21
unit exapascal = pascal * 10 ^ 18
unit petapascal = pascal * 10 ^ 15
unit terapascal = pascal * 10 ^ 12
unit gigapascal = pascal * 10 ^ 9
unit megapascal = pascal * 10 ^ 6
unit kilopascal = pascal * 10 ^ 3
unit hectopascal = pascal * 10 ^ 2
unit decapascal = pascal * 10 ^ 1
unit decipascal = pascal * 10 ^ -1
unit centipascal = pascal * 10 ^ -2
unit millipascal = pascal * 10 ^ -3
unit micropascal = pascal * 10 ^ -6
unit nanopascal = pascal * 10 ^ -9
unit picopascal = pascal * 10 ^ -12
unit femtopascal = pascal * 10 ^ -15
unit attopascal = pascal * 10 ^ -18
unit zeptopascal = pascal * 10 ^ -21
unit yoctopascal = pascal * 10 ^ -24
unit yottajoule = joule * 10 ^ 24
unit zettajoule = joule * 10 ^ 21
unit exajoule = joule * 10 ^ 18
unit petajoule = joule * 10 ^ 15
unit terajoule = joule * 10 ^ 12
unit gigajoule = joule * 10 ^ 9
unit megajoule = joule * 10 ^ 6
unit kilojoule = joule * 10 ^ 3
unit hectojoule = joule * 10 ^ 2
unit decajoule = joule * 10 ^ 1
unit decijoule = joule * 10 ^ -1
unit centijoule = joule * 10 ^ -2
unit millijoule = joule * 10 ^ -3
unit microjoule = joule * 10 ^ -6
unit nanojoule = joule * 10 ^ -9
unit picojoule = joule * 10 ^ -12
unit femtojoule = joule * 10 ^ -15
unit attojoule = joule * 10 ^ -18
```

```
unit zeptojoule = joule * 10 ^ -21
unit yoctojoule = joule * 10 ^ -24
unit yottawatt = watt * 10 ^ 24
unit zettawatt = watt * 10 ^ 21
unit exawatt = watt * 10 ^ 18
unit petawatt = watt * 10 ^ 15
unit terawatt = watt * 10 ^ 12
unit gigawatt = watt * 10 ^ 9
unit megawatt = watt * 10 ^ 6
unit kilowatt = watt * 10 ^ 3
unit hectowatt = watt * 10 ^ 2
unit decawatt = watt * 10 ^ 1
unit deciwatt = watt * 10 ^ -1
unit centiwatt = watt * 10 ^ -2
unit milliwatt = watt * 10 ^ -3
unit microwatt = watt * 10 ^ -6
unit nanowatt = watt * 10 ^ -9
unit picowatt = watt * 10 ^ -12
unit femtowatt = watt * 10 ^ -15
unit attowatt = watt * 10 ^ -18
unit zeptowatt = watt * 10 ^ -21
unit yoctowatt = watt * 10 ^ -24
unit yottacoulomb = coulomb * 10 ^ 24
unit zettacoulomb = coulomb * 10 ^ 21
unit exacoulomb = coulomb * 10 ^ 18
unit petacoulomb = coulomb * 10 ^ 15
unit teracoulomb = coulomb * 10 ^ 12
unit gigacoulomb = coulomb * 10 ^ 9
unit megacoulomb = coulomb * 10 ^ 6
unit kilocoulomb = coulomb * 10 ^ 3
unit hectocoulomb = coulomb * 10 ^ 2
unit decacoulomb = coulomb * 10 ^ 1
unit decicoulomb = coulomb * 10 ^ -1
unit centicoulomb = coulomb * 10 ^ -2
unit millicoulomb = coulomb * 10 ^ -3
unit microcoulomb = coulomb * 10 ^ -6
unit nanocoulomb = coulomb * 10 ^ -9
unit picocoulomb = coulomb * 10 ^ -12
unit femtocoulomb = coulomb * 10 ^ -15
unit attocoulomb = coulomb * 10 ^ -18
unit zeptocoulomb = coulomb * 10 ^ -21
unit yoctocoulomb = coulomb * 10 ^ -24
unit yottavolt = volt * 10 ^ 24
unit zettavolt = volt * 10 ^ 21
unit exavolt = volt * 10 ^ 18
unit petavolt = volt * 10 ^ 15
unit teravolt = volt * 10 ^ 12
unit gigavolt = volt * 10 ^ 9
unit megavolt = volt * 10 ^ 6
unit kilovolt = volt * 10 ^ 3
unit hectovolt = volt * 10 ^ 2
unit decavolt = volt * 10 ^ 1
unit decivolt = volt * 10 ^ -1
unit centivolt = volt * 10 ^ -2
unit millivolt = volt * 10 ^ -3
unit microvolt = volt * 10 ^ -6
unit nanovolt = volt * 10 ^ -9
unit picovolt = volt * 10 ^ -12
unit femtovolt = volt * 10 ^ -15
unit attovolt = volt * 10 ^ -18
unit zeptovolt = volt * 10 ^ -21
unit yoctovolt = volt * 10 ^ -24
```



```
unit yottafarad = farad * 10 ^ 24
unit zettafarad = farad * 10 ^ 21
unit exafarad = farad * 10 ^ 18
unit petafarad = farad * 10 ^ 15
unit terafarad = farad * 10 ^ 12
unit gigafarad = farad * 10 ^ 9
unit megafarad = farad * 10 ^ 6
unit kilofarad = farad * 10 ^ 3
unit hectofarad = farad * 10 ^ 2
unit decafarad = farad * 10 ^ 1
unit decifarad = farad * 10 ^ -1
unit centifarad = farad * 10 ^ -2
unit millifarad = farad * 10 ^ -3
unit microfarad = farad * 10 ^ -6
unit nanofarad = farad * 10 ^ -9
unit picofarad = farad * 10 ^ -12
unit femtofarad = farad * 10 ^ -15
unit attofarad = farad * 10 ^ -18
unit zeptofarad = farad * 10 ^ -21
unit yoctofarad = farad * 10 ^ -24
unit yottaohm = ohm * 10 ^ 24
unit zettaohm = ohm * 10 ^ 21
unit exaohm = ohm * 10 ^ 18
unit petaohm = ohm * 10 ^ 15
unit teraohm = ohm * 10 ^ 12
unit gigaohm = ohm * 10 ^ 9
unit megaohm = ohm * 10 ^ 6
unit kiloohm = ohm * 10 ^ 3
unit hectoohm = ohm * 10 ^ 2
unit decaohm = ohm * 10 ^ 1
unit deciohm = ohm * 10 ^ -1
unit centiohm = ohm * 10 ^ -2
unit milliohm = ohm * 10 ^ -3
unit microohm = ohm * 10 ^ -6
unit nanoohm = ohm * 10 ^ -9
unit picoohm = ohm * 10 ^ -12
unit femtoohm = ohm * 10 ^ -15
unit attoohm = ohm * 10 ^ -18
unit zeptoohm = ohm * 10 ^ -21
unit yoctoohm = ohm * 10 ^ -24
unit yottasiemens = siemens * 10 ^ 24
unit zettasiemens = siemens * 10 ^ 21
unit exasiemens = siemens * 10 ^ 18
unit petasiemens = siemens * 10 ^ 15
unit terasiemens = siemens * 10 ^ 12
unit gigasiemens = siemens * 10 ^ 9
unit megasiemens = siemens * 10 ^ 6
unit kilosiemens = siemens * 10 ^ 3
unit hectosiemens = siemens * 10 ^ 2
unit decasiemens = siemens * 10 ^ 1
unit decisiemens = siemens * 10 ^ -1
unit centisiemens = siemens * 10 ^ -2
unit millisiemens = siemens * 10 ^ -3
unit microsiemens = siemens * 10 ^ -6
unit nanosiemens = siemens * 10 ^ -9
unit picosiemens = siemens * 10 ^ -12
unit femtosiemens = siemens * 10 ^ -15
unit attosiemens = siemens * 10 ^ -18
unit zeptosiemens = siemens * 10 ^ -21
unit yoctosiemens = siemens * 10 ^ -24
unit yottaweber = weber * 10 ^ 24
unit zettaweber = weber * 10 ^ 21
```

```
unit exaweber = weber * 10 ^ 18
unit petaweber = weber * 10 ^ 15
unit teraweber = weber * 10 ^ 12
unit gigaweber = weber * 10 ^ 9
unit megaweber = weber * 10 ^ 6
unit kiloweber = weber * 10 ^ 3
unit hectoweber = weber * 10 ^ 2
unit decaweber = weber * 10 ^ 1
unit deciweber = weber * 10 ^ −1
unit centiweber = weber * 10 ^ −2
unit milliweber = weber * 10 ^ −3
unit microweber = weber * 10 ^ −6
unit nanoweber = weber * 10 ^ −9
unit picoweber = weber * 10 ^ −12
unit femtoweber = weber * 10 ^ −15
unit attoweber = weber * 10 ^ −18
unit zeptoweber = weber * 10 ^ −21
unit yoctoweber = weber * 10 ^ −24
unit yottatesla = tesla * 10 ^ 24
unit zettatesla = tesla * 10 ^ 21
unit exatesla = tesla * 10 ^ 18
unit petatesla = tesla * 10 ^ 15
unit teratesla = tesla * 10 ^ 12
unit gigatesla = tesla * 10 ^ 9
unit megatesla = tesla * 10 ^ 6
unit kilotesla = tesla * 10 ^ 3
unit hectotesla = tesla * 10 ^ 2
unit decatesla = tesla * 10 ^ 1
unit decitesla = tesla * 10 ^ −1
unit centitesla = tesla * 10 ^ −2
unit millitesla = tesla * 10 ^ −3
unit microtesla = tesla * 10 ^ −6
unit nanotesla = tesla * 10 ^ −9
unit picotesla = tesla * 10 ^ −12
unit femtotesla = tesla * 10 ^ −15
unit attotesla = tesla * 10 ^ −18
unit zeptotesla = tesla * 10 ^ −21
unit yoctotesla = tesla * 10 ^ −24
unit yottahenry = henry * 10 ^ 24
unit zettahenry = henry * 10 ^ 21
unit exahenry = henry * 10 ^ 18
unit petahenry = henry * 10 ^ 15
unit terahenry = henry * 10 ^ 12
unit gigahenry = henry * 10 ^ 9
unit megahenry = henry * 10 ^ 6
unit kilohenry = henry * 10 ^ 3
unit hectohenry = henry * 10 ^ 2
unit decahenry = henry * 10 ^ 1
unit decihenry = henry * 10 ^ −1
unit centihenry = henry * 10 ^ −2
unit millihenry = henry * 10 ^ −3
unit microhenry = henry * 10 ^ −6
unit nanohenry = henry * 10 ^ −9
unit picohenry = henry * 10 ^ −12
unit femtohenry = henry * 10 ^ −15
unit attohenry = henry * 10 ^ −18
unit zeptohenry = henry * 10 ^ −21
unit yoctohenry = henry * 10 ^ −24
unit yottalumen = lumen * 10 ^ 24
unit zettalumen = lumen * 10 ^ 21
unit exalumen = lumen * 10 ^ 18
unit petalumen = lumen * 10 ^ 15
```



```
unit teralumen = lumen * 10 ^ 12
unit gigalumen = lumen * 10 ^ 9
unit megalumen = lumen * 10 ^ 6
unit kilolumen = lumen * 10 ^ 3
unit hectolumen = lumen * 10 ^ 2
unit decalumen = lumen * 10 ^ 1
unit decilumen = lumen * 10 ^ −1
unit centilumen = lumen * 10 ^ −2
unit millilumen = lumen * 10 ^ −3
unit microlumen = lumen * 10 ^ −6
unit nanolumen = lumen * 10 ^ −9
unit picolumen = lumen * 10 ^ −12
unit femtolumen = lumen * 10 ^ −15
unit attolumen = lumen * 10 ^ −18
unit zeptolumen = lumen * 10 ^ −21
unit yoctolumen = lumen * 10 ^ −24
unit yottalux = lux * 10 ^ 24
unit zettalux = lux * 10 ^ 21
unit exalux = lux * 10 ^ 18
unit petalux = lux * 10 ^ 15
unit teralux = lux * 10 ^ 12
unit gigalux = lux * 10 ^ 9
unit megalux = lux * 10 ^ 6
unit kilolux = lux * 10 ^ 3
unit hectolux = lux * 10 ^ 2
unit decalux = lux * 10 ^ 1
unit decilux = lux * 10 ^ −1
unit centilux = lux * 10 ^ −2
unit millilux = lux * 10 ^ −3
unit microlux = lux * 10 ^ −6
unit nanolux = lux * 10 ^ −9
unit picolux = lux * 10 ^ −12
unit femtolux = lux * 10 ^ −15
unit attolux = lux * 10 ^ −18
unit zeptolux = lux * 10 ^ −21
unit yoctolux = lux * 10 ^ −24
unit yottabecquerel = becquerel * 10 ^ 24
unit zettabecquerel = becquerel * 10 ^ 21
unit exabecquerel = becquerel * 10 ^ 18
unit petabecquerel = becquerel * 10 ^ 15
unit terabecquerel = becquerel * 10 ^ 12
unit gigabecquerel = becquerel * 10 ^ 9
unit megabecquerel = becquerel * 10 ^ 6
unit kilobecquerel = becquerel * 10 ^ 3
unit hectobecquerel = becquerel * 10 ^ 2
unit decabecquerel = becquerel * 10 ^ 1
unit decibecquerel = becquerel * 10 ^ −1
unit centibecquerel = becquerel * 10 ^ −2
unit millibecquerel = becquerel * 10 ^ −3
unit microbecquerel = becquerel * 10 ^ −6
unit nanobecquerel = becquerel * 10 ^ −9
unit picobecquerel = becquerel * 10 ^ −12
unit femtobecquerel = becquerel * 10 ^ −15
unit attobecquerel = becquerel * 10 ^ −18
unit zeptobecquerel = becquerel * 10 ^ −21
unit yoctobecquerel = becquerel * 10 ^ −24
unit yottagray = gray * 10 ^ 24
unit zettagray = gray * 10 ^ 21
unit exagray = gray * 10 ^ 18
unit petagray = gray * 10 ^ 15
unit teragray = gray * 10 ^ 12
unit gigagray = gray * 10 ^ 9
```

```
unit megagray = gray * 10 ^ 6
unit kilogray = gray * 10 ^ 3
unit hectogray = gray * 10 ^ 2
unit decagray = gray * 10 ^ 1
unit decigray = gray * 10 ^ −1
unit centigray = gray * 10 ^ −2
unit milligray = gray * 10 ^ −3
unit microgray = gray * 10 ^ −6
unit nanogray = gray * 10 ^ −9
unit picogray = gray * 10 ^ −12
unit femtogray = gray * 10 ^ −15
unit attogray = gray * 10 ^ −18
unit zeptogray = gray * 10 ^ −21
unit yoctogray = gray * 10 ^ −24
unit yottasievert = sievert * 10 ^ 24
unit zettasievert = sievert * 10 ^ 21
unit exasievert = sievert * 10 ^ 18
unit petasievert = sievert * 10 ^ 15
unit terasievert = sievert * 10 ^ 12
unit gigasievert = sievert * 10 ^ 9
unit megasievert = sievert * 10 ^ 6
unit kilosievert = sievert * 10 ^ 3
unit hectosievert = sievert * 10 ^ 2
unit decasievert = sievert * 10 ^ 1
unit decisievert = sievert * 10 ^ −1
unit centisievert = sievert * 10 ^ −2
unit millisievert = sievert * 10 ^ −3
unit microsievert = sievert * 10 ^ −6
unit nanosievert = sievert * 10 ^ −9
unit picosievert = sievert * 10 ^ −12
unit femtosievert = sievert * 10 ^ −15
unit attosievert = sievert * 10 ^ −18
unit zeptosievert = sievert * 10 ^ −21
unit yoctosievert = sievert * 10 ^ −24
unit yottakatal = katal * 10 ^ 24
unit zettakatal = katal * 10 ^ 21
unit exakatal = katal * 10 ^ 18
unit petakatal = katal * 10 ^ 15
unit terakatal = katal * 10 ^ 12
unit gigakatal = katal * 10 ^ 9
unit megakatal = katal * 10 ^ 6
unit kilokatal = katal * 10 ^ 3
unit hectokatal = katal * 10 ^ 2
unit decakatal = katal * 10 ^ 1
unit decikatal = katal * 10 ^ −1
unit centikatal = katal * 10 ^ −2
unit millikatal = katal * 10 ^ −3
unit microkatal = katal * 10 ^ −6
unit nanokatal = katal * 10 ^ −9
unit picokatal = katal * 10 ^ −12
unit femtokatal = katal * 10 ^ −15
unit attokatal = katal * 10 ^ −18
unit zeptokatal = katal * 10 ^ −21
unit yoctokatal = katal * 10 ^ −24
unit yottagram = gram * 10 ^ 24
unit zettagram = gram * 10 ^ 21
unit exagram = gram * 10 ^ 18
unit petagram = gram * 10 ^ 15
unit teragram = gram * 10 ^ 12
unit gigagram = gram * 10 ^ 9
unit megagram = gram * 10 ^ 6
unit kilogram = gram * 10 ^ 3
```



```
unit hectogram = gram * 10 ^ 2
unit decagram = gram * 10 ^ 1
unit decigram = gram * 10 ^ −1
unit centigram = gram * 10 ^ −2
unit milligram = gram * 10 ^ −3
unit microgram = gram * 10 ^ −6
unit nanogram = gram * 10 ^ −9
unit picogram = gram * 10 ^ −12
unit femtogram = gram * 10 ^ −15
unit attogram = gram * 10 ^ −18
unit zeptogram = gram * 10 ^ −21
unit yoctogram = gram * 10 ^ −24
unit yottameter = meter * 10 ^ 24
unit zettameter = meter * 10 ^ 21
unit exameter = meter * 10 ^ 18
unit petameter = meter * 10 ^ 15
unit terameter = meter * 10 ^ 12
unit gigameter = meter * 10 ^ 9
unit megameter = meter * 10 ^ 6
unit kilometer = meter * 10 ^ 3
unit hectometer = meter * 10 ^ 2
unit decameter = meter * 10 ^ 1
unit decimeter = meter * 10 ^ −1
unit centimeter = meter * 10 ^ −2
unit millimeter = meter * 10 ^ −3
unit micrometer = meter * 10 ^ −6
unit nanometer = meter * 10 ^ −9
unit picometer = meter * 10 ^ −12
unit femtometer = meter * 10 ^ −15
unit attometer = meter * 10 ^ −18
unit zeptometer = meter * 10 ^ −21
unit yoctometer = meter * 10 ^ −24
unit yottasecond = second * 10 ^ 24
unit zettasecond = second * 10 ^ 21
unit exasecond = second * 10 ^ 18
unit petasecond = second * 10 ^ 15
unit terasecond = second * 10 ^ 12
unit gigasecond = second * 10 ^ 9
unit megasecond = second * 10 ^ 6
unit kilosecond = second * 10 ^ 3
unit hectosecond = second * 10 ^ 2
unit decasecond = second * 10 ^ 1
unit decisecond = second * 10 ^ −1
unit centisecond = second * 10 ^ −2
unit millisecond = second * 10 ^ −3
unit microsecond = second * 10 ^ −6
unit nanosecond = second * 10 ^ −9
unit picosecond = second * 10 ^ −12
unit femtosecond = second * 10 ^ −15
unit attosecond = second * 10 ^ −18
unit zeptosecond = second * 10 ^ −21
unit yoctosecond = second * 10 ^ −24
unit yottaampere = ampere * 10 ^ 24
unit zettaampere = ampere * 10 ^ 21
unit exaampere = ampere * 10 ^ 18
unit petaampere = ampere * 10 ^ 15
unit teraampere = ampere * 10 ^ 12
unit gigaampere = ampere * 10 ^ 9
unit megaampere = ampere * 10 ^ 6
unit kiloampere = ampere * 10 ^ 3
unit hectoampere = ampere * 10 ^ 2
unit decaampere = ampere * 10 ^ 1
```

```
unit deciampere = ampere * 10 ^ −1
unit centiampere = ampere * 10 ^ −2
unit milliampere = ampere * 10 ^ −3
unit microampere = ampere * 10 ^ −6
unit nanoampere = ampere * 10 ^ −9
unit picoampere = ampere * 10 ^ −12
unit femtoampere = ampere * 10 ^ −15
unit attoampere = ampere * 10 ^ −18
unit zeptoampere = ampere * 10 ^ −21
unit yoctoampere = ampere * 10 ^ −24
unit yottakelvin = kelvin * 10 ^ 24
unit zettakelvin = kelvin * 10 ^ 21
unit exakelvin = kelvin * 10 ^ 18
unit petakelvin = kelvin * 10 ^ 15
unit terakelvin = kelvin * 10 ^ 12
unit gigakelvin = kelvin * 10 ^ 9
unit megakelvin = kelvin * 10 ^ 6
unit kilokelvin = kelvin * 10 ^ 3
unit hectokelvin = kelvin * 10 ^ 2
unit decakelvin = kelvin * 10 ^ 1
unit decikelvin = kelvin * 10 ^ −1
unit centikelvin = kelvin * 10 ^ −2
unit millikelvin = kelvin * 10 ^ −3
unit microkelvin = kelvin * 10 ^ −6
unit nanokelvin = kelvin * 10 ^ −9
unit picokelvin = kelvin * 10 ^ −12
unit femtokelvin = kelvin * 10 ^ −15
unit attokelvin = kelvin * 10 ^ −18
unit zeptokelvin = kelvin * 10 ^ −21
unit yoctokelvin = kelvin * 10 ^ −24
unit yottamole = mole * 10 ^ 24
unit zettamole = mole * 10 ^ 21
unit examole = mole * 10 ^ 18
unit petamole = mole * 10 ^ 15
unit teramole = mole * 10 ^ 12
unit gigamole = mole * 10 ^ 9
unit megamole = mole * 10 ^ 6
unit kilomole = mole * 10 ^ 3
unit hectomole = mole * 10 ^ 2
unit decamole = mole * 10 ^ 1
unit decimole = mole * 10 ^ −1
unit centimole = mole * 10 ^ −2
unit millimole = mole * 10 ^ −3
unit micromole = mole * 10 ^ −6
unit nanomole = mole * 10 ^ −9
unit picomole = mole * 10 ^ −12
unit femtomole = mole * 10 ^ −15
unit attomole = mole * 10 ^ −18
unit zeptomole = mole * 10 ^ −21
unit yoctomole = mole * 10 ^ −24
unit yottacandela = candela * 10 ^ 24
unit zettacandela = candela * 10 ^ 21
unit exacandela = candela * 10 ^ 18
unit petacandela = candela * 10 ^ 15
unit teracandela = candela * 10 ^ 12
unit gigacandela = candela * 10 ^ 9
unit megacandela = candela * 10 ^ 6
unit kilocandela = candela * 10 ^ 3
unit hectocandela = candela * 10 ^ 2
unit decacandela = candela * 10 ^ 1
unit decicandela = candela * 10 ^ −1
unit centicandela = candela * 10 ^ −2
```



```
unit millicandela = candela * 10 ^ −3
unit microcandela = candela * 10 ^ −6
unit nanocandela = candela * 10 ^ −9
unit picocandela = candela * 10 ^ −12
unit femtocandela = candela * 10 ^ −15
unit attocandela = candela * 10 ^ −18
unit zeptocandela = candela * 10 ^ −21
unit yoctocandela = candela * 10 ^ −24
```

```
#distance units
unit inch = meter * 2.52 * 10 ^ −2
unit foot = meter * 3.048 * 10 ^ −1
unit yard = meter * 0.9144
unit fathom = meter * 1.8288
unit chain = meter * 2.01168 * 10
unit furlong = meter * 2.01168 * 10 ^ −2
unit cable = foot * 608
unit mile = yard * 1.609344 * 10 ^ 3

#weight units
unit dram = kilogram * 1.77185 * 10 ^ −3
unit ounce = kilogram * 2.83495 * 10 ^ −2
unit pound = kilogram * 0.45359237
```



```
#!/bin/bash

# runexamples

# by Stuart Sierra, ss2806@columbia.edu

# This is a Bash shell script.  Run it like this:
#    bash runexamples
#
# This script runs the example test programs in test/examples/
#
# Each *.in file contains Physicalc source code.  Each *.in file has a
# corresponding *.out file, which is what that program should print
# out when it is run.
#
# This will run the Physicalc interpreter on each *.in file and
# compares the printed output of that program with the corresponding
# *.out file.  If they match, it prints "OK".  If not, it prints the
# "diff" between the expected output and the actual output.  The
# actual output is saved as *.out.actual


source ./profile.sh
make

for infile in test/examples/*.in
do
    outfile=‘echo "$infile" | sed −e ’s/\.in/.out/’‘
    actual="$outfile.actual"
    echo "Testing $infile"
    java physicalc.Main $infile > $actual
    if diff −bu $outfile $actual
    then
        echo "OK"
        rm $actual
    fi
done
```

```
#!/bin/bash

# runexample

# This is a Bash shell script.  Run it like this:
#    bash runexamples [testname]
#
# This script runs a single example test program in test/examples/
#
# See "runexamples" for documentation on the example file format.
#

source ./profile.sh
make

infile=test/examples/$1.in
outfile=test/examples/$1.out

if [ ! -r $infile ]
then
    echo "Missing $infile"
    exit
fi

if [ ! -r $outfile ]
then
    echo "Missing $outfile"
    exit
fi


actual="$outfile.actual"
echo "Testing $infile"
java physicalc.Main $infile > $actual
if diff -bu $outfile $actual
then
    echo "OK"
    rm $actual
fi
```



```
/* ********************************************************************
 * grammar.g : the lexer and the parser, in ANTLR grammar for Physicalc
 *
 * ANTLR Parser Generator Version 2.7.7 (2006-11-01)
 *
 * @author Changlong Jiang, cj2214@columbia.edu
 * @author Stuart Sierra, ss2806@colmbia.edu
 *
 * @version 1.0
 * ********************************************************************/


header {
    package physicalc;
    import java.util.ArrayList;
}

/* *******************************************************************
 * LEXER *
 * ******************************************************************* */
class PhysiLexer extends Lexer;

options {
    charVocabulary = '\11'..'\177';  // Plain 7-bit ASCII
    testLiterals = false;
    k = 2;  // for >= or <= operators
}

protected DIGIT : '0'..'9';
protected LETTER : 'a'..'z' | 'A'..'Z';


/** Identifiers must begin with a letter or underscore, which may be
 * followed by any combination of letters, digits, and underscores. */
ID  options { testLiterals = true; }
    : ( LETTER | '_' ) ( LETTER | DIGIT | '_' )*;

/** Whitespace is ignored. */
WHITESPACE : (' ' | '\t' | '\f')+ { $setType(Token.SKIP); };

/** Line breaks are significant as statement separators, but are not
 * tokens on their own. */
protected NEWLINE : ('\n' | ('\r' '\n') => '\r' '\n' | '\r')
    { newline(); } ;

/** Comments begin with '#' and go to the end of the line.  Since line
 * breaks are used as statement separators, the comment text does NOT
 * include the newline. */
COMMENT : '#' ( ~( '\n' | '\r' ) )*
    { $setType(Token.SKIP); } ;

/** Statements are terminated by (any number of) newlines or
 * semicolons. */
TERMINATOR : (NEWLINE | ';')+;


/** There is no syntactic distinction among integers, decimal numbers,
 * and numbers with exponents.  They're all just numbers. */
NUMBER : ( (DIGIT)+ ( '.' (DIGIT)* )?
         | '.' (DIGIT)+
         )
         ( ('e'|'E') ('+'|'-')? (DIGIT)+ )?
```

```
         ;

/** Strings are surrounded by double quotation marks. A double
 *  quotation character may be inside a string by using two double
 *  quotation marks in a row. */
STRING  : ’"’!
              (  ~(’"’)
               | (’"’!’"’)
              )*
          ’"’!
        ;

PLUS   : ’+’;
MINUS  : ’−’;
TIMES  : ’*’;
DIVIDE : ’/’;
CARET  : ’^’;
LPAREN : ’(’;
RPAREN : ’)’;
LBRACKET : ’[’;
RBRACKET : ’]’;
LBRACE : ’{’;
RBRACE : ’}’;
COMMA  : ’,’;
EQ     : ’=’;  // we use ’set’ for assignment
NEQ     : "!=";
RELOP  : ’>’ | ’<’ | ">=" | "<=" ;


/* *******************************************************************
 * PARSER *
 * ******************************************************************* */
class PhysiParser extends Parser;

options {
    k = 2;  // needed for subscripts, function calls, etc.
    buildAST = true;
}

tokens { /* used in the abstract syntax tree */
    BLOCK;
    EXPR_LIST;
    FUNCALL;
    IF;
    LIST;
    PARAMS;
    SUBSCRIPT;
    UMINUS;
    VECTOR;
    BASEUNIT;
    DERIVEDUNIT;
}

program : (load | def | stmt)+;

/** Load statement: load the file at the path given. */
load : "load"^ STRING TERMINATOR!;

/* **************************************************
 * Definitions
 * ************************************************** */
```



```
/** Definitions. */
def : (unit_def | constant_def | alias_def | function_def)
      TERMINATOR!;

unit_def
    : "unit"! ID
        ( /* nothing − it’s a base unit */
            {#unit_def = #([BASEUNIT, "BASEUNIT"], unit_def); }
        | EQ! expr
            {#unit_def = #([DERIVEDUNIT, "DERIVEDUNIT"], unit_def); }
        )
    ;

constant_def : "constant"^ ID EQ! expr;

alias_def : "alias"^ ID "for"! ID;

function_def : "function"^ ID LPAREN! params RPAREN! TERMINATOR!
    block "done"!;

/** Parameter list for function definitions. */
params : (ID)? (COMMA! ID)*
    {#params = #([PARAMS, "PARAMS"], params); } ;

/* **************************************************
 * Statements
 * ************************************************** */

/** Statements. */
stmt : simple_stmt | compound_stmt;

/** A block of one or more statements. */
block : (stmt)+
    {#block = #([BLOCK, "BLOCK"], block); } ;

/** Simple statement: A single−line statement that must end with a
 * TERMINATOR.  An expression by itself can be a statement.*/
simple_stmt :
    ( expr
    | "return"^ expr
    | "next"^
    | "break"^
    | "set"^ lvalue EQ! expr
    | /* nothing */
    )
    TERMINATOR! ;

/** An lvalue is anything that can be assigned to with "set".
 * Variables and subscript expressions can be assigned. */
lvalue : subscript_expr | ID ;

/** Compound statement: a multi−part statement like if/then/else or
 * while.  Compound statements always end with "done". */
compound_stmt : (if_stmt | while_stmt | for_stmt) "done"! TERMINATOR!;

/** If/then/else.  These rules transfrom an if/elsif/else sequence
 * into nested IF trees.  Each IF subtree has 3 arguments: (1) a test
 * expression, (2) a "then" block, and (3) an optional "else"
 * block. */
if_stmt :
    "if"! expr "then"! TERMINATOR! block
    elsif_stmt
```

```
    {#if_stmt = #([IF,"IF"], if_stmt); } ;

elsif_stmt
    : "elsif"! expr "then"! TERMINATOR! block (elsif_stmt)
        /* need to enclose next IF in a BLOCK for tree walker */
        { #elsif_stmt = #([BLOCK,"BLOCK"], #([IF,"IF"], elsif_stmt)); }
    | else_stmt ;

else_stmt : "else"! TERMINATOR! block
    | /* nothing, but still have to include a block for the tree walker */
        { #else_stmt = #([BLOCK,"BLOCK"], else_stmt); }
    ;


/** "while" loops. */
while_stmt : "while"^ expr "do"! TERMINATOR! block;

/** "for" loops. */
for_stmt : "for"^ ID "from"! expr "to"! expr "step"! expr "do"! TERMINATOR!
           block ;


/* ****************************************************
 * Expressions
 * **************************************************** */

/** A list of expressions, separated by commas.  Used in literal lists
 * and function calls.  */
expr_list
    : expr (COMMA! expr)*
        {#expr_list = #([EXPR_LIST, "EXPR_LIST"], expr_list); }
    | /* nothing, still need a node for the tree walker */
        {#expr_list = #([EXPR_LIST, "EXPR_LIST"], expr_list); }
    ;


/** Expressions */
expr: in_expr;

/* Every binary operator is can repeat infinitely with a '*' closure.
 *
 * This parses expressions like "a < b < c" as "(< (< a b) c)", which
 * makes no sense.
 *
 * However, changing the '*' to a '?' means that anything after the
 * first operator gets ignored, which is clearly wrong.  Better let
 * the back-end decide if "(< (< a b) c)" is reasonable. */

in_expr : or_expr ( "in"^ or_expr )*;

or_expr : and_expr ( "or"^ and_expr )*;

and_expr : eq_expr ( "and"^ eq_expr )*;

eq_expr : neq_expr (EQ^ neq_expr)*;

neq_expr : rel_expr (NEQ^ rel_expr)*;

rel_expr : add_expr (RELOP^ add_expr)*;

add_expr : mul_expr ( (PLUS^ | MINUS^) mul_expr )*;
```



```
mul_expr : exp_expr ( (TIMES^ | DIVIDE^) exp_expr )*;

/** Exponentiation: tail-recursion makes it right-associative. */
exp_expr : not_expr (CARET^ exp_expr)?;

not_expr : ("not"^)? uminus_expr;  /* 'not' expressions cannot be chained */

/** Unary negation operator. Unary plus ("+") is not included because
 * it's meaningless. */
uminus_expr :
    MINUS! atom
    {#uminus_expr = #([UMINUS, "UMINUS"], uminus_expr); }
    | atom;


/** atomic expressions (highest precedence) */
atom
    : ID
    | NUMBER
    | STRING
    | list_literal
    | vector_literal
    | subscript_expr
    | funcall_expr
    | LPAREN! expr RPAREN!
    | "true"
    | "false"
    ;

/** Literal list (in square brackets) */
list_literal : LBRACKET! expr_list RBRACKET!
    {#list_literal = #([LIST,"LIST"], list_literal); };

/** Literal vector (in curly brackets, must have exactly 2 elements. */
vector_literal : LBRACE! expr COMMA! expr RBRACE!
    {#vector_literal = #([VECTOR,"VECTOR"], vector_literal); };

/** Array/list subscripts like "a[b]".  Back-end is responsible for
 * checking that the subscript evaluates to an integer.  Chained
 * subscript expressions like a[b][c] are allowed, but the first token
 * (the 'a') must be an identifier. */
subscript_expr : ID (LBRACKET! expr RBRACKET!)+
    {#subscript_expr = #([SUBSCRIPT, "SUBSCRIPT"], subscript_expr); };

/** Function calls */
funcall_expr : ID LPAREN! expr_list RPAREN!
    {#funcall_expr = #([FUNCALL, "FUNCALL"], funcall_expr); };



/* ******************************************************************
 * TREE WALKER *
 * ****************************************************************** */
class PhysiWalker extends TreeParser;

program returns [ Program p ]
{
    p = new Program();
    Node n;
}
    : ( n=node { p.insert(n); } )+
    ;
```

```
node returns [ Node n ]
{
    n = null;
    Expr e;
    Load l;
    Def d;
    Stmt s;
}
    : e=expr { n = e; }
    | l=load { n = l; }
    | s=stmt { n = s; }
    | d=def  { n = d; }
    ;

expr returns [ Expr e ]
{
    Expr a, b;
    e = null;
}
    /* Logical operators */
    : #("and" a=expr b=expr) { e = new And(a, b); }
    | #("or" a=expr b=expr)  { e = new Or(a, b); }
    | #("not" a=expr) { e = new Not(a); }
    | #("in" a=expr b=expr) { e = new In(a, b); }

    /* Relational operators */
    | #(EQ a=expr b=expr) { e = new Rel("=", a, b); }
    | #(NEQ a=expr b=expr) { e = new Rel("!=", a, b); }
    | #(op:RELOP a=expr b=expr) { e = new Rel(op.getText(), a, b); }

    /* Arithmetic operators */
    | #(PLUS a=expr b=expr) { e = new Arith("+", a, b); }
    | #(MINUS a=expr b=expr) { e = new Arith("-", a, b); }
    | #(TIMES a=expr b=expr) { e = new Arith("*", a, b); }
    | #(DIVIDE a=expr b=expr) { e = new Arith("/", a, b); }
    | #(CARET a=expr b=expr) { e = new Arith("^", a, b); }
    | #(UMINUS a=expr) { e = new Unary(a); }

    /* Other expressions */
    | a=funcall { e = a; }
    | a=subscript { e = a; }
    | a=literal { e = a; }
    | a=literal_list { e = a; }
    | i:ID { e = new Id(i.getText()); }
    ;

expr_list returns [ ExprList elist ]
{
    Expr a;
    elist = null;
}
    : #(EXPR_LIST  { elist = new ExprList(); }
          (a=expr  { elist.insert(a); }
          )*
      )
    ;

funcall returns [ FunCall f ]
{
    ExprList e;
    f = null;
```



```
}
    : #(FUNCALL i:ID e=expr_list) { f = new FunCall(i.getText(), e); }
    ;

literal returns [ Literal lit ]
{
    lit = null;
}
    : n:NUMBER { lit = new Literal(new PNumber(n.getText())); }
    | s:STRING { lit = new Literal(new PString(s.getText())); }
    | "true" { lit = new Literal(new PBoolean(true)); }
    | "false" { lit = new Literal(new PBoolean(false)); }
    ;

literal_list returns [ ExprList elist ]
{
    Expr e;
    elist = null;
}
    : #(LIST  { elist = new ExprList(); }
              #(EXPR_LIST
                (e=expr  { elist.insert(e); }
                )*
              )
       )
    ;


load returns [ Load ld ]
{
    ld = null;
}
    : #("load" file:STRING) { ld = new Load(file.getText()); }
    ;


stmt returns [ Stmt s ]
{
    s = null;
    Expr e,a,from,to,step;
    Block b,c;
}
    : #("set" a=expr e=expr) { s = new Set((LValue)a, e); }
    | "break" { s = new Break(); }
    | "next" { s = new Next(); }
    | #("return" e=expr) { s = new Return(e); }
    | #(IF a=expr b=block c=block) { s = new If(a,b,c); }
    | #("while" a=expr b=block) { s = new While(a,b); }
    | #("for" id2:ID from=expr to=expr step=expr b=block)
        { s = new For(id2.getText(), from, to, step, b); }
    ;


block returns [ Block b ]
{
    b = null;
    Node n;
}
    : #(BLOCK  { b = new Block(); }
          ( n=node { b.insert(n); } )*
       )
    ;
```

```
param_list returns [ ParamList plist ]
{
    plist = null;
}
    : #(PARAMS  { plist = new ParamList(); }
          (id:ID  { plist.insert(id.getText()); }
          )*
      )
    ;

def returns [ Def d ]
{
    d = null;
    Block b;
    Expr e;
    ParamList p;
}
    : #("constant" id1:ID e=expr) { d = new ConstantDef(id1.getText(), e); }
    | #(BASEUNIT id6:ID) { d = new UnitDef(id6.getText()); }
    | #(DERIVEDUNIT id2:ID e=expr) { d = new UnitDef(id2.getText(), e); }
    | #("function" id3:ID p=param_list b=block)
          { d = new FunctionDef(id3.getText(), p, b); }
    | #("alias" id4:ID id5:ID)
          { d = new AliasDef(id4.getText(), id5.getText()); }
    ;


subscript returns [ Access a ]
{
    a = null;
    ExprList elist;
    Expr e;
}
    : #(SUBSCRIPT id:ID  { elist = new ExprList();
                             a = new Access(id.getText(), elist); }
           (e=expr { elist.insert(e); }
           )*
       )
    ;
```



```java
import java.lang.String;
import java.io.Reader;
import java.io.FileReader;
import java.io.FileNotFoundException;
import java.util.*;
import antlr.Token;
import physicalc.*;


/** ParseFile: test the Physicalc parser on a full file on the command
 * line.
 *
 * ParseFile is an executable class that takes a single command-line
 * argument, a file name.  It reads the file and feeds it through the
 * PhysiCalc parser and prints out the Lisp-style abstract syntax tree
 * it generates.
 *
 * Run it like this:
 *
 *    java ParseFile filename
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class ParseFile {
    public static void main(String [] args) {
        Reader reader;
        try {
            reader = new FileReader(args[0]);
        } catch (FileNotFoundException err) {
            System.out.println("File not found.");
            return;
        }

        PhysiLexer lexer = new PhysiLexer(reader);
        PhysiParser parser = new PhysiParser(lexer);
        try {
            parser.program();
            System.out.println(parser.getAST().toStringList());
        } catch (Exception err) {
            System.out.println(err.toString());
        }
    }
}
```

```java
import java.lang.String;
import java.io.InputStream;
import java.io.StringReader;
import java.util.*;
import antlr.Token;
import physicalc.*;


/** TryDatum: test the Datum classes
 *
 * TryDatum will create and test various Datum objects
 *
 * Run it like this:
 *
 *    java TryDatum
 *
 */
public class TryDatum {
    public static void main(String [] args) {

                /*****************
                PNumber & PBoolean Tests − testing basic algebra and logic of in
tegers, decimals, and exponents
                *****************/

                PNumber integer = new PNumber(5);
// you can use a primitive int, float, double, long, short
                PNumber integer2 = new PNumber(new Integer(5));         //or wra
pper class Integer, Float, Double, Long, Short
                PNumber integer3 = new PNumber("5");
//or you can use a String
                PNumber decimal = new PNumber(3.14159);
                PNumber pos_exponent = new PNumber("3e4");
                PNumber neg_exponent = new PNumber("3e−4");
                Datum result = new PNumber();
                PBoolean bool = new PBoolean();

                try {

                        System.out.println("*******************************************
********************");
                        System.out.println("PNumber & PBoolean Tests:\n");
                        result = integer.add(decimal);
                        System.out.println(integer.toString()+" + "+decimal.toStr
ing()+" = "+result.toString());

                        result = pos_exponent.sub(integer);
                        System.out.println(pos_exponent.toString()+" − "+integer.
toString()+" = "+result.toString());

                        result = integer.mul(neg_exponent);
                        System.out.println(integer.toString()+" * "+neg_exponent.
toString()+" = "+result.toString());

                        result = decimal.div(pos_exponent);
                        System.out.println(decimal.toString()+" / "+pos_exponent.
toString()+" = "+result.toString());

                        result = integer.pow(decimal);
                        System.out.println(integer.toString()+" ^ "+decimal.toStr
ing()+" = "+result.toString());
```



```java
                        bool = new PBoolean(integer.equals(integer2));
                        System.out.println(integer.toString()+" == "+integer2.toS
tring()+" = "+bool.toString());

                        bool = integer.lessThan(decimal);
                        System.out.println(integer.toString()+" < "+decimal.toStr
ing()+" = "+bool.toString());
                        System.out.println("*******************************************
********************");

                } catch (TypeError err) {
                        System.out.println(err.toString());
                }

                /*****************
                PUnit Tests − testing basic algebraic symbolic computation of un
its
                *****************/

                PUnit second = new PUnit("second");
                PUnit meter = new PUnit("meter");

                PNumber num1 = new PNumber("0.5");
                PNumber num2 = new PNumber("30");

                Datum result2 = new PUnit();

                try {

                        System.out.println("PUnit Tests:\n");

                        PUnit minute = new PUnit("minute",second.mul(new PNumber(
"60")));  // minute = 60 * seconds
                        System.out.println("minute: "+minute.getConversion().toStr
ing()+minute.toString());

                        PUnit foot = new PUnit("foot",meter.mul(new PNumber("0.304
8")));   // foot = 0.3048 * meters
                        System.out.println("foot: "+foot.getConversion().toString(
)+foot.toString());

                        Datum accel = num1.mul(foot.div(minute.pow(new PNumber("
2"))));  // accel = 0.5 * foot / minute^2 = 0.0000423 * meter / second^2
                        System.out.println("accel: "+accel.toString());

                        Datum time = num2.mul(second);  // time = 30 * second
                        System.out.println("time: "+time.toString());

                        Datum veloc = time.mul(accel);  // veloc = time * accel
= 30 * second * 0.0000423 * meter / second^2 = 0.0762 * meter / second
                        System.out.println("veloc: "+veloc.toString());


                        System.out.println("*******************************************
********************");

                } catch (TypeError err) {
                        System.out.println(err.toString());
                }

                /*****************
                PUnitPair Tests − testing algebraic manipulation of number−unit
```

```
pairs
                ****************/

                PUnitPair pair1 = new PUnitPair(integer,second);
                PUnitPair pair2 = new PUnitPair(decimal,meter);
                PUnitPair pair3 = new PUnitPair(integer,meter);
                Datum result3 = new PUnitPair();

                try {

                        System.out.println("PUnitPair Tests:\n");
                        result3 = pair2.add(pair3);  //note that in practice, ad
dition/subraction with unit pairs must catch exceptions (incompatible units)
                        System.out.println(pair2.toString()+" + "+pair3.toString(
)+" = "+result3.toString());

                        result3 = pair2.sub(pair3);
                        System.out.println(pair2.toString()+" − "+pair3.toString(
)+" = "+result3.toString());

                        result3 = pair1.mul(pair2);
                        System.out.println(pair1.toString()+" * "+pair2.toString(
)+" = "+result3.toString());

                        result3 = pair2.div(pair3);
                        System.out.println(pair2.toString()+" / "+pair3.toString(
)+" = "+result3.toString());

                        bool = new PBoolean(pair1.equals(pair2));
                        System.out.println(pair1.toString()+" == "+pair2.toString
()+" = "+bool.toString());
                        System.out.println("******************************************
*********************");

                } catch (TypeError err) {
                        System.out.println(err.toString());
                }


                /****************
                PVector Tests − testing basic algebra and logic of vectors (2d x
,y component) integers, decimals, and exponents
                ****************/

                /*
                PVector vector1 = new PVector(integer,decimal);
                PVector vector2 = new PVector(decimal,pos_exponent);
                PVector vector3 = new PVector(neg_exponent,integer);
                PVector result4 = new PVector();

                try {

                        System.out.println("PVector Tests:\n");
                        result4 = vector1.add(vector2);
                        System.out.println(vector1.toString()+" + "+vector2.toSt
ring()+" = "+result4.toString());

                        result4 = vector3.sub(vector1);
                        System.out.println(vector3.toString()+" − "+vector1.toSt
ring()+" = "+result4.toString());

                        result4 = vector2.mul(vector3);
```



```
                        System.out.println(vector2.toString()+" * "+vector3.toSt
ring()+" = "+result4.toString());

                        result4 = vector1.div(vector2);
                        System.out.println(vector1.toString()+" / "+vector2.toSt
ring()+" = "+result4.toString());

                        result4 = vector3.pow(vector1);
                        System.out.println(vector3.toString()+" ^ "+vector1.toSt
ring()+" = "+result4.toString());

                        bool = new PBoolean(vector2.equals(vector3));
                        System.out.println(vector2.toString()+" == "+vector3.toS
tring()+" = "+bool.toString());

                        bool = vector1.lessThan(vector2);
                        System.out.println(vector1.toString()+" < "+vector2.toSt
ring()+" = "+bool.toString());
                        System.out.println("**********************************
****************************");

                } catch (TypeError err) {
                        System.out.println(err.toString());
                }

                */

                /****************
                Plist Tests − testing adding and removing PNumber and PUnitPair
elements from list
                ****************/

                PList list1 = new PList();
                list1.push(integer);
                list1.push(second);
                list1.push(pair1);
                list1.push(pair2);

                try {

                        System.out.println("PList Tests:\n");
                        System.out.println(list1.toString());
                        System.out.println("******************************************
*********************");

                } catch (TypeError err) {
                        System.out.println(err.toString());
                }

                /****************
                PString Tests − testing concatenation and lexigraphical comparis
ons of strings
                ****************/

                PString string1 = new PString("apple");
                PString string2 = new PString("sauce");
                PString string3 = new PString("power");
                Datum result5 = new PString();

                try {

                        System.out.println("PString Tests:\n");
```

```
                        result5 = string1.add(string2);
                        System.out.println(string1.toString()+" + "+string2.toStr
ing()+" = "+result5.toString());

                        bool = new PBoolean(string2.equals(string3));
                        System.out.println(string2.toString()+" == "+string3.toSt
ring()+" = "+bool.toString());

                        bool = string3.lessThan(string1);
                        System.out.println(string3.toString()+" < "+string1.toStr
ing()+" = "+bool.toString());
                        System.out.println("******************************************
********************");

                } catch (TypeError err) {
                        System.out.println(err.toString());
                }
    }
}
```



```
import java.lang.String;
import java.io.InputStream;
import java.io.StringReader;
import java.util.*;
import antlr.Token;
import physicalc.*;


/** TryLexer: test the PhysiCalc lexer on the command line.
 *
 * TryLexer is an executable class that takes a single command-line
 * argument, a string.  It feeds the string through PhysiCalc's
 * ANTLR-generated lexer and prints out the list of tokens recognized.
 *
 * Run it like this:
 *
 *    java TryLexer "3 + 4"
 *
 * And you get output like this:
 *
 *     ["3",<11>,line=1,col=1]
 *     ["+",<13>,line=1,col=3]
 *     ["4",<11>,line=1,col=5]
 *
 * On each line, the first field is the string matched, the second is
 * the number of the token type, which you can find in the
 * ANTLR-generated file src/physicalc/PhysiLexerTokenTypes.txt .
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class TryLexer {
    public static void main(String [] args) {
        List<Token> tokens = new ArrayList<Token>();
        try {
            String test = args[0];
            StringReader reader = new StringReader(test);
            PhysiLexer lexer = new PhysiLexer(reader);

            Token t;
            while ((t = lexer.nextToken()) != null) {
                if (t.getType() == antlr.Token.EOF_TYPE) { break; }
                tokens.add(t);
            }
        } catch (antlr.TokenStreamException err) {
            System.out.println(err.toString());
        }

        for (Token t : tokens) {
            System.out.println(t.toString());
        }
    }
}
```

```java
import java.lang.String;
import java.io.InputStream;
import java.io.StringReader;
import java.util.*;
import antlr.Token;
import physicalc.*;


/** TryParser: test the Physicalc parser on the command line.
 *
 * TryParser is an executable class that takes a single command-line
 * argument, a string.  It feeds the string through the PhysiCalc
 * parser and prints out the Lisp-style abstract syntax tree it
 * generates.
 *
 * Run it like this:
 *
 *    java TryAst "Physicalc code here"
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class TryParser {
    public static void main(String [] args) {
        StringReader reader = new StringReader(args[0]);
        PhysiLexer lexer = new PhysiLexer(reader);
        PhysiParser parser = new PhysiParser(lexer);
        try {
            parser.program();
            System.out.println(parser.getAST().toStringList());
        } catch (Exception err) {
            System.out.println(err.toString());
        }
    }
}
```



```java
package physicalc;

import java.lang.*;
import java.util.*;

/** Access implements list access with [] subscripts.
 *
 * @see Node
 * @see PList
 * @see Set
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Access extends Expr implements LValue {

    private String id;
    private ExprList subscripts;

    public Access(String identifier, ExprList subExprs) {
        //System.out.println("Constructing an Access");
        id = identifier;
        subscripts = subExprs;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        //System.out.println("Calling eval() in Access");

        /* Look up id in the symbol tables -- global first, then local
         * -- or throw UndefinedError it it's not in either. */
        RuntimeObject r;
        r = globals.get(id);
        if (r == null) {
            r = locals.get(id);
            if (r == null) {
                throw new UndefinedError(id);
            }
        }


        /* Get the value stored in the symbol table, check that it is
         * an instanceof Variable.  If not, throw an
         * InterpreterError. */
        Datum value;
        if (r instanceof Variable) {
            /* Cast the object from the symbol table to a Variable. */
            Variable var = (Variable)r;
            value = var.getValue();
        } else if (r instanceof Constant) {
            Constant constant = (Constant)r;
            value = constant.getValue();
        } else {
            throw new InterpreterError("Symbol '" + id + "' is not a variable.");
        }


        int index;
        for (Expr e : subscripts.getContents()) {
            index = ((PNumber)e.eval(globals,locals)).toInt();
            if (value instanceof PList) {
                value = ((PList)value).getIndex(index);
            } else {
                throw new InterpreterError("Tried to access element in a non-list.");
```

```java
            }
        }


        return value;
    }



    public void setValue(SymbolTable globals, SymbolTable locals,
                         Datum newValue) {
        //System.out.println("Calling setValue() in Access");

        /* Look up id in the local symbol table, or throw
         * UndefinedError it it’s not there. */
        RuntimeObject r;
        r = locals.get(id);
        if (r == null) {
            throw new UndefinedError(id);
        }


        /* Get the value stored in the symbol table, check that it is
         * an instanceof Variable.  If not, throw an
         * InterpreterError. */
        Variable var;
        if (r instanceof Variable) {
            /* Cast the object from the symbol table to a Variable. */
            var = (Variable)r;
        } else {
            throw new InterpreterError("Symbol ’" + id + "’ is not a variable.");
        }


        Datum value = var.getValue();
        PList list = null;
        int index = 0;
        for (Expr e : subscripts.getContents()) {
            index = ((PNumber)e.eval(globals,locals)).toInt();
            if (value instanceof PList) {
                list = (PList)value;
                try {
                    value = ((PList)value).getIndex(index);
                } catch (java.lang.IndexOutOfBoundsException error) {
                    value = null;
                }
            } else {
                throw new InterpreterError("Tried to access element in a non−list.");
            }
        }

        if (list == null) {
                throw new InterpreterError("Tried to access element in a non−list.");

        } else {
            list.set(index, newValue);
        }
    }

}
```



```java
package physicalc;

/** @author Ici Li, il2117@columbia.edu
 */
public class AliasDef extends Def {

    private String newSymb;
    private String oldSymb;

    public AliasDef(String newSymbol, String oldSymbol) {
        newSymb = newSymbol;
        oldSymb = oldSymbol;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        //lookup old symbol in global symbol table
        //then, if it’s not defined, throw an error
        //add new entry to global symbol table with newSymb as the symbol
        //value as value

        RuntimeObject R = globals.get(oldSymb);

        if(R == null) {
            throw new UndefinedError(oldSymb);
        }
        else {
            globals.put(newSymb, R);
        }

        return null;
    }
}
```

```java
package physicalc;

/** And is a node implementing the "and" logical operator.
 *
 * @see Node
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class And extends Logical  {
    private Expr left;
    private Expr right;

    public And(Expr leftOperand, Expr rightOperand) {
        left = leftOperand;
        right = rightOperand;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        if (!(left.eval(globals, locals).isTrue())) {
            /* short-circuit if left operand is false */
            return new PBoolean(false);
        } else if (right.eval(globals, locals).isTrue()) {
            return new PBoolean(true);
        } else {
            return new PBoolean(false);
        }
    }
}
```



```java
package physicalc;

import java.lang.String;

/** Arith is a node implementing "+","-","*","/", and "^"
 *
 * @see Node
 * @author Changlong Jiang cj2214@columbia.edu
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Arith extends Op {
    private Expr left;
    private Expr right;
    private String op;

    public Arith(String operator, Expr leftOperand, Expr rightOperand) {
        op = operator;
        left = leftOperand;
        right = rightOperand;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        Datum leftValue = left.eval(globals, locals);
        Datum rightValue = right.eval(globals, locals);

        /* Datum classes take care of type checking. */
        if (op.equals("+")) {
            return leftValue.add(rightValue);
        } else if (op.equals("-")) {
            return leftValue.sub(rightValue);
        } else if (op.equals("*")) {
            return leftValue.mul(rightValue);
        } else if (op.equals("/")) {
            return leftValue.div(rightValue);
        } else if (op.equals("^")) {
            return leftValue.pow(rightValue);
        } else {
            /* This will only happen if the tree walker is wrong. */
            throw new InterpreterError("GHASTLY ERROR: Arith class with invalid operator.");
        }
    }
}
```

```java
package physicalc;

import java.util.ArrayList;
import java.util.List;

/** A Block is container for a list of Nodes.  It is used in two
 * places: 1) the body of a loop, and 2) the body of a function.
 *
 * Evaluating a block evaluates all its sub-nodes in order, and
 * returns the value of the last node.
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Block extends Stmt {

    private ArrayList<Node> contents;

    public Block() {
        //System.out.println("Constructing a Block");
        contents = new ArrayList<Node>();
    }

    public void insert(Node n) {
        //System.out.println("Adding to a Block");
        contents.add(n);
    }

    public List<Node> getContents() {
        return contents;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        //System.out.println("Calling eval() in Block");
        //System.out.println("   the block has " + ((Integer)contents.size()).to
String() + " nodes");

        Datum result = null;
        for (Node n : contents) {
            //System.out.println("Executing a Node inside a Block");
            result = n.eval(globals, locals);
        }
        return result;
    }
}
```



```java
package physicalc;

/** BoundsError is raised when you attempt to access a value beyond
 * the end of a list or vector.
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
class BoundsError extends InterpreterError {}
```

```
package physicalc;

/** "break" statement
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Break extends Stmt {

    public Break() {
        ;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        throw new BreakSignal();
    }
}
```



```
package physicalc;

/** Signal used to break out of a loop.
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class BreakSignal extends ControlSignal {

    public BreakSignal() {
        ;
    }
}
```

```java
package physicalc;


/** A Constant stores a value globally.  It cannot be changed.
 *
 * @see SymbolTable
 * @see ConstantDef
 * @author Ici Li, il2117@columbia.edu
 */
public class Constant implements RuntimeObject {

    Datum constant1;

    public Constant() {
        constant1 = null;
    }

    public Constant(Datum initialValue) {
        constant1 = initialValue;
    }

    public Datum getValue() {
        return constant1; // remove
    }
}
```



```java
package physicalc;

/** @author Ici Li, il2117@columbia.edu
 */
public class ConstantDef extends Def {

    private String id1;
    private Expr valueExpr1;

    public ConstantDef(String id, Expr valueExpr) {
        id1 = id;
        valueExpr1 = valueExpr;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        Constant c1 = new Constant(valueExpr1.eval(globals, locals));

        RuntimeObject R = c1;

        globals.put(id1, R);

        return null;
    }
}
```

```
package physicalc;

/** ControlSignal is an abstract base class for "exceptions" that are
 * used to signal changes in the control flow of a Physicalc program:
 * "break", "next", and "return" statements.
 *
 * This is an abuse of the Java exception mechanism, but it's the
 * easiest way to unwind the stack, since Java does not provide a
 * general-purpose condition system like Common Lisp.
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public abstract class ControlSignal extends RuntimeException {
}
```



```
package physicalc;

import java.lang.*;

/** Datum is an abstract base class for all data objects in a
 * Physicalc program.
 *
 * The methods in Datum just raise errors.  Sub-classes must override
 * the supported operations.
 * @author Stuart Sierra, ss2806@columbia.edu
*/
public abstract class Datum {

    /** Returns the result of this + that.  Does not modify this. */
    public Datum add(Datum that) throws TypeError {
        //System.out.println("called Datum#add");
        throw new TypeError("+", this, that);
    }

    /** Returns the result of this - that.  Does not modify this. */
    public Datum sub(Datum that) throws TypeError {
        throw new TypeError("-", this, that);
    }

    /** Returns the result of this * that.  Does not modify this. */
    public Datum mul(Datum that) throws TypeError {
        throw new TypeError("*", this, that);
    }

    /** Returns the result of this / that.  Does not modify this. */
    public Datum div(Datum that) throws TypeError {
        throw new TypeError("/", this, that);
    }

    /** Returns the result of this ^ that.  Does not modify this. */
    public Datum pow(Datum that) throws TypeError {
        throw new TypeError("^", this, that);
    }

    /** Returns the result of the unary minus operator, (- this).
     * Does not modify this. */
    public Datum neg() throws TypeError {
        throw new TypeError("unary -", this, null);
    }

    /** Returns true if "that" is the same type and has the same value
     * as this. */
    public boolean equals(Object that) {
        return false;  // Default; sub-classes should override
    }

    /** Returns true if this object is "true" in the Physicalc sense.
     * Anything that is not the literal boolean "false" is considered
     * true in Physicalc. */
    public boolean isTrue() {
        return true;
    }

    public PBoolean lessThan(Datum that) throws TypeError {
        throw new TypeError("<", this, that);
    }
```

```java
    public PBoolean lessEqual(Datum that) throws TypeError {
        throw new TypeError("<=", this, that);
    }

    public PBoolean greaterThan(Datum that) throws TypeError {
        throw new TypeError(">", this, that);
    }

    public PBoolean greaterEqual(Datum that) throws TypeError {
        throw new TypeError(">=", this, that);
    }

    /** Returns a string representation of this Datum suitable for
     * display in program output. */
    public String toString() {
        return "Datum";  // sub-classes must override
    }

    /** For lists, returns the nth item in the collection.  For
     * strings, returns the nth character. For vectors, index 0
     * returns the x component, and index 1 returns the y
     * component. */
    public Datum getIndex(int index) throws TypeError, BoundsError {
        throw new TypeError("[]", this, this);
    }

}
```



```java
package physicalc;

/** Def is an abstract base class for all definition nodes.
 *
 * @see Node
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public abstract class Def extends Node {
}
```

```
package physicalc;

/** @author Ici Li, il2117@columbia.edu */
public class ExitFunction extends Function {

    public ExitFunction() { ; }

    public Datum call(SymbolTable globals, SymbolTable locals, ExprList argument
s) {
//      System.err.println("Calling call() in ExitFunction");
      System.exit(0);
      return null;
  }

}
```



```
package physicalc;

/** Expr is an abstract base class for all expression nodes.
 *
 * @see Node
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public abstract class Expr extends Node {
}
```

```java
package physicalc;

import java.util.ArrayList;
import java.util.List;

/** An ExprList is container for a list of Expr objects.  It is used
 * in two places: 1) list literals, and 2) function calls.
 *
 * For a list literal, you call the "eval" method and ExprList returns
 * a new PList containing the results of all the expressions inside
 * it.
 *
 * For a function call, the FunCall object can call the getContents()
 * method to retrieve the Expr child nodes directly.
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class ExprList extends Expr {

    private ArrayList<Expr> contents;

    public ExprList() {
        //System.out.println("Constructing an ExprList");
        contents = new ArrayList<Expr>();
    }

    public void insert(Expr e) {
        //System.out.println("Adding to an ExprList");
        contents.add(e);
    }

    public List<Expr> getContents() {
        return contents;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        //System.out.println("Calling eval() in ExprList");

        PList result = new PList();
        for (Expr e : contents) {
            result.push(e.eval(globals, locals));
        }
        return result;
    }
}
```



```java
package physicalc;

import java.lang.String;
/**this is for class
 *  for identifier from expressions1 to experssion2 step expression3 do
 *        statements
 *  done
 *
 * @see Node
 * @author Changlong Jiang cj2214@columbia.edu
 * @author Stuart Sierra, ss2806@columbia.edu
 */


public class For extends Stmt {
    String idname;
    Expr expr1,expr2,expr3;
    Block block1;

    public For(String id,Expr fromExpr, Expr toExpr, Expr stepExpr,
               Block b) {
        idname = id;
        expr1 = fromExpr;
        expr2 = toExpr;
        expr3 = stepExpr;
        block1 = b;
    }



    public Datum eval(SymbolTable globals, SymbolTable locals) {

        Datum from = expr1.eval(globals,locals);
        Datum to = expr2.eval(globals,locals);
        Datum step = expr3.eval(globals,locals);

        Id id = new Id(idname);

        id.setValue(globals, locals, from);

        while( id.eval(globals,locals).lessEqual(to).isTrue() ){


          try {
                block1.eval(globals,locals);
          }
          catch (BreakSignal breaksignal){
                break;
          }
          catch (NextSignal nextsignal) {
                continue;
          }

            id.setValue(globals,locals, id.eval(globals,locals).add(step));

        }
        return null; // remove
    }
}
```

```java
package physicalc;

import java.lang.*;
import java.util.*;

/** FunCall implements a function call.
 *
 * @see Node
 * @see Function
 * @author Brian Foo, bwf2101@columia.edu
 */
public class FunCall extends Expr {

        private String functionName;
        private ExprList argumentList;
        private Function func;

    public FunCall(String f, ExprList al) {
                functionName = f;
                argumentList = al;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {

        //System.out.println("Calling eval() in FunCall");

        /* Look up functionName in the global symbol table,
         * throw UndefinedError if it’s not there. */
         if ( globals.get(functionName).equals(null) ) {
                throw new UndefinedError(functionName);
         }

        /* Get the object out of the symbol table, check that it’s an
         * instanceof Function, then cast it to a Function. */
         if ( globals.get(functionName) instanceof Function ) {
                 func = (Function) globals.get(functionName);
         } else {
                 throw new InterpreterError("FunCall on a non−Function object");
         }

        /* Call the function’s "call" method, passing in the global
         * symbol table and the argument list. */
         return func.call(globals,locals,argumentList);

    }
}
```



```java
package physicalc;

import java.lang.*;
import java.util.*;

/** A Function object stores a user-defined function.  Sub-classes may
 * implement built-in functions.
 *
 * @see SymbolTable
 * @see FunCall
 * @author Brian Foo, bwf2101@columia.edu
 */
public class Function implements RuntimeObject {

        private ParamList parameterList;
        private Block bodyStatements;

    /** Protected default constructor; only built-in function may be
     * created without a parameter list or block. */
    protected Function() { ; }

    public Function(ParamList pl, Block bs) {
                parameterList = pl;
                bodyStatements = bs;
    }

    public Datum call(SymbolTable globals, SymbolTable locals, ExprList argument
s) {


        /* Check that the arguments list is the same length as the
         * parameter list; throw error if it’s not. */
         if ( arguments.getContents().size() != parameterList.getContents().size
() ) {
                throw new InterpreterError("Function called with improper number of arguments"
);
         }

         /* Create a new SymbolTable for local variables.  */
         SymbolTable function_locals = new SymbolTable();

        /* For each name in the parameter list, create a new Variable
         * and add it to the local SymbolTable you just created */
         for (Iterator it = parameterList.getContents().iterator(); it.hasNext()
; ) {
                RuntimeObject r = new Variable();
                function_locals.put( (String) it.next(), r );
         }

        /* Evaluate each argument in the ExprList and assign its value
         * to one of the local Variables you just created. */
         Iterator argIt = arguments.getContents().iterator();
         for (Iterator it2 = parameterList.getContents().iterator(); it2.hasNext
(); ) {
                ((Variable)function_locals.get((String) it2.next())).setValue(((
Expr)argIt.next()).eval(globals,locals));
         }

        /* Call "eval" on the Block of body statements, passing in the
         * global symbol table and the local symbol table you created.
         * Return the value of "eval". */
         try {
```

```
                return bodyStatements.eval(globals,function_locals);
        } catch (ReturnSignal rs) {
                return rs.getValue();
        }

    }
}
```



```
package physicalc;

/** Function Definitions
 * @author Brian Foo, bwf2101@columia.edu
 */
public class FunctionDef extends Def {

        private String id;
        private ParamList paramList;
        private Block bodyBlock;

    public FunctionDef(String i, ParamList pl, Block bb) {
        id = i;
        paramList = pl;
        bodyBlock = bb;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {

                RuntimeObject func = new Function(paramList,bodyBlock);
        globals.put(id,func);

        return null; // definitions always return null
    }
}
```

```
package physicalc;

/**
 * @author Brian Foo, bwf2101@columia.edu
 */
public class GetNumberFunction extends Function {

    public GetNumberFunction() { ; }

    public Datum call(SymbolTable globals, SymbolTable locals, ExprList argument
s) {

                //System.out.println("Calling call() in GetNumberFunction");

                if (arguments.getContents().size() != 1) {
                        throw new InterpreterError("Cannot call getNumber on more than one
 argument");
                }

                Expr expr = arguments.getContents().get(0);

                Datum pair = expr.eval(globals,locals);

                if (pair instanceof PUnitPair) {
                        return ((PUnitPair)pair).getNumber();
                } else {
                        throw new InterpreterError("Must call GetNumberFunction on a Unit
Pair");
                }
    }

}
```



```
package physicalc;

/**
 * @author Brian Foo, bwf2101@columia.edu
 */
public class GetUnitFunction extends Function {

    public GetUnitFunction() { ; }

    public Datum call(SymbolTable globals, SymbolTable locals, ExprList argument
s) {

                //System.out.println("Calling call() in GetUnitFunction");

                if (arguments.getContents().size() != 1) {
                        throw new InterpreterError("Cannot call getUnit on more than one arg
ument");
                }

                Expr expr = arguments.getContents().get(0);

                Datum pair = expr.eval(globals,locals);

                if (pair instanceof PUnitPair) {
                        PUnit u = ((PUnitPair)pair).getUnit();
                        u.setUnitMode();
                        return u;
                } else {
                        throw new InterpreterError("Must call GetUnitFunction on a UnitPair
");
                }
    }

}
```

```java
package physicalc;

import java.lang.String;

/** Id is a node implementing any source-code identifier.  It also
 * implements LValue, so it can be assigned in a "Set" statement.
 *
 * @see Node
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Id extends Expr implements LValue {

    private String name;

    public Id(String idName) {

        name = idName;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {

        /* Look up idName in the local symbol table, or throw
         * UndefinedError it it's not there. */

        /* Check the type of the object you got with instanceof.  If
         * it's a Variable or Constant, get the value and return it.
         * If it's a Unit, return the Unit.  Anything else, throw an
         * InterpreterError. */

        RuntimeObject R = locals.get(name);

        if(R == null) {
            R = globals.get(name);
        }
        if(R == null) {
            throw new UndefinedError(name);
        }

        if(R instanceof Variable) {
            return ((Variable)R).getValue();
        } else if (R instanceof Constant) {
            return ((Constant)R).getValue();
        }
        else if(R instanceof Unit) {
            return ((Unit)R).getValue();
        }
        else {
                throw new InterpreterError("Tried to get value of a non-Variable/Constant/Unit"
);
        }
    }

    public void setValue(SymbolTable globals, SymbolTable locals,
                         Datum newValue) {
        //System.out.println("Calling setValue() in Name");

        RuntimeObject r;
        r = locals.get(name);
        if (r == null) {
            r = new Variable(newValue);
            locals.put(name, r);
        } else if (r instanceof Variable) {
```



```java
            ((Variable)r).setValue(newValue);
        } else if (globals.get(name) != null) {
            throw new InterpreterError("Tried to assign to a non-variable.");
        } else {
            throw new UndefinedError(name);
        }

    }
}
```

```java
package physicalc;

/** this is If Class
 *  Syntax: if expression1 then
 *              statements1
 *           elsif expression2 then
 *              statements2
 *                       else
 *                          statements3
 *                       done
 *
 *
 * @see Node
 * @author Changlong Jiang cj2214@columbia.edu
 * @author Stuart Sierra, ss2806@columbia.edu
 */

public class If extends Stmt {

    private Expr expr1;
    private Block block1;
    private Block block2;
    public If(Expr condition, Block thenBlock, Block elseBlock) {
        expr1 = condition;
        block1 = thenBlock;
        block2 = elseBlock;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        //System.out.println("Calling eval() in If");

        if (expr1.eval(globals, locals).isTrue()) {
            //System.out.println("'If' condition was true; executing 'then' bloc
k.");

            return block1.eval(globals,locals);
        }
        else {
            return block2.eval(globals,locals);
        }

    }
}
```



```java
package physicalc;

/** In is a node implementing the "in" unit-conversion operator.
 *
 * @see Node
 * @author Brian Foo, bwf2101@columia.edu
 */
public class In extends Op  {
    private Expr left;
    private Expr right;

    public In(Expr leftOperand, Expr rightOperand) {
        left = leftOperand;
        right = rightOperand;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {

                System.err.println("Calling eval() in In");

                Datum leftUnit = left.eval(globals,locals);
                Datum rightUnit = right.eval(globals,locals);

                if (rightUnit instanceof PUnit) {

                        String fromName;
                        String fromBase;
                        PNumber fromConversion;
                        PNumber fromNumber;

                        String toName = ((PUnit)rightUnit).getName();
                        String toBase = ((PUnit)rightUnit).getBaseUnit();
                        PNumber toConversion = ((PUnit)rightUnit).getConversion(
);

                        if (leftUnit instanceof PUnit) {
                                fromName = ((PUnit)leftUnit).getName();
                                fromBase = ((PUnit)leftUnit).getBaseUnit();
                                fromConversion = ((PUnit)leftUnit).getConversion
();
                                fromNumber = new PNumber("1");
                        } else if (leftUnit instanceof PUnitPair) {
                                fromName = ((PUnitPair)leftUnit).getUnit().getNa
me();
                                fromBase = ((PUnitPair)leftUnit).getUnit().getBa
seUnit();
                                fromConversion = ((PUnitPair)leftUnit).getUnit()
.getConversion();
                                fromNumber = ((PUnitPair)leftUnit).getNumber();
                        } else {
                                throw new InterpreterError("Left operand in 'in' must be a
 unit or number*unit");
                        }

                        //System.err.println("toName: "+toName+", toBase: "+toBa
se+", toConv: "+toConversion.toString() );
                        //System.err.println("fromName: "+fromName+", fromBase:
"+fromBase+", fromConv: "+fromConversion.toString()+", fromNumber: "+fromNumber.
toString());

                        if ( fromBase.equals(toBase) ) {
                                return new PUnitPair( ((fromConversion.mul(fromN
```

```java
umber)).div(toConversion)), (PUnit)rightUnit, true );
                        } else {
                                throw new InterpreterError("Base units do not match with
'"+fromName+"' in "+toName+"'");
                        }

                } else {
                        throw new InterpreterError("Right operand in 'in' must be a unit");
                }

    }
}
```



```java
package physicalc;

import java.io.*;
import antlr.CommonAST;
import antlr.collections.AST;

/** An Interpreter object is responsible for running Physicalc code.
 * It has input, output, and error streams, which default to STDIN,
 * STDOUT, and STDERR, respectively; but may be changed by the calling
 * code for testing.
 *
 * To use this class, create an instance of it and call "eval",
 * passing in a stream for the code you want to run.
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Interpreter {

    private Reader in;

    private PrintWriter out;

    private PrintWriter err;

    /** Constructor.  Creates a new interpreter instance.  Input,
     * output, and error streams default to system STDIN, STDOUT, and
     * STDERR, respectively. */
    public Interpreter() { ; }

    /** Changes the stream that this Interpreter uses as its standard
     * input. */
    public void setInputStream(InputStream inputStream) {
        System.setIn(inputStream);
    }

    /** Changes the stream that this Interpreter uses as its standard
     * output. */
    public void setOutputStream(OutputStream outputStream) {
        System.setOut(new PrintStream(outputStream));
    }

    /** Changes the stream that this Interpreter uses as its standard
     * error. */
    public void setErrorStream(OutputStream errorStream) {
        System.setErr(new PrintStream(errorStream));
    }

    /** eval() executes Physicalc source code.
     *
     * @param code A Reader containing Physicalc source code.  For
     * normal use this would be a file stream, but it could be a
     * string reader for testing.  It could even be standard input.
     */
    public void eval(Reader code) {
        try {
            PhysiLexer lexer = new PhysiLexer(code);
            PhysiParser parser = new PhysiParser(lexer);

            parser.program();

            CommonAST parseTree = (CommonAST)parser.getAST();
```

```java
            PhysiWalker walker = new PhysiWalker();
            Program p = walker.program(parseTree);

            SymbolTable globals = setupGlobalSymbols();
            SymbolTable topLevel = new SymbolTable();
            Datum result = p.eval(globals, topLevel);

            // Print the result of the last expression, for testing only.
            if (result == null) {
                //System.out.println("null");
            } else {
                //System.out.println(result.toString());
            }

        } catch(Exception e) {
            //System.out.println(e.toString());
        }
    }

    private SymbolTable setupGlobalSymbols() {
        SymbolTable globals = new SymbolTable();
        globals.put("print", ((RuntimeObject)(new PrintFunction())));
        globals.put("nprint", ((RuntimeObject)(new NPrintFunction())));
        globals.put("toInt", ((RuntimeObject)(new ToIntFunction())));
        globals.put("toString", ((RuntimeObject)(new ToStringFunction())));
        globals.put("getUnit", ((RuntimeObject)(new GetUnitFunction())));
        globals.put("getNumber", ((RuntimeObject)(new GetNumberFunction())));
        return globals;
    }
}
```



```java
package physicalc;

import java.lang.String;

/** General parent class for all errors generated by user code.
 * @author Stuart Sierra, ss2806@columbia.edu
 */
class InterpreterError extends RuntimeException {

    private String message;

    public InterpreterError() {; }

    public InterpreterError(String errorMessage) {
        message = errorMessage;
    }

    public String toString() {
        return message;
    }
}
```

```java
package physicalc;

/** LValue is an interface implemented by any Node that can be
 * assigned a value in a "set" statement.  Identifiers (i.e. instances
 * of Id) and list items (i.e. instances of Access) are both LValues.
 *
 * @see Id
 * @see Access
 * @see Set
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public interface LValue {

    public void setValue(SymbolTable globals, SymbolTable locals,
                         Datum newValue);
}
```



```java
package physicalc;

/** Literal is a node implementing any source-code literal, such as a
 * number or a list.
 *
 * @see Node
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Literal extends Expr {
    private Datum value;

    public Literal(Datum theValue) {
        value = theValue;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        //System.out.println("Calling eval() in Literal");
        return value;
    }
}
```

```java
package physicalc;

import java.lang.String;
import java.io.*;
import antlr.CommonAST;
import antlr.collections.AST;

/** Load implements the "load" statement.
 *
 * @see Node
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Load extends Node {

    private String file;

    public Load(String filename) {
        file = filename;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        Reader input;
        try {
            input = new FileReader(file);
        } catch (FileNotFoundException e) {
            //System.out.println("File '" + file + "' not found.");
            System.exit(1);
            return null; // to keep the compiler happy
        }

        try {
            PhysiLexer lexer = new PhysiLexer(input);
            PhysiParser parser = new PhysiParser(lexer);

            parser.program();

            CommonAST parseTree = (CommonAST)parser.getAST();

            PhysiWalker walker = new PhysiWalker();
            Program p = walker.program(parseTree);

            p.eval(globals, locals);

        } catch(Exception e) {
            //System.out.println(e.toString());
        }

        return null;
    }
}
```



```java
package physicalc;

/** Logical is an abstract base class for all logical operator nodes.
 *
 * @see Node
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public abstract class Logical extends Expr  {
}
```

```java
package physicalc;

import java.io.*;
import java.lang.String;

/** Main executable class
 * @author Stuart Sierra, ss2806@columbia.edu
*/
class Main {

    public static void main(String[] args) {
        // TODO: add code to check syntax of command line

        Reader input;
        try {
            input = new FileReader(args[0]);
        } catch (FileNotFoundException e) {
            //System.out.println("File '" + args[0] + "' not found.");
            return;
        }

        Interpreter interpreter = new Interpreter();
        interpreter.eval(input);
    }

}
```



```java
package physicalc;

/**
 * @author Ici Li, il2117@columbia.edu
 */
public class NPrintFunction extends Function {

    public NPrintFunction() { ; }

    public Datum call(SymbolTable globals, SymbolTable locals, ExprList argument
s) {
        //System.out.println("Calling call() in NPrintFunction");

        for ( Expr expr : arguments.getContents() ) {
            System.out.print( expr.eval(globals, locals).toString() );
        }

        return null;
    }

}
```

```java
package physicalc;

/** "next" statement
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Next extends Stmt {

    public Next() {
        ;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        throw new NextSignal();
    }
}
```



```java
package physicalc;

/** Signal to continue next iteration of a loop.
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class NextSignal extends ControlSignal {

    public NextSignal() {
        ;
    }
}
```

```java
package physicalc;

/** The Node class is an abstract base class for all abstract program
 * representations in the Interpreter.
 *
 * Each Node sub-class represents a specific type of program
 * structure, such as an "if" statement or an addition expression.  A
 * tree of these Nodes is generated by the tree walker.
 *
 * Sub-classes must provide constructors with the appropriate argument
 * types to be used by the tree walker.  For example, a binary
 * operator node would have a constructor with two Expr arguments, one
 * for the left operand and one for the right.
 *
 * Each Node sub-class must provide an "eval" method.  "Eval" is
 * responsible for executing whatever logical part of the program is
 * represented by its node, recursively calling the "eval" methods of
 * its child nodes.  So, for example, an "if" node would "eval" its
 * conditional expression and use that result to decide which block to
 * "eval."
 *
 * "Eval" takes two SymbolTable arguments.  The first is the global
 * symbol table, which will remain constant throughout the program.
 * The second is the current local symbol table.  There is one local
 * symbol table for each function invocation, and one "top-level"
 * symbol table for statements executed outside any function body.
 * All nodes will pass these symbol tables unmodified to their child
 * nodes, except for function calls, which create a new local symbol
 * table.
 *
 * "Eval" returns a Datum object, which, if applicable, is the result
 * of evaluating this node's expression.  Statements and definitions
 * can simply return null.
 *
 * @see Program
 * @see Interpreter
 * @see SymbolTable
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public abstract class Node {

    public abstract Datum eval(SymbolTable globals, SymbolTable locals)
        throws InterpreterError;
}
```



```java
package physicalc;

/** Not is a node implementing the "not" logical operator.
 *
 * @see Node
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 * @author Changlong Jiang cj2214@columbia.edu
  */
public class Not extends Logical {
    private Expr oper;

    public Not(Expr Operand) {
        oper = Operand;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {

        if(oper.eval(globals, locals).isTrue()) {
                return new PBoolean(false);
        }
        else {
                return new PBoolean(true);
        }
    }
}
```

```java
package physicalc;

/** Op is an abstract base class for all operator nodes.
 *
 * @see Node
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public abstract class Op extends Expr {
}
```



```java
package physicalc;

/** Or is a node implementing the "or" logical operator.
 *
 * @see Node
 * @author Ici Li, il2117@columbia.edu
 */
public class Or extends Expr {
    private Expr left;
    private Expr right;

    public Or(Expr leftOperand, Expr rightOperand) {
        left = leftOperand;
        right = rightOperand;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        if ((left.eval(globals, locals).isTrue())) {
            /* short-circuit if left operand is true */
            return new PBoolean(true);
        } else if ((right.eval(globals, locals).isTrue())) {
            return new PBoolean(true);
        } else {
            return new PBoolean(false);
        }
    }
}
```

```java
package physicalc;

import java.lang.*;

/**
 * @author Brian Foo, bwf2101@columia.edu
 */
public class PBoolean extends Datum {

    protected Boolean boolValue;

    public PBoolean() {
                boolValue = new Boolean(false);
        }

    public PBoolean(boolean value) {
                boolValue = new Boolean(value);
    }

    public boolean isTrue() {
                return boolValue.booleanValue();
    }

    public String toString() {
                return boolValue.toString();
        }
}
```



```java
package physicalc;

import java.lang.*;
import java.util.*;

/** PList is the list data structure which
 * basically a front for java's ArrayList class
 * @author Brian Foo, bwf2101@columia.edu
 */
public class PList extends Datum  {

        protected ArrayList<Datum> list;

        public PList () {
                list = new ArrayList<Datum>();
        }

        public PList (int initCapacity) {
                list = new ArrayList<Datum>(initCapacity);
        }

        public Datum add(Datum that) {
                if (that instanceof PList) {
                        PList returnList = this;
                        PList thatList = (PList)that;
                        for (Iterator it = thatList.list.iterator (); it.hasNext
 (); ) {
                                returnList.push( (Datum) it.next() );
                        }
                        return returnList;
                } throw new TypeError("+", this, that);
        }

        public boolean push(Datum d) {
                return list.add(d);
        }

        public boolean contains(Datum d) {
                return list.contains(d);
        }

        public int indexOf(Datum d) {
                return list.indexOf(d);
        }

        public Datum getIndex(int index) throws TypeError, BoundsError {
                return list.get(index);
        }

        public Datum remove(int index) {
                return list.remove(index);
        }

        public Datum set(int index,Datum d) {
            //System.out.println("Calling set() in List");
            while (index >= list.size()) {
                //System.out.println("Enlarging a List");
                list.add(new PBoolean(false));
            }
            list.set(index,d);
            return d;
        }
```

```java
        public int size() {
                return list.size();
        }

    /** Returns true if "that" is the same type and has the same value
     * as this. */
    public boolean equals(Object that) {
                if (that instanceof PList) {
                        return equals((PList) that);
                }
            return false;
    }

    /** Returns true if "that" has the same value as this. */
        private boolean equals(PList that) {
                if ( list.size() != that.list.size() ) { return false; }
                Iterator thatIt = that.list.iterator();
                for (Iterator it = list.iterator (); it.hasNext (); ) {
                    Datum thisD = (Datum) it.next();
                    Datum thatD = (Datum) thatIt.next();
                    if ( !thisD.equals(thatD) ) { return false; }
                }
                return true;
    }

    /** Returns true if this object is "true" in the Physicalc sense.
     * Anything that is not the literal boolean "false" is considered
     * true in Physicalc. */
    public boolean isTrue() {
                return (!list.equals(false));
    }

    public void clear() {
                list.clear();
        }

    /** Returns a string representation of this Datum suitable for
     * display in program output. */
    public String toString() {
                String returnString = "{";
                for (Iterator it = list.iterator (); it.hasNext (); ) {
                        Datum d = (Datum) it.next();
                        returnString += d.toString()+",";
                }
                returnString += "}";
                return returnString;
    }

}
```



```java
package physicalc;

import java.lang.*;
import java.lang.Math;
import java.math.*;

/** PNumber is the number data class which includes
        intergers, decimals, and exponents.
        This is similar to the abstract Number class in java,
        but will convert all types into a Double object before
        algebraic processing
 @author Brian Foo, bwf2101@columia.edu
*/
public class PNumber extends Datum  {

        protected BigDecimal numValue;

        public PNumber () {
                numValue = new BigDecimal(0);
        }

        public PNumber (int number) {
                Integer n = new Integer(number);
                numValue = new BigDecimal(new BigInteger(n.toString()));
        }

        public PNumber (float number) {
                Float n = new Float(number);
                numValue = new BigDecimal(n.doubleValue());
        }

        public PNumber (double number) {
                numValue = new BigDecimal(number);
        }

        public PNumber (long number) {
                Long n = new Long(number);
                numValue = new BigDecimal(n.doubleValue());
        }

        public PNumber (short number) {
                Short n = new Short(number);
                numValue = new BigDecimal(n.doubleValue());
        }

        public PNumber (Integer number) {
                numValue = new BigDecimal(number.doubleValue());
        }

        public PNumber (Float number) {
                numValue = new BigDecimal(number.doubleValue());
        }

        public PNumber (Double number) {
                numValue = new BigDecimal(number.doubleValue());
        }

        public PNumber (Long number) {
                numValue = new BigDecimal(number.doubleValue());
        }

        public PNumber (Short number) {
```

```java
                numValue = new BigDecimal(number.doubleValue());
        }

        public PNumber (BigDecimal number) {
                numValue = number;
        }

        public PNumber (BigInteger number) {
                numValue = new BigDecimal(number);
        }

        public PNumber (String number) throws NumberFormatException {
                numValue = new BigDecimal(number);
                //numValue.setScale(7,java.math.BigDecimal.ROUND_HALF_EVEN);
        }

    /** Returns the result of this + that.  Does not modify this. */
    public Datum add(Datum that) throws TypeError {
                if (that instanceof PNumber) {
                        return new PNumber(numValue.add(((PNumber)that).numValue
));
                } throw new TypeError("+", this, that);
    }

    /** Returns the result of this − that.  Does not modify this. */
    public Datum sub(Datum that) throws TypeError {
                if (that instanceof PNumber) {
                        return new PNumber(numValue.subtract(((PNumber)that).num
Value));
                } throw new TypeError("−", this, that);
    }

    /** Returns the result of this * that.  Does not modify this. */
    /** Case: number*number returns number */
    /** Case: number*unit returns unit pair */
    /** Case: number*unitpair return unit pair */
    public Datum mul(Datum that) throws TypeError {
                if (that instanceof PNumber) {
                        return new PNumber(numValue.multiply(((PNumber)that).num
Value));
                } else if (that instanceof PUnit) {
                        return new PUnitPair(((PUnit)that).conversion.mul(this),
(PUnit)that);
                } else if (that instanceof PUnitPair) {
                        return new PUnitPair(this.mul(((PUnitPair)that).getNumbe
r()),((PUnitPair)that).getUnit());
                } else {
                        throw new TypeError("*", this, that);
                }
    }

    /** Returns the result of this / that.  Does not modify this. */
    /** Case: number/number return number */
    /** Case: number/unitpair return unit pair */
    public Datum div(Datum that) throws TypeError {
                if (that instanceof PNumber) {
                        return new PNumber((numValue.divide(((PNumber)that).numV
alue,20,java.math.BigDecimal.ROUND_HALF_EVEN)).toString());
                } else if (that instanceof PUnitPair) {
                        return new PUnitPair(this.div(((PUnitPair)that).getNumbe
r()),((PUnitPair)that).getUnit().neg());
                } else {
```



```java
                        throw new TypeError("/", this, that);
                }
    }

    /** Returns the result of this ^ that.  Does not modify this. */
    public Datum pow(Datum that) throws TypeError {
                if (that instanceof PNumber) {
                        return new PNumber(java.lang.Math.pow(numValue.doubleVal
ue() ,((PNumber)that).numValue.doubleValue()));
                } throw new TypeError("^", this, that);
    }

    /** Returns the result of the unary minus operator, (− this).
     * Does not modify this. */
    public Datum neg() throws TypeError {
                return new PNumber(numValue.negate());
    }

    /** Returns true if "that" is the same type and has the same value
     * as this. */
    public boolean equals(Object that) {
                if (that instanceof PNumber) {
                        return equals((PNumber) that);
                }
            return false;
    }

    private boolean equals(PNumber that) {
                return numValue.compareTo(that.numValue) == 0;
        }

    /** Returns true if this object is "true" in the Physicalc sense.
     * Anything that is not the literal boolean "false" is considered
     * true in Physicalc. */
    public boolean isTrue() {
                return !numValue.equals(false);
    }

    public PBoolean lessThan(Datum that) throws TypeError {
                if (that instanceof PNumber) {
                        return new PBoolean( numValue.compareTo(((PNumber)that).
numValue) < 0 );
                } throw new TypeError("<", this, that);
    }

    public PBoolean lessEqual(Datum that) throws TypeError {
                if (that instanceof PNumber) {
                        return new PBoolean( numValue.compareTo(((PNumber)that).
numValue) <= 0 );
                } throw new TypeError("<=", this, that);
    }

    public PBoolean greaterThan(Datum that) throws TypeError {
                if (that instanceof PNumber) {
                        return new PBoolean( numValue.compareTo(((PNumber)that).
numValue) > 0 );
                } throw new TypeError(">", this, that);
    }

    public PBoolean greaterEqual(Datum that) throws TypeError {
                if (that instanceof PNumber) {
                        return new PBoolean( numValue.compareTo(((PNumber)that).
```

```java
numValue) >= 0 );
                } throw new TypeError(">=", this, that);
    }

    /** Returns a string representation of this Datum suitable for
     * display in program output. */
    public String toString() {
                Double d = new Double(numValue.toString());
                return d.toString();
    }

    /** Returns this number as an int. */
    public int toInt() {
        return numValue.intValue();
    }

}
```



```java
package physicalc;

import java.lang.*;

/** PString is the string data class which is
        basically a front for java's String class
* @author Brian Foo, bwf2101@columia.edu
*/
public class PString extends Datum  {

        protected String sValue;

        public PString () {
                sValue = "";
        }

        public PString (String s) throws NumberFormatException {
                sValue = s;
        }

    /** Returns the result of this + that.  Does not modify this. */
    public Datum add(Datum that) throws TypeError {
                if (that instanceof PString) {
                        return new PString(sValue+((PString)that).sValue);
                } throw new TypeError("+", this, that);
    }

    /** Returns true if "that" is the same type and has the same value
     * as this. */
    public boolean equals(Object that) {
                if (that instanceof PString) {
                        return equals((PString) that);
                }
            return false;
    }

    /** Returns true if "that" has the same value as this. */
        private boolean equals(PString that) {
                return sValue.compareTo(that.sValue) == 0;
    }

    /** Returns true if this object is "true" in the Physicalc sense.
     * Anything that is not the literal boolean "false" is considered
     * true in Physicalc. */
    public boolean isTrue() {
                return (!sValue.equals(false));
    }

    public PBoolean lessThan(Datum that) throws TypeError {
                if (that instanceof PString) {
                        return new PBoolean( sValue.compareTo(((PString) that).s
Value) < 0 );
                } throw new TypeError("<", this, that);
    }

    public PBoolean lessEqual(Datum that) throws TypeError {
                if (that instanceof PString) {
                        return new PBoolean( sValue.compareTo(((PString) that).s
Value) <= 0 );
                } throw new TypeError("<=", this, that);
    }
```

```
    public PBoolean greaterThan(Datum that) throws TypeError {
                if (that instanceof PString) {
                        return new PBoolean( sValue.compareTo(((PString) that).s
Value) > 0 );
                } throw new TypeError(">", this, that);
    }

    public PBoolean greaterEqual(Datum that) throws TypeError {
                if (that instanceof PString) {
                        return new PBoolean( sValue.compareTo(((PString) that).s
Value) >= 0 );
                } throw new TypeError(">=", this, that);
    }

    /** Returns a string representation of this Datum suitable for
     * display in program output. */
    public String toString() {
                return sValue.toString();
    }

}
```



```
package physicalc;

import java.lang.*;
import java.lang.Math;
import java.util.*;

/** PUnit is the symbolic data class which includes
        the algebraic manipulation of symbolic units
        stored as HashMaps
*
* @author Brian Foo, bwf2101@columia.edu */
public class PUnit extends Datum  {

        protected String name;                                          // ie. m
inute
        protected String rootName;                              // ie. second
        protected PNumber conversion;                           // ie. 60
        protected HashMap<String,PNumber> unitMap;
        protected boolean unitMode;

        public PUnit () {
                name = "";
                conversion = new PNumber("1");
                unitMap = new HashMap<String,PNumber>();
                unitMode = false;
        }

        public PUnit (HashMap<String,PNumber> map, PNumber conv) {
                name = "";
                rootName = "";
                conversion = conv;
                unitMap = map;
                unitMode = false;
        }

        public PUnit (String s) {
                name = s;
                rootName = s;
                conversion = new PNumber("1");
                unitMap = new HashMap<String,PNumber>();
                unitMap.put(s,new PNumber("1"));
                unitMode = false;
        }

        public PUnit (String s, Datum p) throws TypeError {
                name = s;
                if ( p instanceof PUnitPair ) {
                        conversion = ((PUnitPair) p).getNumber();
                        unitMap = ((PUnitPair) p).getUnit().unitMap;
                        rootName = ((PUnitPair) p).getUnit().rootName;
                        unitMode = false;
                } else { throw new TypeError(p, "PUnitPair", this); }
        }

        /** lets you know if the two units can be added */
        public Datum add(Datum that) throws TypeError {
                if ( that instanceof PUnit ) {
                        return new PBoolean(equals(((PUnit)that)));
                } throw new TypeError("+", this, that);
        }

        /** lets you know if the two units can be subtracted */
```

```
        public Datum sub(Datum that) throws TypeError {
                if ( that instanceof PUnit ) {
                        return new PBoolean(equals(((PUnit)that)));
                } throw new TypeError("-", this, that);
    }

    /** Returns the result of this * that.  Does not modify this. */
    /** Case: number*unit returns unit pair */
    public Datum mul(Datum that) throws TypeError {
                if (that instanceof PUnit) {
                        HashMap<String,PNumber> returnMap = new HashMap<String,P
Number>();
                        Iterator it = unitMap.keySet().iterator();
                        String thisKey = "";
                        PNumber thisValue = new PNumber();
                        PNumber thatValue = new PNumber();
                        while(it.hasNext()) { // go through this unit hashmap
                                thisKey = (String)it.next();
                                thisValue = new PNumber( unitMap.get(thisKey).to
String() );
                                if (((PUnit)that).unitMap.containsKey(thisKey))
{
                                        thatValue = new PNumber( ((PUnit)that).u
nitMap.get(thisKey).toString() );
                                        returnMap.put( thisKey,(PNumber)thisValu
e.add(thatValue) );
                                }
                                else {
                                        returnMap.put( thisKey,thisValue );
                                }
                        }
                        it = ((PUnit)that).unitMap.keySet().iterator();
                        String thatKey = "";
                        while(it.hasNext()) { // go through that unit hashmap
                                thatKey = (String)it.next();
                                thatValue = new PNumber( ((PUnit)that).unitMap.g
et(thatKey).toString() );
                                if (!unitMap.containsKey(thatKey)) {
                                        returnMap.put( thatKey,thatValue );
                                }
                        }
                        return new PUnit(returnMap,(PNumber) conversion.mul(((PU
nit)that).conversion));
                } else if (that instanceof PNumber) {
                        PUnit returnUnit = this;
                        return new PUnitPair(conversion.mul((PNumber)that),retur
nUnit);
                } else {
                        throw new TypeError("*", this, that);
                }
    }

    /** Returns the result of this / that.  Does not modify this. */
        public Datum div(Datum that) throws TypeError {
                if (that instanceof PUnit) {
                        HashMap<String,PNumber> returnMap = new HashMap<String,P
Number>();
                        Iterator it = unitMap.keySet().iterator();
                        String thisKey = "";
                        PNumber thisValue = new PNumber();
                        PNumber thatValue = new PNumber();
                        while(it.hasNext()) { // go through this unit hashmap
```



```
                                thisKey = (String)it.next();
                                thisValue = new PNumber( unitMap.get(thisKey).to
String() );
                                if (((PUnit)that).unitMap.containsKey(thisKey))
{
                                        thatValue = new PNumber( ((PUnit)that).u
nitMap.get(thisKey).toString() );
                                        //System.err.println("putting in "+thisK
ey);
                                        returnMap.put( thisKey,(PNumber) thisVal
ue.sub(thatValue) );
                                }
                                else {
                                        //System.err.println("putting in "+thisK
ey);
                                        returnMap.put( thisKey,thisValue );
                                }
                        }
                        it = ((PUnit)that).unitMap.keySet().iterator();
                        String thatKey = "";
                        while(it.hasNext()) { // go through that unit hashmap
                                thatKey = (String)it.next();
                                thatValue = new PNumber( ((PUnit)that).unitMap.g
et(thatKey).toString() );
                                if (!unitMap.containsKey(thatKey)) {
                                        //System.err.println("putting in "+thatK
ey);
                                        returnMap.put( thatKey,(PNumber) thatVal
ue.neg() );
                                }
                        }
                        return new PUnit(returnMap,(PNumber) conversion.div(((PU
nit)that).conversion));
                } else {
                        throw new TypeError("/", this, that);
                }
        }

        public Datum pow(Datum n) {
                if (n instanceof PNumber) {
                        HashMap<String,PNumber> returnMap = new HashMap<String,P
Number>();
                        Iterator it = unitMap.keySet().iterator();
                        String thisKey = "";
                        PNumber thisValue = new PNumber();
                        while(it.hasNext()) { // go through this unit hashmap
                                thisKey = (String)it.next();
                                thisValue = new PNumber( unitMap.get(thisKey).to
String() );
                                returnMap.put( thisKey,(PNumber)thisValue.mul(n)
 );
                        }
                        return new PUnit(returnMap,(PNumber)conversion.pow((PNum
ber)n));
                } else {
                        throw new TypeError("^", this, n);
                }
        }

        public Datum neg() {
                HashMap<String,PNumber> returnMap = new HashMap<String,PNumber>(
);
```

```
                Iterator it = unitMap.keySet().iterator();
                String thisKey = "";
                PNumber thisValue = new PNumber();
                while(it.hasNext()) { // go through this unit hashmap
                        thisKey = (String)it.next();
                        thisValue = new PNumber( unitMap.get(thisKey).toString()
 );
                        returnMap.put( thisKey,(PNumber)thisValue.neg() );
                }
                return new PUnit(returnMap,(PNumber)(new PNumber("1")).div(conve
rsion));
        }

        /** Returns true if "that" is the same type and has the same value
         * as this. */
        public boolean equals(Object that) {
                if (that instanceof PUnit) {
                        return equals((PUnit) that);
                }
                return false;
        }

        /** Returns true if "that" has the same value as this. */
        private boolean equals(PUnit that) {
                if ( unitMap.size() != that.unitMap.size() ) { return false; }
                Iterator it = unitMap.keySet().iterator();
                String thisKey = "";
                PNumber thisValue = new PNumber();
                PNumber thatValue = new PNumber();
                while(it.hasNext()) { // go through this unit hashmap
                        thisKey = (String)it.next();
                        thisValue = new PNumber( unitMap.get(thisKey).toString()
 );
                        if (that.unitMap.containsKey(thisKey)) {
                                thatValue = new PNumber( that.unitMap.get(thisKe
y).toString() );
                                if ( !thisValue.equals(thatValue) ){ return fals
e; }
                        }
                        else {
                                return false;
                        }
                }
                return true;
        }

        /** Returns true if this object is "true" in the Physicalc sense.
         * Anything that is not the literal boolean "false" is considered
         * true in Physicalc. */
        public boolean isTrue() {
                return !unitMap.isEmpty();
    }

    public PNumber getConversion() {
                return conversion;
        }

        public String getName() {
                return name;
        }

        public String getBaseUnit() {
```



```
                return rootName;
        }
        public void setUnitMode() {
                unitMode = true;
        }
        public void unsetUnitMode() {
                unitMode = false;
        }


        public String toUnit() {
                Iterator it = unitMap.keySet().iterator();
                String returnString = "";
                String negString = "";
                String thisKey = "";
                int negCount = 0;
                boolean first = true;
                boolean firstneg = true;

                while(it.hasNext()) { // go through this unit hashmap
                        thisKey = (String)it.next();
                        PNumber value = unitMap.get(thisKey);

                        if ( ( value.greaterThan(new PNumber("1")) ).isTrue() )
{
                                if (!first) { returnString += "*"; } else { firs
t = false; }
                                if (!(thisKey.compareTo("") == 0)) {
                                returnString += thisKey + "^" + value.toString()
; }
                        }
                        else if (value.equals(new PNumber("1")) ) {
                                if (!first) { returnString += "*"; } else { firs
t = false; }
                                if (!(thisKey.compareTo("") == 0)) {
                                returnString += thisKey; }
                        }
                        else if ( ( value.lessThan(new PNumber("-1")) ).isTrue()
 ) {
                                if (!firstneg) { negString += "*"; } else { firs
tneg = false; }
                                if (!(thisKey.compareTo("") == 0)) {
                                negString += thisKey + "^" + value.neg().toStrin
g(); }
                                negCount++;
                        }
                        else if (value.equals(new PNumber("-1")) ) {
                                if (!firstneg) { negString += "*"; } else { firs
tneg = false; }
                                if (!(thisKey.compareTo("") == 0)) {
                                negString += thisKey;}
                                negCount++;
                        }
                        else if ( ( value.greaterThan(new PNumber("0")) ).isTrue
() ) {
                                if (!first) { returnString += "*"; } else { firs
t = false; }
                                if (!(thisKey.compareTo("") == 0)) {
                                returnString += thisKey + "^" + value.toString()
; }
                        }
                        else if ( ( value.lessThan(new PNumber("0")) ).isTrue()
```

```java
) {
                                if (!firstneg) { negString += "*"; } else { firs
tneg = false; }
                                if (!(thisKey.compareTo("") == 0)) {
                                negString += thisKey + "^" + value.neg().toStrin
g(); }
                                negCount++;
                        }
                        else {
                                returnString += "";
                        }
                }

                // now put the positive and negative together
                if ( returnString.equals("") ) { returnString = "1"; }
                if ( negCount > 1 ) {
                        if ( returnString.equals("") ) { returnString = "("+negS
tring+")^-1"; }
                        else { returnString += "*("+negString+")^-1"; }
                } else {
                        if ( negCount > 0 ){
                                if ( returnString.equals("") ) { returnString =
negString+"^-1"; }
                                else { returnString += "*"+negString+"^-1"; }
                        }
                }

                return returnString;
        }

    /** Returns a string representation of this Datum suitable for
         * display in program output. */
        public String toString() {
                if ( unitMode )
                        return this.toUnit();
                else
                        return conversion.toString()+"*"+this.toUnit();
    }

}
```



```java
package physicalc;

import java.lang.*;
import java.lang.Math;

/** PUnitPair is the number data class which includes
        intergers, decimals, and exponents.
        This is similar to the abstract Number class in java,
        but will convert all types into a Double object before
        algebraic processing
* @author Brian Foo, bwf2101@columia.edu */
public class PUnitPair extends Datum  {

        protected PNumber number;
        protected PUnit unit;
        protected boolean forceName;

        public PUnitPair () {
                number = new PNumber(0);
                unit = new PUnit();
        }

        public PUnitPair (Datum n,Datum u) throws TypeError {
                if ( n instanceof PNumber ) {
                        number = (PNumber) n;
                        forceName = false;
                } else { throw new TypeError(n, "PNumber", this); }
                if ( u instanceof PUnit ) {
                        unit = (PUnit) u;
                        forceName = false;
                } else { throw new TypeError(u, "PUnit", this); }
        }

        public PUnitPair (Datum n,Datum u,boolean _forceName) throws TypeError {
                if ( n instanceof PNumber ) {
                        number = (PNumber) n;
                        forceName = _forceName;
                } else { throw new TypeError(n, "PNumber", this); }
                if ( u instanceof PUnit ) {
                        unit = (PUnit) u;
                        forceName = _forceName;
                } else { throw new TypeError(u, "PUnit", this); }

        }

    /** Returns the result of this + that.  Does not modify this. */
    public Datum add(Datum that) throws TypeError {
                if (that instanceof PUnitPair) {
                        if ( (unit.add(((PUnitPair)that).unit)).isTrue() ) {
                                return new PUnitPair(number.add(((PUnitPair)that
).number),unit);
                        }
                        throw new TypeError("+", this, that);
                }
                throw new TypeError("+", this, that);
    }

    /** Returns the result of this + that.  Does not modify this. */
        public Datum sub(Datum that) throws TypeError {
                if (that instanceof PUnitPair) {
                        if ( (unit.sub(((PUnitPair)that).unit)).isTrue() ) {
                                return new PUnitPair(number.sub(((PUnitPair)that
```

```
).number),unit);
                        }
                        throw new TypeError("-", this, that);
                }
                throw new TypeError("-", this, that);
    }

    /** Returns the result of this * that.  Does not modify this. */
    /** Case: unitpair*unitpair return unit pair */
    /** Case: number*unitpair return unit pair */
    public Datum mul(Datum that) throws TypeError {
                if (that instanceof PUnitPair) {
                        return new PUnitPair(number.mul(((PUnitPair)that).number
),unit.mul(((PUnitPair)that).unit));
                } else if (that instanceof PNumber) {
                        return new PUnitPair(number.mul(((PNumber)that)),unit);
                } else {
                        throw new TypeError("*", this, that);
                }
    }

    /** Returns the result of this / that.  Does not modify this. */
        /** Case: unitpair/unitpair return unit pair */
        /** Case: number/unitpair return unit pair */
        public Datum div(Datum that) throws TypeError {
                if (that instanceof PUnitPair) {
                        return new PUnitPair(number.div(((PUnitPair)that).number
),unit.div(((PUnitPair)that).unit));
                } else if (that instanceof PNumber) {
                        return new PUnitPair(number.div(((PNumber)that)),unit);
                } else {
                        throw new TypeError("/", this, that);
                }
    }

    /** Returns the result of this ^ n.  Does not modify this. */
    public Datum pow(Datum n) throws TypeError {
                if ( n instanceof PNumber ) {
                        return new PUnitPair(number.pow((PNumber)n),unit.pow((PN
umber)n));
                } throw new TypeError("^", this, n);
    }

    /** Returns the result of the unary minus operator, (- this).
     * Does not modify this. */
    public Datum neg() throws TypeError {
                return new PUnitPair(number.neg(),unit);
    }

    /** Returns true if "that" is the same type and has the same value
     * as this. */
    public boolean equals(Object that) {
                if (that instanceof PUnitPair) {
                        return equals((PUnitPair) that);
                }
            return false;
    }

    /** Returns true if "that" has the same value as this. */
        private boolean equals(PUnitPair that) {
                return number.equals(that.number) && unit.equals(that.unit);
    }
```



```
    /** Returns true if this object is "true" in the Physicalc sense.
     * Anything that is not the literal boolean "false" is considered
     * true in Physicalc. */
    public boolean isTrue() {
                return number.isTrue() && unit.isTrue();
    }

    public void setNumber(PNumber n) {
                number = n;
        }

        public void setUnit(PUnit u) {
                unit = u;
        }

    public PNumber getNumber() {
                return number;
        }

        public PUnit getUnit() {
                return unit;
        }

    /** Returns a string representation of this Datum suitable for
     * display in program output. */
    public String toString() {
                if ( !forceName )
                        return number.toString()+"*"+unit.toUnit();
                else
                        return number.toString()+"*"+unit.getName();
    }

}
```

```java
package physicalc;

import java.lang.String;
import java.util.ArrayList;
import java.util.List;

/** A ParamList is a container for a list of function parameters.  It
 * is used in function definitions.
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class ParamList extends Expr {

    private ArrayList<String> contents;

    public ParamList() {
        //System.out.println("Constructing a ParamList");
        contents = new ArrayList<String>();
    }

    public void insert(String i) {
        //System.out.println("Adding to a ParamList");
        contents.add(i);
    }

    public List<String> getContents() {
        return contents;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        /* This shouldn't be called. */
        throw new InterpreterError("GHASTLY ERROR: Called 'eval' method on a ParamList.");
    }
}
```



```java
package physicalc;

/**
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class PrintFunction extends Function {

    public PrintFunction() { ; }

    public Datum call(SymbolTable globals, SymbolTable locals, ExprList argument
s) {
        //System.out.println("Calling call() in PrintFunction");

        for ( Expr expr : arguments.getContents() ) {
            System.out.print( expr.eval(globals, locals).toString() );
        }
        System.out.println();
        return null;
    }

}
```

```java
package physicalc;

import java.util.ArrayList;
import java.util.List;

/** A Program is a container for a collection of Nodes representing a
 * complete program.
 *
 * Evaluating a Program evaluates all its sub−nodes in order, and
 * returns the value of the last node.
 *
 * A Program creates its own top−level symbol table for variables
 * defined outisde of any function definitions.
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Program extends Node {

    private ArrayList<Node> contents;

    public Program() {
        //System.out.println("Constructing a Program");
        contents = new ArrayList<Node>();
    }

    public void insert(Node n) {
        //System.out.println("Adding to a Program");
        contents.add(n);
    }

    public List<Node> getContents() {
        return contents;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        //System.out.println("Calling eval() in Program");

        locals = new SymbolTable();

        Datum result = null;
        for (Node n : contents) {
            result = n.eval(globals, locals);
        }
        return result;
    }
}
```



```java
package physicalc;

import java.lang.String;

/** Rel is a node implementing any relational operator, including
 * equals and not−equals.
 *
 * @see Node
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Rel extends Logical {
    private Expr left;
    private Expr right;
    private String op;

    public Rel(String operator, Expr leftOperand, Expr rightOperand) {
        op = operator;
        left = leftOperand;
        right = rightOperand;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        Datum leftValue = left.eval(globals, locals);
        Datum rightValue = right.eval(globals, locals);

        /* Datum classes take care of type checking. */
        if (op.equals("=")) {
            /* equals() returns a Java boolean; we must create a
             * PBoolean to match our return type. */
            return new PBoolean(leftValue.equals(rightValue));
        } else if (op.equals("!=")) {
            /* Same thing, but take the logical opposite. */
            return new PBoolean(!(leftValue.equals(rightValue)));
        } else if (op.equals("<")) {
            return leftValue.lessThan(rightValue);
        } else if (op.equals("<=")) {
            return leftValue.lessEqual(rightValue);
        } else if (op.equals(">")) {
            return leftValue.greaterThan(rightValue);
        } else if (op.equals(">=")) {
            return leftValue.greaterEqual(rightValue);
        } else {
            /* This will only happen if the tree walker is wrong. */
            throw new InterpreterError("GHASTLY ERROR: Rel class with invalid operator.");
        }
    }
}
```

```java
package physicalc;

/**
 * @author Ici Li, il2117@columbia.edu
 */
public class Return extends Stmt {

        private Expr returnVal;

    public Return(Expr rv) {
        returnVal = rv;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        throw new ReturnSignal(returnVal.eval(globals,locals));
    }
}
```



```java
package physicalc;

/** ReturnSignal is used to signal to a function call that a "return"
 * statement has been executed.  It carries the value to be returned
 * from the function.
 *
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class ReturnSignal extends ControlSignal {

    private Datum value;

    public ReturnSignal(Datum returnValue) {
        value = returnValue;
    }

    public Datum getValue() {
        return value;
    }
}
```

```java
package physicalc;

/** A RuntimeObject is anything that can be bound to a symbol in the
 * SymbolTable, i.e. a Function, Variable, Constant, or Unit.
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public interface RuntimeObject {
}
```



```java
package physicalc;

import java.lang.String;

/**
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class Set extends Stmt {

    LValue lvalue;
    Expr valueExpr;

    public Set(LValue place, Expr valueExpression) {
        lvalue = place;
        valueExpr = valueExpression;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        lvalue.setValue(globals, locals, valueExpr.eval(globals,locals));
        return null;
    }
}
```

```java
package physicalc;

/** Stmt is an abstract base class for all statement nodes.
 *
 * @see Node
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public abstract class Stmt extends Node {
}
```



```java
package physicalc;

import java.util.HashMap;

/** A SymbolTable associates symbols (strings) with run-time objects
 * (functions, variables, units, or constants).
 *
 * Physicalc's symbol tables do not have a parent node, because
 * Physicalc has no nested scopes.  At any time there are exactly two
 * scopes in effect: a global scope for definitions and a local scope
 * for variables and function arguments.
 *
 * SymbolTable stores any object that implements the RuntimeObject
 * interface.
 *
 * @see RuntimeObject
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class SymbolTable {

    private HashMap<String, RuntimeObject> table;

    /** Creates a new, empty symbol table. */
    public SymbolTable() {
        table = new HashMap<String, RuntimeObject>();
    }

    /** Associates an object with a symbol in the symbol table.  If
     * "symbol" is already in the table, its value will be
     * overwritten.
     */
    public void put(String symbol, RuntimeObject object) {
        table.put(symbol, object);
    }

    /** Associates "newSymbol" with the value of "oldSymbol".  If
     * "oldSymbol" is not defined, throws an UndefinedError.
     *
     * Aliases are not references.  If "oldSymbol" is redefined to
     * point to a new object, the alias continues to point to the old
     * object.
     */
    public void putAlias(String newSymbol, String oldSymbol) {
        if (table.containsKey(oldSymbol)) {
            table.put(newSymbol, table.get(oldSymbol));
        } else {
            throw new UndefinedError(oldSymbol);
        }
    }

    /** Looks up and returns the value of "symbol" in the table.
     * Returns null if this table does not contain "symbol".
     *
     * Returns a generic RuntimeObject reference.  Callers of this
     * method must check the type of the returned object and cast it
     * appropriately.
     */
    public RuntimeObject get(String symbol) {
        return table.get(symbol);
    }
}
```

```java
package physicalc;

/**
 * @author Brian Foo, bwf2101@columia.edu
 */
public class ToIntFunction extends Function {

    public ToIntFunction() { ; }

    public Datum call(SymbolTable globals, SymbolTable locals, ExprList argument
s) {

        if (arguments.getContents().size() != 1) {
            throw new InterpreterError("Cannot call toInt on more than one argument");
        }


        Expr expr = arguments.getContents().get(0);

        Datum number = expr.eval(globals,locals);

        if (number instanceof PNumber) {
            return new PNumber( ((PNumber)number).toInt() );
        } else {
            throw new InterpreterError("Cannot call ToInt function on non−number");
        }
    }

}
```



```java
package physicalc;

/**
 * @author Ici Li, il2117@columbia.edu
 */
public class ToStringFunction extends Function {

    public ToStringFunction() { ; }

    public Datum call(SymbolTable globals, SymbolTable locals, ExprList argument
s) {

        if (arguments.getContents().size() != 1) {
            throw new InterpreterError("Cannot call toInt on more than one argument");
        }


        Expr expr = arguments.getContents().get(0);

                Datum string1 = expr.eval(globals,locals);

                return new PString(string1.toString());
    }

}
```

```java
package physicalc;

import java.lang.*;

/** TypeError is raised when an operation cannot be performed because
 * the types of its arguments are incompatible.
 *
 * @author Brian Foo, bwf2101@columia.edu
 */
public class TypeError extends InterpreterError {

    String errorMessage;

    public TypeError(String operation, Datum object1, Datum object2) {
                errorMessage = "Types are incompatible in the following operation:\n" + object1.
toString() + " " + operation + " " + object2.toString() + "\n";
    }

    /* wrong types for instantiation */
    public TypeError(Datum found, String requires, Datum object) {
                        errorMessage = "Cannot instantiate " + object.toString() + " wi
th " + found.toString() + ", requires " + requires + "\n";
    }

    public String toString() {
                return errorMessage;
        }
}
```



```java
package physicalc;

/** Unary is a node implementing the unary minus operator.  It returns
 * the numeric negative of its argument.
 *
 * @see Node
 * @author Ici Li, il2117@columbia.edu
 */
public class Unary extends Op {
    private Expr op;

    public Unary(Expr operand) {
        op = operand;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
        Datum oper = op.eval(globals, locals);

        return oper.neg();
    }
}
```

```java
package physicalc;

/** Error when an undefined symbol is used.
 * @author Stuart Sierra, ss2806@columbia.edu
 */
public class UndefinedError extends InterpreterError {
    private String symbol;

    public UndefinedError(String undefinedSymbol) {
        symbol = undefinedSymbol;
    }

    public String toString() {
        return "’" + symbol + "’ is not defined";
    }
}
```



```java
package physicalc;


/** A Unit stores a value locally.  Unit are created and
 * modified with the "set" statement.  The name of the Unit is
 * stored in the SymbolTable.  The value of a Unit can be changed.
 *
 * @see SymbolTable
 * @see Set
 * @author Brian Foo, bwf2101@columia.edu
 */
public class Unit implements RuntimeObject {

        PUnit unit;

    public Unit() {
                unit = new PUnit();
    }

    public Unit(String id) {
                //System.out.println("Creating Base Unit");
                unit = new PUnit(id);
        }

    public Unit(String id,Datum initialValue) {
                //System.out.println("Creating Derived Unit");
                if ( initialValue instanceof PUnit ) {
                        unit = (PUnit) initialValue;
                } else if (initialValue instanceof PUnitPair ) {
                        unit = new PUnit(id,(PUnitPair)initialValue);
                } else {
                        throw new InterpreterError("Unit initialized with illegal expression")
;
                }
    }

    public Datum getValue() {
                return unit;
    }

    public Datum setValue(String id,Datum newValue) {
                if ( newValue instanceof PUnitPair ) {
                        unit = new PUnit(id,(PUnitPair)newValue);
                        return unit;
                } else if (newValue instanceof PUnit) {
                        unit = (PUnit) newValue;
                        return unit;
                } else {
                        throw new InterpreterError("Unit set with non−unit object");
                }
    }

    public Datum setValue(String id) {
                unit = new PUnit(id);
                return unit;
        }
}
```

```java
package physicalc;

/**
 * @author Brian Foo, bwf2101@columia.edu
 */
public class UnitDef extends Def {

        String id;
        Expr valueExpr;

    public UnitDef(String i) {
                id = i;
                valueExpr = null;
    }

    public UnitDef(String i, Expr val) {
        id = i;
        valueExpr = val;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
                //System.out.println("Calling eval() in UnitDef");
                if (valueExpr == null) {
                        //System.out.println("Found Unit definition with null ex
pression");
                        globals.put(id,new Unit(id));
                } else {
                        //System.out.println("Found Unit definition with express
ion");
                        Datum val = valueExpr.eval(globals,locals);
                        globals.put(id,new Unit(id,val));
                }

        return null; // definitions always return null
    }
}
```



```java
package physicalc;


/** A Variable stores a value locally.  Variable are created and
 * modified with the "set" statement.  The name of the Variable is
 * stored in the SymbolTable.  The value of a Variable can be changed.
 *
 * @see SymbolTable
 * @see Set
 * @author Ici Li, il2117@columbia.edu
 */
public class Variable implements RuntimeObject {

    Datum var;

    public Variable() {
        var = null;
    }

    public Variable(Datum initialValue) {
        var = initialValue;
    }

    public Datum getValue() {
        return var;
    }

    public Datum setValue(Datum newValue) {
        var = newValue;
        return var;
    }
}
```

```java
package physicalc;

import java.lang.*;

/** While Statement
 *  While expr1 do
 *        block1
 *  done
 *
 * @author Changlong Jiang cj2214@columbia.edu
 * @author Stuart Sierra, ss2806@columbia.edu
 */

public class While extends Stmt {

        private Expr expr1;
        private Block block1;

        public While() { expr1 = null; block1 =null;}

    public While(Expr testExpr, Block block) {
        expr1 = testExpr;
                block1 = block;
    }

    public Datum eval(SymbolTable globals, SymbolTable locals) {
      while (expr1.eval(globals,locals).isTrue()) {
                try {
                    block1.eval(globals,locals);
                }
                catch (BreakSignal breaksignal)
                {
                        break;
                }
                catch (NextSignal nextsignal)
                {
                        continue;
                }

      }
        return null;
    }
}
```



```java
package physicalc;

import java.lang.String;
import java.io.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Ignore;
import org.junit.Test;
import static org.junit.Assert.*;
import junit.framework.JUnit4TestAdapter;

public class InterpreterTest {

    /** An interpreter instance used by each test. */
    private Interpreter interpreter;

    @Before public void setupInterpreter() {
        interpreter = new Interpreter();
    }

    @Test public void alwaysPasses() {
        assertEquals("This test always passes.", true, true);
    }

    @Test public void doNothing() {
        assertPrints("Should do nothing.", "", "");
    }

    @Test public void arithmetic() {
        assertPrints("print(2 + 3)\n", "5\n");
        assertPrints("print(1.1 + 2.2)\n", "3.3\n");
        assertPrints("print(9 − 5)\n", "4\n");
        assertPrints("print(3 * 4)\n", "12\n");
        assertPrints("print(2 ^ 8)\n", "256\n");

        /* Need Unary class for the following: */
        assertPrints("print(−3)\n", "−3\n");
        assertPrints("print(4 ^ −2)\n", "0.0625\n");
    }

    @Test public void strings() {
        assertPrints("print(\"foo\" + \"bar\")\n", "foobar\n");
    }

    @Test public void ifStmt() {
        assertPrints("if true then\n print(\"yes\")\n done\n",
                     "yes\n");
        assertPrints("if false then\n print(\"yes\")\n done\n",
                     "");
        assertPrints("if true then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "yes\n");
        assertPrints("if false then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "no\n");
        assertPrints("if true then \n print(\"a\") \n elsif true then \n print(\"b\") \n elsif true then \n print(\"c\"
) \n elsif true then \n print(\"d\") \n else \n print(\"e\") \n done \n ", "a\n");
        assertPrints("if false then \n print(\"a\") \n elsif true then \n print(\"b\") \n elsif true then \n print(\"c\
") \n elsif true then \n print(\"d\") \n else \n print(\"e\") \n done \n ", "b\n");
        assertPrints("if false then \n print(\"a\") \n elsif false then \n print(\"b\") \n elsif true then \n print(\"c
\") \n elsif true then \n print(\"d\") \n else \n print(\"e\") \n done \n ", "c\n");
        assertPrints("if false then \n print(\"a\") \n elsif false then \n print(\"b\") \n elsif false then \n print(\"
c\") \n elsif true then \n print(\"d\") \n else \n print(\"e\") \n done \n ", "d\n");
        assertPrints("if false then \n print(\"a\") \n elsif false then \n print(\"b\") \n elsif false then \n print(\"
c\") \n elsif false then \n print(\"d\") \n else \n print(\"e\") \n done \n ", "e\n");
    }
```

```
    @Test public void relational() {
        assertPrints("if 3 < 4 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "yes\n");
        assertPrints("if 4 < 3 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "no\n");
        assertPrints("if 3 <= 4 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "yes\n");
        assertPrints("if 4 <= 3 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "no\n");
        assertPrints("if 3 > 4 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "no\n");
        assertPrints("if 4 > 3 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "yes\n");
        assertPrints("if 3 >= 4 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "no\n");
        assertPrints("if 4 >= 3 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "yes\n");
        assertPrints("if 3 = 3 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "yes\n");
        assertPrints("if 3 != 3 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "no\n");
        assertPrints("if 4 = 3 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "no\n");
        assertPrints("if 4 != 3 then\n print(\"yes\")\n else\n print(\"no\")\n done\n ",   "yes\n");
    }

    @Test public void helloWorld() {
        assertPrints("print(\"Hello, world!\")\n",
                     "Hello, world!\n");
    }

    @Test public void testWhile() {
        assertPrints("while true do \n print(\"a\") \n break \n print(\"b\") \n done \n",
                     "a\n");
        assertPrints("while false do \n print(\"a\") \n break \n print(\"b\") \n done \n",
                     "");
    }



    /** The suite() method is required for compatibility with older
     * JUnit versions. */
    public static junit.framework.Test suite() {
        return new JUnit4TestAdapter(InterpreterTest.class);
    }

    /** The assertPrints() method is an assertion that runs the
     * interpreter and checks that it prints out a certain string.
     *
     * @param message A string explaining what the test does.
     * @param program A string of Physicalc source code.  Remember the
     *                terminating line break or semicolon!
     * @param expected A string of what the interpreter should print.
     */
    private void assertPrints(String message,
                              String program,
                              String expected) {
        StringReader code = new StringReader(program);
        OutputStream output = new ByteArrayOutputStream();
        interpreter.setOutputStream(output);
        interpreter.eval(code);
        assertEquals(message, expected, output.toString());
    }


    private void assertPrints(String program,
                              String expected) {
        String message = "Should execute: \n" + program;
        StringReader code = new StringReader(program);
        OutputStream output = new ByteArrayOutputStream();
        interpreter.setOutputStream(output);
        interpreter.eval(code);
```



```
        assertEquals(message, expected, output.toString());
    }
}
```

```java
package physicalc;

import java.lang.String;
import java.io.StringWriter;
import java.io.StringReader;
import org.junit.After;
import org.junit.Before;
import org.junit.Ignore;
import org.junit.Test;
import static org.junit.Assert.*;
import junit.framework.JUnit4TestAdapter;

/** Class to test the Number class and its arithmetic methods.  This
 * assumes a Number constructor method that takes a string (from the
 * Lexer) and returns a Number instance. */
public class NumberTest {

    /** Integers. */
    private Number one, two, three, four, five, six, eight;

    /** Decimals. */
    private Number onePointOne, twoPointThree, twoPointFiveThree,
        threePointFour, fivePointTwoNine;

    /** Numbers with exponents. */
    private Number onePoint2e24, oneE24, twoE23, twoE47, fourE46;


    @Before public void setValues() {
        one = new Number("1");
        two = new Number("2");
        three = new Number("3");
        four = new Number("4");
        five = new Number("5");
        six = new Number("6");
        eight = new Number("8");

        onePointOne = new Number("1.1");
        twoPointThree = new Number("2.3");
        twoPointFiveThree = new Number("2.53");
        threePointFour = new Number("3.4");
        fivePointTwoNine = new Number("5.29");

        onePoint2e24 = new Number("1.2e24");
        oneE24 = new Number("1e24");
        twoE23 = new Number("2e23");
        twoE47 = new Number("2e47");
        fourE46 = new Number("4e46");
    }

    /* **************************************************
     * INTEGER ARITHMETIC *
     * **************************************************/

    @Test public void addInts() {
        assertEquals("2 + 3 = 5", five, two.add(three));
    }

    @Test public void subtractInts() {
        assertEquals("4 - 3 = 1", one, four.sub(three));
    }
```



```java
    @Test public void multiplyInts() {
        assertEquals("2 * 3 = 6", six, two.mul(three));
    }

    @Test public void divideInts() {
        assertEquals("6 / 2 = 3", three, six.div(two));
    }

    @Test public void exponentInts() {
        assertEquals("2 ^ 3 = 8", eight, two.pow(three));
    }


    /* **************************************************
     * DECIMAL ARITHMETIC *
     * **************************************************/

    @Test public void addDecimals() {
        assertEquals("1.1 + 2.3 = 3.4", threePointFour,
                     onePointOne.add(twoPointThree));
    }

    @Test public void subtractDecimals() {
        assertEquals("3.4 - 1.1 = 2.3", twoPointThree,
                     threePointFour.sub(onePointOne));
    }

    @Test public void multiplyDecimals() {
        assertEquals("1.1 * 2.3 = 2.53", twoPointFiveThree,
                     onePointOne.mul(twoPointThree));
    }

    @Test public void divideDecimals() {
        assertEquals("2.53 / 1.1 = 2.3", twoPointFiveThree,
                     twoPointFiveThree.div(onePointOne));
    }

    @Test public void exponentDecimals() {
        assertEquals("2.3 ^ 2 = 5.29", fivePointTwoNine,
                     twoPointThree.pow(two));
    }


    /* **************************************************
     * NUMBER WITH EXPONENT ARITHMETIC *
     * **************************************************/

    @Test public void addExponents() {
        assertEquals("1e24 + 2e23 = 1.2e24", onePoint2e24,
                     oneE24.add(twoE23));
    }

    @Test public void subtractExponents() {
        assertEquals("1.2e24 - 2e23 = 1e24", oneE24,
                     onePoint2e24.sub(twoE23));
    }

    @Test public void multiplyExponents() {
        assertEquals("1e24 * 2e23 = 2e47", twoE47,
                     oneE24.mul(twoE23));
    }
```

```java
    @Test public void divideExponents() {
        assertEquals("2e47 / 2e23 = 1e24", oneE24,
                     twoE47.div(twoE23));
    }

    @Test public void exponentExponents() {
        assertEquals("2e23 ^ 2 = 4e46", fourE46,
                     twoE23.pow(two));
    }


    /** The suite() method is required for compatibility with older
     * JUnit versions. */
    public static junit.framework.Test suite() {
        return new JUnit4TestAdapter(NumberTest.class);
    }
}
```



```java
package physicalc;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)

@Suite.SuiteClasses({
    InterpreterTest.class
    // add additional test classes here, separated by commas
})

public class PhysicalcSuite {
    // the class remains completely empty,
    // being used only as a holder for the above annotations
    // see http://radio.javaranch.com/lasse/2006/07/27/1154024535662.html
}
```

```java
package physicalc;

import java.lang.String;
import org.junit.After;
import org.junit.Before;
import org.junit.Ignore;
import org.junit.Test;
import static org.junit.Assert.*;
import junit.framework.JUnit4TestAdapter;

/** Class to test the Unit class and its arithmetic methods. */
public class UnitTest {

    private Unit second, minute;
    private Unit meter, foot;
    private Number three, four, twelve;

    @Before public void setValues() {
        second = new Unit("second");
        meter = new Unit("meter");

        three = new Number("3");
        four = new Number("4");
        twelve = new Number("12");
    }

    @Test public void deriveFromNumbers() {
        minute = new Unit("minute", second.mul(new Number("60")));
        foot = new Unit("foot", meter.mul(new Number("3.2808399")));
    }

    @Test public void combineWithNumbers() {
        assertEquals("Should combine numbers and units when multiplying.",
                     "3 * meter",
                     three.mul(meter).toString());
        assertEquals("Should combine numbers and units when multiplying.",
                     "3 * meter",
                     meter.mul(three).toString());
    }

    @Test public void exponent() {
        assertEquals("Should accept units raised to an integer power.",
                     "meter ^ 3",
                     meter.pow(three).toString());
        assertEquals("Should accept units raised to a negative power.",
                     "meter ^ -3",
                     meter.pow(three.neg()).toString());
        assertEquals("Should accept units raised to a fractional power.",
                     "meter ^ 0.75",
                     meter.pow(three.div(four)).toString());
    }

    @Test(expected=TypeError.class)
    public void unitAsPower() {
        meter.pow(second);
    }

    @Test public void multiply() {
        assertEquals("Should combine units when multiplying them.",
                     "meter * second",
                     meter.mul(second).toString());
    }
```



```java
    @Test public void divide() {
        assertEquals("Should combine units when dividing them.",
                     "meter / second",
                     meter.div(second).toString());
    }

    /** The suite() method is required for compatibility with older
     * JUnit versions. */
     public static junit.framework.Test suite() {
        return new JUnit4TestAdapter(UnitTest.class);
     }
}
```

```
unit second
unit minute = 60 * second
unit hour = 60 * minute

set x = 1 * hour
print(x)
```



```
3600.0*second
```

```
unit meter
alias m for meter
set x = 3*m
print(x)
```



```
3.0*meter
```

```
constant x = 100
print(x)
```



```
100.0
```

```
set a = 1;
set b = 10;
set c = 2;
for i from a to b step c do
print(i)
  if (i >= 5) then
     exit()
  done
done
```



```
1.0
3.0
5.0
```

```
for i from 1 to 5 step 1 do
print(i)
done
```



```
1.0
2.0
3.0
4.0
5.0
```

```
for i from 1 to 10 step 2 do
print(i)
done
```



```
1.0
3.0
5.0
7.0
9.0
```

```
function foo(a, b)
        print(a + b)
done

foo(2, 4)
```



```
6.0
```

```
unit meter
set a = 3*meter
print(getNumber(a))
```



```
3.0
```

```
unit meter
set a = 3*meter
print(getUnit(a))
```



```
meter
```

```
unit inch
unit foot = 12*inch
print(36*inch in foot)
```



```
3.0*foot
```

```
unit second
unit minute = 60 * second
unit hour = 60 * minute
unit day = 24 * hour
unit year = 365 * day

set x = 1 * year
print(x)

unit meter
unit kilogram
unit newton = meter * kilogram / second ^ 2

set Pi = 3.1415926
set omiga = 2 * Pi/ x

set G = 6.67 * 10^-11*newton * meter ^ 2/ kilogram^2
set r = 1.50 * 10^11 meter
set m = r^3 * omiga^2 / G
print(m)
```



```
3.1536E7*second
2.01E30*kilogram
```

```
nprint("foo")
nprint("bar")
print("123")
```



```
foobar123
```

Dec 16, 07 12:44 **phy1.in** Page 1/1

```
# Calculate Factorial
# This program is testing Looping function
# Test Program written by Changlong Jiang cj2214@columbia.edu
# Date 12/15/2007

# use For Loops
set y=1
for x from 1 to 10 step 1 do
set y = y*x
done
print("use For Loop")
print(y)

# use While Loops
set y=1
set z=1
while y <= 10 do
set z = z * y
set y = y+1
done
print("use While Loop")
print(z)

#use IF
print("use While Loop and If")
set y=1
set z=1
while y <= 10 do
set z = z * y
  if y>9 then
    print("y=",y," ","greater than 9, stop")
    return y;
  elsif y>6 then
    print("y=",y," ","greater than 6, continue")
    print("result=",z)
  else
    print("y=",y," ","less and equal than 6,continue")
    print("result=",z)
  done
set y = y+1
done
```

Dec 16, 07 21:45 **phy1.out** Page 1/1

```
use For Loop
3628800.0
use While Loop
3628800.0
use While Loop and If
y=1.0 less and equal than 6,continue
result=1.0
y=2.0 less and equal than 6,continue
result=2.0
y=3.0 less and equal than 6,continue
result=6.0
y=4.0 less and equal than 6,continue
result=24.0
y=5.0 less and equal than 6,continue
result=120.0
y=6.0 less and equal than 6,continue
result=720.0
y=7.0 greater than 6, continue
result=5040.0
y=8.0 greater than 6, continue
result=40320.0
y=9.0 greater than 6, continue
result=362880.0
y=10.0 greater than 9, stop
```

```
# Calculate Factorial
# This program is testing function and logical operation
# Test Program written by Changlong Jiang cj2214@columbia.edu
# Date 12/15/2007

#this is function for factorial number
function factorial(x)
  print("x=",x)
  set y=1
  set z=1
  while y <= x do
    set z = z * y
    set y = y+1
  done
  nprint(x,"!=")
  print(z)
done

print("use function to calculate factorial")
factorial(6)

# this is function to find the biggest number
function findbiggest(x,y,z)
   print("x=",x," ","y=",y," ","z=",z)
   if x>=y and y>=z then
     print("x is biggest")
   done
   if x>=y or y>=z then
     print("x is not smallest")
   done
   if not(y>=x) then
     print("y is smaller than x")
   done
done

set x = [7,5,3]
findbiggest(x[0],x[1],x[2])
```



```
use function to calculate factorial
x=6.0
6.0!=720.0
x=7.0 y=5.0 z=3.0
x is biggest
x is not smallest
y is smaller than x
```

```
# This is for Sun Mass Calculation
# Estimate the mass of the sun given the Earth’s distance from the sun
# r=1.50*10^11 meter
# Assume the Earch follows a circular orbit
# Universal Gravitational consatant G=6.67*10^(−11)*Newton*meter^2/kilogram^2
# source from http://zebu.uoregon.edu/~probs/mech/grav
# test program written by Changlong Jiang : cj2214@columbia.edu
# Date 12/15/2007

# define the unit
unit second
unit minute = 60 * second
unit hour = 60 * minute
unit day = 24 * hour
unit year = 365 * day

unit meter
alias m for meter
unit kilogram
unit newton = m * kilogram / second ^ 2

# define the variable and calculate
set x = 1 * year
set Pi = 3.1415926
set omiga = 2 * Pi/x
set G = 6.67E−11 * newton * (1*m ^2) / (1 *kilogram ^ 2)
set r = 1.50E11 * m
set mass = (1*r^3) * (1*omiga^2)/G

#print result
print(mass)
```



```
2.0086045922465554E30*kilogram
```

```
#This is for Calculate the radius of orbit of the Moon
#Universal Gravitational consatant G=6.67*10^(-11)*Newton*meter^2/kilogram^2
#Earth Mass is 5.98E24 * kilogram
#Source from http://zebu.uoregon.edu/~probs/mech/grav/distmoon
#Test Program written by Changlong Jiang : cj2214@columbia.edu
#Date 12/15/2007

#load the pre-defined unit
load "si.phy"

#set variable
set x = 29.53 * day

nprint("seconds:",x)
print()
#print(x in hour)
set y = 29.53 * 24 * 3600*second
print("Number is:", getNumber(y))
print("Unit is:",getUnit(y))
print("hours:",y in hour)

set Pi = 3.1415926
set G = 6.67E-11 * newton * (1*meter ^ 2) /(1*kilogram^2)

set masse = 5.98E24 * kilogram
set r = ((x*(1*G*(1*masse))^(1/2))/(2*Pi))^(2/3)
print(r)
print("Number is:", getNumber(r))
print("Unit is:",getUnit(r))
```



```
seconds:2551392.0*second
Number is:2551392.0
Unit is:second
hours:708.72*hour
4.036521081066972E8*meter^1.0
Number is:4.036521081066972E8
Unit is:meter^1.0
```

```
print("Hello, World!")
```



```
Hello, World!
```

```
set x = 42
print(x)
```



```
42.0
```

Dec 08, 07 11:56 **toInt.in** Page 1/1

```
set a = 4.1
print(toInt(a))
set a = 4.9
print(toInt(a))
set a = 4.5
print(toInt(a))
```

Dec 16, 07 21:46 **toInt.out** Page 1/1

```
4.0
4.0
4.0
```

```
set n = 42
set s = toString(n)
print(s + "abc")
```



```
42.0abc
```

```
print(−4)
print(−(3*2))
print(−3*2)
set x = 42
print(−x)
```



```
−4.0
−6.0
−6.0
−42.0
```

```
unit meter
print(3*meter)
```



```
3.0*meter
```

```
unit inch
unit foot = 12*inch
print(3*foot)
```



```
36.0*inch
```

```
unit mile
unit hour
unit mph = mile / hour
print(55*mph)
```



```
55.0*mile*hour^-1
```

```
unit feet
unit mile = 5280 * feet
unit second
unit hour = 3600 * second
unit mph = mile / hour

set x = 55*mph
print(x)
```



```
80.6666667*feet/second
```

```
unit meter
unit second
unit mpss = meter / second ^ 2
print(9.8*mpss)
```



```
9.8*meter/second^2
```

```
unit meter
unit second
unit kilogram
unit newton = kilogram * meter / second ^ 2
print(100*newton)
```



```
100*kilogram*meter/second^2
```

```
unit meter
unit second
unit kilogram
unit newton = meter * kilogram / second ^ 2
set y = 2 * newton * (1* second ^2)
print(y)
```



```
2.0*meter*kilogram
```

```
while true do
print("a")
print("b")
print("c")
break
done
```



```
a
b
c
```

```
set x = 0
while x < 5 do
print(x)
set x = x + 1
done
```



```
0.0
1.0
2.0
3.0
4.0
```

```
set x = 0
while true do
    if x >= 5 then
        break
    done
    print(x)
    set x = x + 1
done
```



```
0.0
1.0
2.0
3.0
4.0
```

```
set x = 0
while x < 6 do
    print(x)
    if x = 3 then
        set x = 5
        next
    done
    print(x)
    set x = x + 1
done
```



```
0.0
0.0
1.0
1.0
2.0
2.0
3.0
5.0
5.0
```