Next
Final Report

Ernesto Arreguin (eja2124)
Danny Park (dsp2120)
Morgan Ulinski (mu2189)
Xiaowei Zhang (xz2242)

Contents

1 Introduction

The Next programming language provides a way to easily create text-based computer games. Users of the language can specify characters, locations and items that will appear in the game, and they can design the general plot of the game that ties these together. This language is ideal for creating text-based RPGs (or digital "choose your own adventure" stories).

1.1 Language Elements

Declarations

The program is primarily structured around declarations. The user will create declarations for characters, locations, items, and attributes.

Items

Items are objects that the game player can pick up and maintain in their inventory, and which can be contained in locations. The player could be required to collect item, or he might need a certain item to perform certain actions.

Characters

Characters can appear in locations throughout the game. A declaration for a character will provide the character's name, items he holds, and attribute values.

Locations

Locations provide a basic map of the game's landscape. These are the "rooms" in which a game player can find characters to interact with, items to pick up, or options for actions. For each location in the game, there are two types of declarations. It has a regular declaration, similar to a character declaration, in which its name is declared, as well as items found in that location, characters found in that location, and attributes of the location. Each location also mandatorily has an associated "start statement" which defines what gameplay will take place in that location. This start statement can contain any of the allowable statements in the language.

Actions

The way a player can interact with the game to make choices regarding gameplay is by selecting from various "actions." The programmer defines a list of possible actions, and the code that will be executed when each action is chosen. This is what allows the players to control the plot of the game.

Randomness

A key element of a language designed to create games is some element of randomness, so the gameplay will not be entirely predictable. This is implemented in our language through probability statements, a form of selection statement in which each option is assigned a probability , and the option to be executed is chosen according to what number is returned by a random number generator.

Basic Math

Our language provides the tools for performing basic math, such as addition, subtraction, and boolean operations. These can be used to declare additional variables, update the values of attributes, examining features of attributes for conditional results for actions, or anything else the programmer can think of to make use of them.

2 Language Tutorial

2.1 Tutorial 1

In this section we will create a simple program in the Next programming language that outputs "Hello world". Keep in mind that in this step by step each line individually explained is part of a single program. The Next programming language can be thought of as a language made up of declarations. A programmer can declare a number or a string. We will begin by declaring an integer and setting its value to 0:

```
int num = 0;
```

We will then declare a string:

```
string announce = "Not in the world";
```

In Next we can also declare items, characters and locations. Here we will declare a character and a location. What follows is a character declaration. The first parenthesis contains the characters attributes while the second contains the items the character is carrying. We will leave the second parenthesis empty because our character is not carrying anything:

```
character  you {(string slogan = "Hello world!"),()}
```

What follows is a location declaration. The first parenthesis contains the locations attributes, the second contains the items the location contains and the third the characters the location contains. We will leave the first two empty and only include a character in the location declaration. This code looks as follows:

```
location here  {(),(),(you)}
```

There is a special type of declaration in next in which we declare an event to start. Once a location has been declared, a start declaration must be made for that location. In order to use a location in conjunction with a start statement the location must be declared first. In order to use a character in a location that character must be declared first. In order to use an item in a character or a location the item must be declared first. This is why in the Next language it is a good programming practice to create all of the item, character and location declarations first and in that order followed by the start declarations. The start declarations must contain the programming logic for each location because aside from other declarations Next does not allow statements to be placed outside of a start declaration. The start location we will use for this program looks as follows:

```
start here end (num == 1) {
    if (exists here.you) then
    output you.slogan;
    else
    output announce;
    num = 1;

}
```

This start declaration starts the here location and specifies that it will end when integer num is equal to 1. The expression

```
if (exist here.you)  then
```

means if character you exists in location here then execute the code that follows. In this case if true then the slogan attribute from the you character is output with the line

```
output you.slogan;
```

Else the string announce is output. To finish execution of the here start statement integer num is set equal to 1 to meet the end condition. If this had not been done the start statement would iterate forever over its body of code. Because you exists in location here the program will output "Hello World!" Here is the complete program:

```
int num = 0;
string announce = "not in the world:";
character you {(string slogan = "Hello world!"),()}
location here {(),(),(you)}

start here end (num == 1) {
    if (exists here.you) then
    output you.slogan;
    else
    output announce;
    num = 1;
}
```

2.2 Compiling and Running a Next Program

Once you have created a next program save it with a .next file extension. Go to where the Next file is located. Use the Next file and your .next extension file as input and output the results into a file called Next.java. The command to do this would look like this

```
$ ./Next < Tutorial1.next > Next.java
```

In this example the Next source code file is Tutorial1.next which is located in the same folder as the Next file. This outputs a java file which can then be compiled and executed as any other java source code file. For example to compile the file you could use the command

```
$ javac Next.java
```

Then to run the file you could use

```
$ java Next
```

2.3 Tutorial 2

This tutorial assumes Tutorial 1 has been reviewed by the reader. In this tutorial we will make use of some of the more interesting features included in the Next language. The program that follows will utilize a probability statement to non-deterministically execute segments of code and we will use a choose statement to take simple keyboard input from the user and execute commands accordingly. The commands will make an item show or hide(disappear) from a location and make a character drop or grab an item from a location if the conditions are right or output the automatic error message if it is not possible to perform the action.

We will begin by creating an integer, a location and a character. Because no value will be given to the integer it will take the Next default value of 0. The location and character declarations will be created with empty lists which means that they will not have attributes, items or in the case of the location, characters. Those declarations look as follows:

```
int num;
character person {(),()}
location here {(),(),()}
```

We will then declare an item. Items can only have attributes and in this case we will also leave this items attribute list empty. The declaration looks as follows:

```
item object {()}
```

Next we will declare a start statement, very much like we did in the last example:

```
start here end (num == 1) {
```

Inside this start declaration we will begin by putting a probability statement. Probability statements begin with "[?! which in Next is known as an opening probability bracket. Probability statements end with a closing probability bracket which looks like this "?]". Inside the probability statement code that is executed with a given probability is defined with the notation prob number statement where number is an integer that specifies the probability that the statement code will be executed. The probability statement can have many such prob number statement clusters. One of them is chosen from inside

the probability statement according to the number part which specifies the probability of that cluster. It is important to remember that the sum of all the integers defined by the number part of the cluster must equal 100. The probability statement used for this program looks as follows:

```
[?
     prob 50 output "This is line 1 of a possible 2";
     prob 50 output "This is line 2 of a possible 2";
 ?]
```

This probability statement means that with a 50 percent probability the line "This is line 1 of a possible 2" will be output. If that line is not output then "This is line 2 of a possible 2" will be output (which, of course, means the second line also has a 50 percent probability of occurring as specified in the probability statement).

Following the probability statement we use two if statements. The first one checks if the item object exists in the person character. It outputs a message that notifies if it does or not. The second if statement checks if the item object exists in location here and again outputs a message that notifies the result. The code looks like this:

```
if (exists person.object) then
     output "The person has the object";
else
     output "The person does not have the object";

if(exists here.object) then
     output "The object is in the location";
else
     output ""The object is not in the location";
```

These if statements become useful to easily see the results of the choose statement that follows. A choose statement contains lists of three elements that specify a variable name, a string that will assist in output and a letter that becomes mapped to the keyboard. The first list of the choose statement in this program looks as follows:

```
(grabItem, "character grab item", "g")
```

This means that when the choose statement is executed the output "Press g for character grab item" will be presented to the user. The g keyboard key will be mapped and if the user presses the g key followed by the enter key the code associated with the grabItem variable will be executed. These associations occur through the use of "when" statements which will soon be explained. An arbitrary amount of lists for action/keyboard mappings may be used in a choose statement. The choose statement and its lists used in this example are as follows:

```
choose (grabItem, "character grab item", "g")
     (dropItem, "character drop item", "d")
     (showItem, "show item in location", "s")
     (hideItem, "hide item from location", "h")
     (exit, "exit", "e")
```

The lists for a choose statement for action/keyboard bindings are followed by a series of when statements enclosed by braces. These when statements specify code to be executed for each action. At the end of the code specified for each when statement a next statement is used that specifies a location to be executed next. The first when statement in this example is:

```
when grabItem
     grab person.object;
next here
```

This means that when the grabItem action is specified (in this example by the user choosing the g key on the keyboard) the character person will grab the item object if object is present in the current location. This means that if the item object is present in the location grab will cause object to be removed from the location and added to the list of the characters items. If the item is not present in the location when grab is used an error message is output and execution continues with no changes to the character or the location. Once grab is executed in the above when statement the next statement sends execution to the beginning of the here location.

The when statement that follows in our example is:

```
when dropItem
    drop person.object;
next here
```

In this case the drop action is used to have the person character drop the object item in the current location. drop removes an item from a characters item list if the character has that item and adds it to a locations item list. If the character does not have that item an error message is output and execution continues.

After that the following two when statements follow:

```
when showItem
     show here.object;
next here

when hideItem
     hide here.object;
next here
```

show and hide statements are used to add or remove an item or character from a location. In this case the item object is added to the here location by using the show keyword or removed by using the hide keyword. show, hide, drop and grab are used in conjunction with exists to create connections between items, locations and characters in Next programs to simulate real world relationships when creating games.

Finally the last when statement is used to end execution of location here by making its end condition true. The entire program can be seen bellow.

```
int num;

character person {(),()}
location here {(),(),()}
item object {()}

start here end (num == 1) {

     [?
          prob 50 output "This is line 1 of a possible 2";
          prob 50 output "This is line 2 of a possible 2";
     ?]

 if (exists person.object) then
      output "The person has the object";
```

```
 else
      output "The person does not have the object";

 if(exists here.object) then
      output "The object is in the location";
 else
      output "The object is not in the location";


 choose (grabItem, "character grab item", "g")
      (dropItem, "character drop item", "d")
      (showItem, "show item in location", "s")
      (hideItem, "hide item from location", "h")
      (exit, "exit", "e")
 {
     when grabItem
          grab person.object;
     next here

      when dropItem
           drop person.object;
      next here

      when showItem
           show here.object;
      next here

      when hideItem
           hide here.object;
      next here

      when exit
           num = 1;
      next here
 }

}
```

3 Language Manual

3.1 Lexicon

The Next programming language uses a standard grammar and character set. Characters in the source code are grouped into tokens, which can be punctuators, operators, identifiers, keywords, or string literals. The compiler forms the longest possible token from a given string of characters; tokens end when white space is encountered, or when it would not be possible for the next character to be part of the token. White space is defined as space characters, tab characters, return characters, and newline characters.

The compiler processes the source code and identifies tokens and locates error conditions. There are three types of errors:

- Lexical errors occur when the compiler cannot form a legal token from the character stream.
- Syntax errors occur when a legal token can be formed, but the compiler cannot make a legal statement from the tokens.
- Semantic errors, which are grammatically correct and thus pass through the parser, but break another Next rule. For example, it is possible to `kill` a character or item, but not a location.

3.1.1 Character Set

The Next programming languages accepts standard ASCII characters.

3.1.2 Identifiers

An identifier is a sequence of characters that represents a name for a:

- Variable
- Location
- Character
- Item

- Action (only within a `choose` statement)

Rules for identifiers:

- Identifiers consist of a sequence of one or more uppercase or lowercase characters, the digits 0 to 9, and the underscore character (_).
- Identifier names are case sensitive.
- Identifiers cannot begin with a digit or an underscore.
- Keywords are not identifiers.

3.1.3 Comments

Comments are introduced by `/*` and ended by `*/`, except within a string literal, where the characters `"/*"` would be displayed directly. Comments cannot be nested. If a comment is started by `/*`, the next occurrence of */ ends the comment.

3.1.4 Keywords

Keywords identify statement constructs and specify basic types. Keywords cannot be used as identifiers. The keywords are listed in Table 1.

Table 1: Keywords

`if`	`then`	`else`	`and`	`or`
`start`	`end`	`when`	`choose`	`kill`
`grab`	`hide`	`exists`	`drop`	`output`
`character`	`location`	`not`	`item`	`int`
`string`	`next`	`show`		

Keywords are used:

- To specify a data type (`character, location, item, int, string`)
- As part of a statement or start function (`if, then, else, start, end, when, choose, kill, grab, hide, drop, output, next, show`)
- As operators on expressions (`and, or, exists, not`)

3.1.5 Operators

Operators are tokens that specify an operation on at least one operand and yield a result (a value, side effect, or combination). Operands are expressions. Operators with one operand are unary operators, and operators with two operands are binary operators.

Operators are ranked by precedence, which determines which operators are evaluated before others in a statement.

Some operators are composed of more than one character, and others are single characters.

The single character operators are shown in Table 2.

Table 2: Single-character operators

+	-	*	/	>	<	=	.

The multiple-character operators are shown in Table 3.

Table 3: Multiple-character operators

>=	<=	==	!=	and	or	not	exists

3.1.6 Punctuators

Table 4 shows the punctuators in Next. Each punctuator has its own syntactic and semantic significance. Some characters can either be punctuators or operators; the context specifies the meaning.

3.1.7 String and integer literals

Strings are sequences of zero or more characters. String literals are character strings surrounded by quotation marks. String literals can include any

Table 4: Punctuators

{ }	Compound statement delimiter
( )	Member variable list; also used in expression grouping
,	Member variable separator
;	Statement end
" "	String literal
[? ?]	Probability statements

valid character, including white-space characters, except for quotation marks.

Integers are used to represent whole numbers. Next does not support floating point numbers. Integers are specified by a sequence of decimal digits. The value of the integer is computed in base 10.

3.2 Basic Concepts

3.2.1 Blocks and Compound Statements

A block is a section of code consisting of a sequence of statements. A compound statement is a block surrounded by brackets { }.

This is a block:

```
int x = 1;
x = x + 1;
output x;
```

This is a compound statement:

```
{
   int x = 1;
   x = x + 1;
   output x;
}
```

3.2.2 Scope

All declarations of locations, characters, items, strings, and integers have global scope.

Actions are listed within a `choose` statement, and their scope is that `choose` statement. Actions are variables that are given values within that scope, but they are not explicitly declared in a declaration statement.

3.3 Side Effects and Sequence Points

Any operation that affects an operand's storage has a side effect. This includes the assignment operation, and operations that alter the items, attributes, etc. within a location or a character.

Sequence points are checkpoints in the program where the compiler ensures that operations in an expression are concluded. The most important example of this in Next is the semicolon that marks the end of a statement. All expressions and side effects are completely evaluated when the semicolon is reached.

3.4 Data Types

A type is assigned to an object in its declaration. Table 5 shows the types that are used in Next.

Table 5: Data Types

`int`
`string`
`location`
`character`
`item`

3.5 Declarations

Declarations introduce identifiers (variable names) in the program, and, in the case of the complex types (`character, location, item`), specify important information about them such as attributes. When an object is declared, space is immediately allocated for it, and it is immediately assigned a value. Declarations of integers and strings do not have to include an explicit value; they will be assigned default values of 0 and " ", respectively. Next does not support complex declarations without a value for the variable.

3.5.1 Primitive Types

The primitive types in Next are integer and string. These can stand on their own, or they can serve as attributes within a complex type. Primitive types are declared as follows:

```
int identifier = value
string identifier = value
```

or, receiving default values:

```
int identifier
string identifier
```

where:

- `identifier` stands in for a variable name.
- `value` stands in for an expression.

3.5.2 Complex Types

Each of the complex types in Next (`item, character`, and `location`) has its own declaration syntax. The declarations are as follows:

```
item identifier = { primitive_declaration_list }

character identifier = { primitive_declaration_list,
                         item_list }
```

```
location identifier = { primitive_declaration_list,
                        item_list,
                        character_list }
```

where:

- `identifier` stands in for a variable name.
- `primitive_declaration_list` stands in for a list of attribute declarations in the form `(declaration_1, declaration_2, ... declaration_n)`, and each `declaration` is in the form described above in the description of primitive types. These represent the attributes (and values for those attributes) for a given item, character, or location.
- `item_list` and `character_list` stand in for lists of item and character variable names, respectively, in the form `(name_1, name_2, ... name_n)`. These represent the list of items a character is carrying or are found in a location, and the characters that are physically in a location.
- Empty lists are permitted in place of any `primitive_declaration_list`, `item_list` or `character_list`, to indicate that there are none of that type of member variable.
- All objects used within an `item_list` or `character_list` must have been declared earlier in the program.

3.6 Expressions and Operators

An expression is a sequence of Next operators and operands that produces a value or generates a side effect. The simplest expressions yield values directly, such as ints, strings, and variable names. Other expressions combine operators and subexpressions to produce values. Every expression has a type as well as a value. Operands in expressions must have compatible types.

3.6.1 Primary Expressions

The most simple type of expressions are those that denote a value directly. These include identifiers that refer to variables that have already been declared, and integer and string literals. In addition, any more complicated expression can be enclosed in parentheses and still be a valid expression.

3.6.2 Overview of the Next Operators

Variables and literals can be used in conjunction with operators to make more complex expressions. Table 6 shows the Next operators.

Table 6: Next Operators

Operator	**Example**	**Description/Meaning**
`.`	`c.a`	Attribute selection in a character, location, or item
`-` [unary]	`-a`	Negative of a
`+`	`a + b`	a plus b
`-` [binary]	`a - b`	a minus b
$*$	`a * b`	a times b
`/`	`a / b`	a divided by b
$<$	`a < b`	1 1 if a $<$ b; 0 otherwise
$>$	`a > b`	1 if a $>$ b; 0 otherwise
`<=`	`a <= b`	1 if a $\leq$ b; 0 otherwise
`>=`	`a >= b`	1 if a $\geq$ b; 0 otherwise
`==`	`a == b`	1 if a equal to b; 0 otherwise
`!=`	`a != b`	1 if a not equal to b; 0 otherwise
`and`	`a and b`	Logical AND of a and b (yields 0 or 1)
`or`	`a or b`	Logical OR of a and b (yields 0 or 1)
`not`	`not a`	Logical NOT of a (yields 0 or 1)
`=`	`a = b`	a, after b is assigned to it
`exists`	`exists x.a`	1 if a exists in location x or in the inventory of character x; 0 otherwise

The Next operators fall into the following categories:

- Unary prefix operators, which precede a single operand.
- Binary operators, which take two operands and perform some arithmetic or logical operation.
- Assignment operators, which assign a value to a variable.

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator:

```
x = 7 + 3 * 2;           /* x is assigned 13, not 20 */
```

The previous statement is equivalent to the following:

```
x = 7 + (3 * 2);
```

Using parentheses in an expression alters the default precedence. For example:

```
x = (7 + 3) * 2;          /* (7 + 3) is evaluated first */
```

In an unparenthesized expression, operators of higher precedence are evaluated before those of lower precedence. Consider the following expression:

```
A+B*C
```

The identifiers `B` and `C` are multiplied first because the multiplication operator (`*`) has higher precedence than the addition operator (`+`).

Table 7 shows the precedence the Next compiler uses to evaluate operators. Operators with the highest precedence appear at the top of the table; those with the lowest precedence appear at the bottom. Operators of equal precedence appear in the same row.

Associativity relates to precedence, and resolves any ambiguity over the grouping of operators with the same precedence. Most operators associate left-to-right, so the leftmost expressions are evaluated first. The assignment operator and the unary operators associate right-to-left.

3.6.3 Unary Operators

Unary expressions are formed by combining a unary operator with a single operand. The unary operators in Next are `-`, `not`, and `exists`. The operators

Table 7: Precedence of Next Operators

Category	**Operator**	**Associativity**
Dot	`.`	Left to right
Gameplay operators	`exists`	Non-associative
Unary	`- not`	Right to left
Multiplicative	`* /`	Left to right
Additive	`+ -`	Left to right
Relational	`< <= > >=`	Left to right
Equality	`== !=`	Left to right
Logical AND	`and`	Left to right
Logical OR	`or`	Left to right
Assignment	`=`	Right to left

`-` and `not` have equal precedence and both have right-to-left associativity, and `exists` has a higher precedence and is non-associative.

Unary Minus

The following expression:

```
- expression
```

represents the negative of the operand. The operand must have an arithmetic type.

Logical Negation

The following expression:

```
not expression
```

results in the logical (Boolean) negation of the expression. If the value of the expression is 0, the negated result is 1. If the value of the expression is not 0, the negated result is 0. The type of the result is `int`. The expression must have a scalar type.

Gameplay Operators (exists)

The following expressions:

```
exists location.x
exists character.y
```

tells us whether `x` is present either in the given location (`x` must be an `item` or `character`) or in the given character's inventory (`x` must be an `item`). The expression returns 1 if x exists and 0 otherwise. The result is type `int`.

3.6.4 Binary Operators

The binary operators are categorized as follows:

- Dot operator (`.`)
- Multiplicative operators: multiplication (`*`) and division (`/`)
- Additive operators: addition (`+`) and subtraction(`-`)
- Relational operators: less than (<), less than or equal to (<=), greater than (>), and greater than or equal to (>=)
- Equality operators: equality (==) and inequality (!=)
- Logical operators: AND (`and`) and OR (`or`)
- Assignment operator (=)

Dot operator

The dot (`.`) operator is used to select from among the attributes, items, or characters within an item, character, or location. An `item` has only attributes; a `character` has attributes and items; and a `location` has attributes, items, and characters. The syntax is:

```
x.y
```

where `x` is the containing object and `y` is the name of the sub-object or attribute which we are trying to access. The result of the expression is a reference to the sub-object `y`.

Multiplicative Operators

The multiplicative operators in Next are `*` and `/`. Operands must have arithmetic type.

The `*` operator performs multiplication.

The `/` operator performs division. If two integers don't divide evenly, the result is truncated, not rounded.

Additive Operators

The additive operators in Next are `+` and `-`. They perform addition and subtraction respectively.

Relational Operators

The relational operators compare two operands and produce an integer literal result. The result is 0 if the relation is false, and 1 if it is true. The operators are: less than (<), greater than (>), less than or equal (<=), and greater than or equal (>=). Both operands must have an arithmetic type.

The relational operators associate from left to right. Therefore, the following statement first relates `a` to `b`, resulting in either 0 or 1. The resulting 0 or 1 is compared with `c` for the expression result.

```
if ( a < b < c)
{
   statement;
}
```

Equality Operators

The equality operators in Next, equal (`==`) and not-equal (`!=`), like relational operators, produce a result of an integer literal. In the following statement, the result is 1 if both operands have the same value, and 0 if they do not:

```
a == b
```

The operands must be type `int` or type `string`, and they must be of identical type.

Logical Operators

The logical operators are `and` and `or`. These operators have left-to-right evaluation. The resulting integer literal is either 0 (false) or 1 (true). Both operators must have scalar types. If the compiler can make an evaluation by examining only the left operand, the right operand is not evaluated.

In the following expression,

```
E1 and E2
```

the result is 1 if both operands are nonzero, or 0 if one operand is 0.

In the same way, the following expression is 1 if either operand is nonzero, and 0 otherwise. If expression `E1` is nonzero, expression `E2` is not evaluated.

```
E1 or E2
```

3.6.5 Assignment Operator

There is only one assignment operator (`=`) in Next. An assignment results in the value of the target variable after the assignment. It can be used as subexpressions in larger expressions. Outside of the declaration section, assignment operators can only operate on primitive types; these are integer and string literals. Assignment expressions have two operands: a modifiable value on the left and an expression on the right. In the following assignment:

```
E1 = E2;
```

the value of expression `E2` is assigned to `E1`. The type is the type of `E1`, and the result is the value of `E1` after completion of the operation.

3.7 Statements

Statements are executed in the sequence in which they appear in the code, unless otherwise specified.

3.7.1 Labeled Statements

Next uses labeled statements within the content of a `choose` statement. The location name included in a `start` statement is also a type of label. Both of these statements will be discussed in more detail in later sections.

3.7.2 Compound Statements

A compound statement, or block, allows a sequence of statements to be treated as a single statement. A compound statement begins with a left brace, contains (optionally) statements, and ends with a right brace, as in the following example:

```
{
   fred.speed = 5;
   if (fred.strength > enemy.strength) then
   {
      enemy.health = enemy.health - 10;
   }
   else
   {
      fred.health = fred.health - 10;
   }
   enemy.strength = 50;
}
```

3.7.3 Expression Statements

Any valid expression can be used as a statement by following the expression with a semicolon.

3.7.4 Selection Statements

A selection statement selects among a set of statements depending on the value of a controlling expression, or, in the case of `prob` statements, the value of a random number. The selection statements in Next are the `if` statement, the `choose` statement, and the `prob` statement.

The if Statement

The `if` statement has the following syntax:

```
if expression then
   statement
else
   statement2
```

The statement following the control expression is executed if the value of the control expression is true (nonzero). The statement in the `else` clause is executed if the control expression is false (0). Next does not allow `if` statements without a corresponding `else` clause.

```
if x + 1 then
   output "x was not negative one";
else
   output "x was negative one";
```

When `if` statements are nested, an `else` clause matches the most recent `if` statement that does not have an `else` clause, and is in the same block.

The choose Statement

The `choose` statement maps keyboard keys to specific actions and executes the statements associated with that action, given user input.

Syntax:

```
/* associates action1 with key1 and action2 with key2 */
choose (action1,action1name,key1) (action2, action2name, key2)
{
```

```
    when action1
    {
        statement
    } next location
    when action2
    {
        statement
    } next location
 }
```

For example:

```
choose (boom,"blowup","a") (punch,"punch","u")
{
    when boom
    {
        link.life = link.life - 80;
    } next palace
    when punch
    {
        [? prob 40 link.life = link.life - 1;
           prob 60 link.life = link.life - 33; ?]
     } next dungeon
}
```

If the game player enters "a" on the keyboard, code within the `boom` block is executed. If the game player enters "u" on the keyboard, code within the `punch` block is executed.

The prob Statement

A probability statement executes a statement randomly according to the given probabilities. It selects and executes a statement randomly given a certain probability distribution. The probabilities listed within a given `prob` statement must add up to 100.

Syntax:

```
[?  prob expression statement
      ...
 ?]
```

For example:

```
/* Increments count by 1 60% of the time
   and decrements count by 1 40% of the time */
[?  prob 40 count = count-1;
    prob 60 count = count+1; ?]
```

3.7.5 Gameplay Statements

Grab and Drop

The statements:

```
grab y.x;
drop y.x;
```

operate on the item `x` in character `y`'s inventory. The `grab` statement removes an item from the current location and adds in to the character's inventory. The `drop` statement removes an item from the character's inventory and adds it to the current location. The object `x` must be of type `item`.

Hide and Show

The statements:

```
hide y.x;
show y.x;
```

operate on the item or character `x` within the location `y`. The `hide` statement removes `x` from `y`, and the `show` statement adds `x` to `y`. The object `y` must be a `location`. The object `x` must be an `item` or `character`.

Output

The statement:

```
output expression;
```

outputs `expression` to the screen. Object `expression` must be of type `string` or `int`.

Kill

The statement:

```
kill x;
```

removes `x` permanently from global memory, not just for a given character or location. There is no way to retrieve `x` from any part of the game after this operation has taken place. If you would like to make a character or item disappear temporarily, use the `hide` command. The object `x` must be of type `item` or `character`.

Next

The statement:

```
next l;
```

moves the main character from the current location to location `l`. Code continues execution at the beginning of the `start` statement marked with identifier `l`.

3.8 Start Statements

Start statements are where iteration takes place in Next. They begin with the `start` keyword. The statements inside the following block are executed until an end condition is met. The end condition must have a scalar type. An iteration statement is specified by the location in which the action occurs and an end condition that clarifies when gameplay should stop. Every location that is declared in the declarations section must have a corresponding start

statement somewhere in the following code. All other types of statements must occur within the start statement block.

The syntax of a `start` statement is as follows:

```
start location_identifier end (expression)
   statement
```

where `location_identifier` is the identifier representing the location, and `expression` represents the end condition. For example:

```
start myplace end (junior.life < 1)
{
   junior.life = junior.life - 1;
   output "You have reached the mountains of myplace!";
}
```

3.9 Example Program

```
int count;

item the_greatest_sword_ever {(int damage = 100000000)}

character xiaowei_the_greatest_man_ever {(int life = 100000000,
int level=99999, string haha="hahahahaha"), (the_greatest_sword_ever)}

location where_is_this_place {(int sizex = 10000, int sizey=9283), (),
(xiaowei_the_greatest_man_ever)}

start where_is_this_place end (xiaowei_the_greatest_man_ever.life < 0) {
     choose (attack,"hia!","a") (up,"up","u") (fin, "end", "f") {
          when attack
          {
              xiaowei_the_greatest_man_ever.life+1;

           } next where_is_this_place

          when up
```

```
          {
               [? prob 40 {
                    count = count-1;
                   output "count:";
                    output count;
               }
               prob 60 {
                    count = count+1;
                    output "count:";
                    output count;
               }
                ?]
          } next where_is_this_place

          when fin
          {
               xiaowei_the_greatest_man_ever.life = -1;
          } next where_is_this_place


     }
}
```

4 Project Plan

4.1 Process

We started work on our language by brainstorming what sort of features we wanted it to include, and sketching out mini programs of the kind we wanted our language to support. We filled up many sheets of notebook paper with ideas for the syntax and structure of our programs. As these ideas solidified, we began to create our parser, scanner, and AST, and develop the Language Reference Manual. The discussion while writing the code for the parser and scanner, while cross-referencing with the LRM, fine-tuned many of the details of our language.

After the parser and scanner were complete, we made the decision to make our compiler translate Next code into Java. We divided the work to

create the translator into different sections for each person to work on. Each person was responsible for the code for certain elements of the language, for example expressions, declarations, and the various types of statements.

As the translator code neared completion, we also began work on the testing of our program. We created many small unit tests to test the features of our language, and running these allowed us to see what was and wasn't working properly in our code. As we tested, sometimes implementation details needed to be changed, and we discussed these in the group before changing them.

Throughout the work on our project, we had biweekly meetings to make sure everyone was on the same page and work through issues that required whole-group input.

4.2 Programming Style Guide

4.2.1 Width of the page

80 characters Column Limit

4.2.2 Height of the page

A function should always fit one screenful (of about 70 lines) , at most three

4.2.3 Using tab stops

No Tab Characters is allowed

4.2.4 Comments

Comments Go Above the Code They Reference

4.2.5 Indentation

The change in indentation between successive lines of the program is 2 spaces

indent let ... in constructs

The expression following a definition introduced by let is indented to the same level as the keyword let, and the keyword in which introduces it is written at the end

indent if ... then ... else ...

```
if cond1  then
else if cond2  then
else if cond3  then
```

indent pattern-matching

All the pattern-matching clauses are introduced by a vertical bar, including the first one. For a match or a try align the clauses with the beginning of the construct. If the expression on the right of the pattern matching arrow is too large, cut the line after the arrow.

For Pattern matching in named functions. Indent the clauses 2 spaces with respect to the beginning of the construct

4.3 Project timeline

We kept a calendar to keep track of the important dates for this project.

9/20: Begin biweekly meetings.
9/29: Project proposal submitted.
10/5: Received feedback on proposal, began work on parser/scanner/AST.
11/3: Parser, scanner, AST, and Language Reference Manual complete.
12/1: Ocaml code complete, begin testing and debugging.
12/15: All unit tests written.
12/20: Report drafts finished.
12/22: Project is done!

4.4 Roles and Responsibilities

All members of the team helped write the OCaml code for this project and prepare Next test cases.

Project Leader: Morgan - divided jobs among team members, arbitrated disagreements about language elements, put together the written reports, wrote the implementation for selection statements (if, choose, prob), and much of the supporting Java code.

Language Guru: Ernesto - primary source of ideas for our language and what it should do, wrote scanner, wrote implementation of start statements and action statements (kill, grab, drop, show, hide), wrote the larger sample Next programs.

IT Support: Danny - managed our Github repository, set up automation for test scripts, wrote the implementation of declarations and expressions.

AST Expert: Xiao - wrote much of the parser, go-to person for problem with the parser or AST, wrote code to do type-checking on AST, wrote implementation of compound statements and output statements.

4.5 Software Development Environment

Our translator was written using OCaml. We used ocamllex and ocamlyacc to create the scanner and parser, respectively. Code was developed in text files and compiled at the command line with a make file. Since our compiler translates code into Java, we also used the Java compiler and JVM to run our programs. Source control was done via Github.

5 Architectural Design

5.1 Architecture Block Diagram

The Next compiler consists of the following major components:

1. Scanner (implemented using Ocamllex)
   Scans the input source file and generates tokens

2. Parser (implemented using Ocamlyacc)
   Parse the tokens from the scanner and generates concrete syntax tree (CST) for the given source program

3. Semantic and type checking module
   Checks the CST for semantic or type errors and also builds up a symbol table

4. Java library code
   Necessary Java class, function and method thats included in the final output java code

5. Java code generator
   Translates the CST to java code and appends the Java library code to the final output

The interaction of these components is shown in figure 1. The Next compiler takes a Next Source File (conventionally suffixed by .Next) and feeds it to the Next scanner implemented using Ocamllex. The scanner captures the input source file, generates a token stream and passes it to the parser. On receiving the token stream, the parser checks the syntax and builds up the concrete syntax tree. The generated concrete syntax tree is first fed to the checking module to detect any semantic or type errors. During this checking pass, a symbol table is built up as well. After the checking pass, the Java code generator picks up the symbol table, goes through the concrete syntax tree again and translates the CST to java code. In the final stage, the necessary library code is included in the final output java file by the java code generator as well.

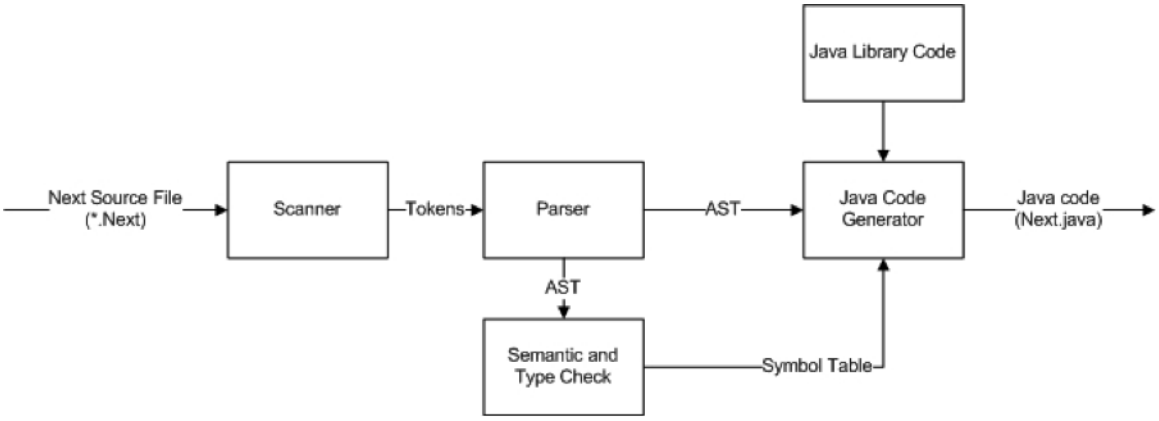

Figure 1: Block diagram of Next compiler

5.2 Interfaces Description

1. Scanner
   Scanner has one public interface which takes an input from the stdin and returns a token stream. If any invalid token is captured, an exception will be raised

2. Parser
   Parser has one public interface which takes the token stream from the scanner as input and returns the generated CST. If any syntax error is captured, a parsing exception will be raised.

3. Semantic and type checking module
   This checking module has one major public interface which takes the CST generated by the parser and returns a symbol table. Also this module provides several query functions for the Java code generator to call to determine the type of variables or expressions. Namely, Check_id function takes in an identifier in the CST together with the symbol table and returns the type of the identifier.
   Check_expr function takes in an expression in Next and returns the type of the expression These functions are called by the Java code generator to when its necessary to know the type of the expression or identifier.

4. Java code generator
   The code generator provides one public interface which takes the symbol table and the CST and output the final java code to stdout. The code generator calls the check_id and check_expr functions provided by the checking module when the type information about a specific identifier or expression is need.

6 Test Plan

A test plan was created to reflect the aspects of the program that were covered in the Language Reference Manual. We thought that creating tests in this manner would allow us to create a test suite that would be comprehensive and at the same time ensure the clarity of our LRM. Since any developer that would be using our language would refer to our manual, everything mentioned in our manual should work in the way described.

The tests were first divided up into tests that could be automated and tests that would need to be tested manually.

Manual Tests:

1. Choose Statement
2. Probability Statement
3. Start Statement

Manual tests were ones that were either non-deterministic or required user input.

Automated Tests:

1. And, Or Operators
2. Arithmetic Operators
3. Comparison Operators
4. Dot Operator
5. Exists Operator
6. Unary Operators
7. Character Declarations
8. Integer Declarations
9. Item Declarations
10. String Declarations
11. Location Declarations
12. Output Statement
13. Show Statement
14. Kill Statement

15. Compound Statements

16. Grab, Drop Statements

All automated tests had a deterministic output that was compared to a reference file using a Perl Script.

Automation:

The Perl Script (RunTests.pl) that was created to automate the testing process does so in the following steps.

1. Run CleanTests.pl, which removes any previous java and output files in each respective test folder.

2. Iterate through each test folder and perform the following steps
   a. Compile the *.next file
   b. Compile the Next.java file
   c. Run the compiled Next program
   d. Compare the output file with the reference file
   e. Record in FailedTests.txt any files that do not have matching output and reference file

Complete Sample and Tutorial Programs:

Sample and tutorial programs that incorporate various aspects of the Next programming language were also created in order to make sure that the Next language works in its entirety and not just in separate pieces. These samples programs are included in the appendix of this report.

7 Lessons Learned

7.1 Ernesto

This semester was filled with lots of lessons to be learned in PLT. It was interesting to see how while creating our language, every time we hit a stumbling block on the creation of our AST the solution was to create a new type that took the other elements of the language bellow it (another level of indirection). In general just learning about tokens and how languages work

changed my way of looking at programming. It was exciting to see how expressions resolve and become statements to go on into declarations (at least for the language my team created). I also learned about the many considerations that go into making a programming language such as scope, syntax and structure. Specifically for the language we created I learned about flexibility. Our language would have been a lot more flexible and intuitive if we had used smaller statements such as separating the next statement from the when statement. It is true that less is better in this case. It was also interesting to see how programming languages currently seem to be set to work in certain typical ways but really, they can be organized and made to work in any way as long as a language is well defined. It leaves a lot of room for creativity.

Regarding team work I learned that having strong and involved team members and a person with the ultimate word really helps. Disputes are resolved with a lot more ease if one person has the last say. It is a lot more efficient.

Regarding development I learned that Ocaml is a fantastic language when it comes to creating programming languages because of its strong data type system and the way its functions are created and used. At the same time I learned about the weaknesses of trying to port a language that uses integer values to express true/false to a language that uses Boolean values. Expressions can become quite complex to handle.

For students next semester I would agree that it is definitely good to start early, especially on the details of the language. My team had the parser, scanner and AST done within a month and it gave us a lot of room to think about what we had and tweak decisions with something to fall back on rather than finding out last minute something is terribly wrong. Also a system of checks I would say helps. Have someone specialize on a certain part of the report but have someone else overview and understand what is happening. This helps for decisions to get a second opinion for debugging and error checking and when the rest of the tea does not understand something the expert is usually too immersed on the material to be able to explain things well but a person who understands but is not so impregnated is able to clarify concepts in lay-mans terms.

7.2 Danny

1. Pick a source control you are comfortable with.

I proposed to the group that we use Git as our source control program. Although learning a new source control seemed exciting and like a great idea at the time, if I had to do it again, I would have picked SVN or CVS. The reason being is that Git worked for us most of the time but when things went wrong, such as a merge, it was very frustrating and time consuming figuring out how to fix it. There were a couple of times where I lost work because of some mix-up. Time lost figuring out Git could have been used better elsewhere.

2. Set Goals

Our group had a Google Calendar that we setup early in the semester with weekly meeting times as well milestones that we wanted to hit. We did not always hit our milestones, for example being code complete by Thanksgiving, but it did provide a sense of where we were compared to where we wanted to be. This translated into us working harder in order to stay on schedule so we would not be dealing with a waterfall of work in the last week.

3. Meet Regularly (In person or electronically)

I think this was the main reason that made our group as efficient as it was. Regular meetings allowed us to clarify and refine our language and make incremental changes as we went along as opposed to changing the whole architecture of our language every 2-3 weeks. This also helped to keep everybody accountable since it is hard not to do any work when everybody else has something to show every couple days.

7.3 Morgan

It really helps to very specifically designate jobs and responsibilities to each person. This ensures that everyone is clear on what they should be doing, and it helps the work get done more efficiently. It is also useful when it comes to debugging and fixing errors in the code to know exactly who was responsible for that section, so that someone familiar with the design of that section can be the expert on fixing that problem.

Communication is also key. Our group met twice a week to discuss our project and work on issues that had come up. This system made sure that everyone was always on the same page in regard to the project. We also communicated frequently via email as problems came up, which was also effective as all members of our group responded quickly to emails.

Our project ran very smoothly overall, so I think that the systems we had in place worked well. It also helps if all members of the group have a good work ethic and are responsible in completing tasks, which we were lucky in having as well.

7.4 Xiaowei

Communication is very important, especially in a collaborative project like this. It really takes time for everyone to understand each other (ambiguous in language might be the cause, need a better parser). Source control is another important thing. If we dont have the source control, I can imagine what a mess it is going to be.

8 Appendix

Attached to the end of this report are a complete code listing of our translator, sample Next programs, and test files.

Appendix

Translator Code Listing

```
(*action.ml by Ernesto *)
open Ast
open Checktype
open Check



module type ACTION = sig

   val kill_to_java : string -> string list
   val grab_to_java : string -> string -> string list
   val drop_to_java : string -> string -> string list
   val show_to_java : string -> string -> ('a VarMap.t VarMap.t) -> string list
   val hide_to_java : string -> string -> ('a VarMap.t VarMap.t) -> string list

end

module Action : ACTION = struct

let kill_to_java str =
  ["killFunction(\"" ^ str ^ "\");"]
let grab_to_java str1 str2 =
  [str1 ^ ".addItem (\"" ^ str2 ^"\", currentLocation, locations);"]
let drop_to_java str1 str2 =
  [str1 ^ ".removeItem (\"" ^ str2 ^"\", currentLocation, locations);"]

let show_to_java str1 str2 mapt =
  if (VarMap.mem (str2,Item) mapt) then
      [str1 ^ ".addItem (\"" ^ str2^ "\",items);"]
  else [str1 ^ ".showCharacter (\"" ^ str2^ "\", characters);"]

let hide_to_java str1 str2 mapt =
  if (VarMap.mem (str2,Item) mapt) then
      [str1 ^ ".removeItem (\"" ^ str2 ^"\");"]
  else [str1 ^ ".hideCharacter (\"" ^ str2 ^"\");"]

end
```

```
(* ast.mli by Xiao *)
type operator =
      Add
  | Sub
  | Mul
  | Div
  | Or
  | And
  | Eq
  | Lt
  | Gt
  | Neq
  | Leq
  | Geq



type pridec =
      Strdec of string
  | Intdec of string
  | Strdecinit of string * expr
  | Intdecinit of string * expr

and membervarlist = membervar list

and membervar =
      Primember of pridec
  | Varref of string

and expr =
    Binop of expr * operator * expr
  | Asn of id * expr
  | Lit of int
  | LitS of string
  | Exists of string * string
  | Neg of expr
  | Not of expr
  | Ident of id

and id =
        Var of string
      | Has of string * string

and probexpr =
      Unitprob of int * stmt

and probexprlist = probexpr list

and actiondec =
      Unitaction of string * string * string

and whenexpr =
      Unitwhen of string * stmt * string

and stmt =
      Ifelse of expr * stmt * stmt
  | Chwhen of actiondeclist * whenexprlist
  | Prob of probexprlist
  | Kill of string
  | Grab of string * string
  | Drop of string * string
  | Show of string * string
  | Hide of string * string
  | Atomstmt of expr
  | Cmpdstmt of block
  | Nostmt of int
  | Print of expr
and block = stmt list
and actiondeclist = actiondec list
and whenexprlist = whenexpr list

and globaldec =
```

```
    IntStrdec of pridec
  | Charadec of string * membervar list * membervar list
  | Itemdec of string * membervar list
  | Locdec of string * membervar list * membervar list * membervar list
  | Startend of string * expr * stmt

and globaldecs = globaldec list

and program = globaldecs
```

```
(* check.ml by Xiao *)
open Ast
open Checktype

module VarMap = Map.Make( struct
  type t = string * Checktype.t
  let compare x y = Pervasives.compare x y
  end)

module StringMap = Map.Make(String)

exception DupVar of string
exception NotFound of string
exception WrongType of string
exception ProbError of string
exception InvalidKey of string


let print_next_type = function
      Integer -> print_string "int"
  | String -> print_string "string"
  | Character -> print_string "character"
  | Item -> print_string "item"
  | Location -> print_string "location"
  | Action -> print_string "action"
  | Key -> print_string "key"

let print_symboltable symt =
  let print_entry (name,t) tt =
    print_string name;
    print_next_type t;
    print_endline "" in
  VarMap.iter print_entry symt


let check_id symt = function
    Var (name) ->
      if (not (VarMap.mem (name,String) symt)) &&
         (not (VarMap.mem (name,Integer) symt)) &&
         (not (VarMap.mem (name,Location) symt)) &&
         (not (VarMap.mem (name,Character) symt)) &&
         (not (VarMap.mem (name,Item) symt)) then
         raise ( NotFound("undefined variable " ^ name))
        else if (VarMap.mem (name,String) symt) then String
        else if (VarMap.mem (name,Integer) symt) then Integer
        else if (VarMap.mem (name,Location) symt) then Location
        else if (VarMap.mem (name,Character) symt) then Character
        else Item
  | Has (name, subname) ->
      if (not (VarMap.mem (name,Character) symt) ) &&
         (not (VarMap.mem (name,Item) symt) ) &&
         (not (VarMap.mem (name,Location) symt) ) then
            raise ( NotFound("undefined variable " ^ name))
        else
            if (VarMap.mem (name,Character) symt) then
              let subsymt = VarMap.find (name,Character) symt in
              if (not (VarMap.mem (subname, Integer) subsymt)) &&
                 (not (VarMap.mem (subname, String) subsymt)) then
                  raise ( NotFound("member not found " ^ name ^"." ^ subname))
              else
                  if (VarMap.mem (subname, Integer) subsymt) then Integer
                  else String
            else if (VarMap.mem (name,Location) symt) then
              let subsymt = VarMap.find (name,Location) symt in
              if (not (VarMap.mem (subname, Integer) subsymt)) &&
                 (not (VarMap.mem (subname, String) subsymt)) then
                raise ( NotFound("member not found " ^ name ^"." ^ subname))
              else
                  if (VarMap.mem (subname, Integer) subsymt) then Integer
                  else String
            else
              let subsymt = VarMap.find (name,Item) symt in
```

```
              if (not (VarMap.mem (subname, Integer) subsymt)) &&
                 (not (VarMap.mem (subname, String) subsymt)) then
                  raise ( NotFound("member not found " ^ name ^"." ^ subname))
              else
                  if (VarMap.mem (subname, Integer) subsymt) then Integer
                  else String


let rec check_expr symt = function
    Binop (expr1 , op, expr2) ->
      if ((op = Eq) || (op = Neq)) then
      if ((check_expr symt expr1)=String &&
         (check_expr symt expr2) = String) then
        Integer
      else if ((check_expr symt expr1)=Integer &&
              (check_expr symt expr2) = Integer) then
        Integer
      else
        raise (WrongType("Type does not match"))
      else
      if (check_expr symt expr1)=Integer &&
         (check_expr symt expr2) = Integer then
        Integer
      else raise (WrongType("Type does not match"))
  | Asn (id, expr) ->
      if ((check_id symt id) = Integer &&
         (check_expr symt expr) = Integer) then Integer
        else if ((check_id symt id) = String &&
        (check_expr symt expr) = String) then String
        else raise (WrongType("Type does not match"))
  | Lit (intvalue) -> Integer
  | LitS (strvalue) -> String
  | Exists (name, subname) ->
      if (VarMap.mem (name, Location) symt) then
            (*print_symboltable subsymt;*)
          if ( (not (VarMap.mem (subname, Item) symt)) &&
             (not (VarMap.mem (subname, Character) symt)) ) then
              raise ( NotFound("Exist error 1" ^ name ^"." ^ subname))
            else
              Integer
        else if (VarMap.mem (name, Character) symt) then
            (*let subsymt = VarMap.find (name,Character) symt in*)
            if ( not (VarMap.mem (subname, Item) symt) ) then
              raise ( NotFound("Exist error 2" ^ name ^"." ^ subname))
            else
              Integer
        else
            raise ( NotFound("Exist error 3" ^ name ^"." ^ subname))



  | Neg (expr) ->
      if (check_expr symt expr) = Integer then Integer
      else raise (WrongType("Type does not match"))
  | Not (expr) ->
      if (check_expr symt expr) = Integer then Integer
      else raise (WrongType("Type does not match"))
  | Ident (id) -> check_id symt id


let check_action = fun actionmap actiondec ->
  match actiondec with
      Unitaction (vname, outstr, key) ->
        if (VarMap.mem (vname,Action) actionmap) then
          raise ( DupVar("duplicated action name " ^ vname))
        else if (VarMap.mem (key,Key) actionmap) then
          raise ( DupVar("Multiple binding for key " ^ key))
        else if not (String.length key = 1) then
          raise (InvalidKey ("Key should be one single character" ^ key))
        else if not ( (Char.code key.[0]) >= (Char.code 'a') &&
                      (Char.code key.[0]) <= (Char.code 'z') ||
                        (Char.code key.[0]) >= (Char.code 'A') &&
```

```
                        (Char.code key.[0]) <= (Char.code 'Z') ||
                              (Char.code key.[0]) >= (Char.code '0') &&
                              (Char.code key.[0]) <= (Char.code '9') ) then
              raise (InvalidKey ("Key should be either a digit or a letter" ^ key))
          else
              let actionmap = VarMap.add (vname, Action) Action actionmap in
              VarMap.add (key, Key) Key actionmap

let rec check_probexpr symt total probexpr =
  match probexpr with
      Unitprob (pvalue, probstmt) ->
        check_stmt symt probstmt;
        total+pvalue
and check_whenexpr symt actionmap whenexpr =
  match whenexpr with
      Unitwhen (actionname, whenstmt, locname) ->
       (if not (VarMap.mem (actionname, Action) actionmap) then
          raise ( NotFound("action not defined " ^ actionname))
        else check_stmt symt whenstmt);
        if not (VarMap.mem (locname, Location) symt) then
          raise ( NotFound("Location not found in next " ^ locname))
        else ()
and check_stmt symt =  function
      Ifelse (cond, truestmt, falsestmt) ->
        if not ((check_expr symt cond) = Integer) then
          raise (WrongType("Type does not match"))
        else
          check_stmt symt truestmt;
          check_stmt symt falsestmt
  | Chwhen (actiondeclist, whenexprlist) ->
      let actionmap = List.fold_left check_action VarMap.empty actiondeclist in
        List.iter (check_whenexpr symt actionmap) whenexprlist



  | Prob (probexprlist) ->
      let total = List.fold_left (check_probexpr symt) 0 probexprlist in
        if not (total = 100) then
          raise ( ProbError ("Total Probability is not 100 "))
        else ()
  | Kill (name) ->
      if (not (VarMap.mem (name,Character) symt) ) &&
         (not (VarMap.mem (name,Item) symt) ) then
        raise ( NotFound("Var not found, kill fail " ^ name))
        else ()
  | Grab (name, subname) ->
      if (not (VarMap.mem (name,Character) symt) ) then
        raise ( NotFound("Charactor not found, invalid grab " ^ name))
        else if (not (VarMap.mem (subname,Item) symt) ) then
          raise ( NotFound("Item not found, invalid grab " ^ subname))
        else ()
  | Drop (name, subname) ->
      if (not (VarMap.mem (name,Character) symt) ) then
        raise ( NotFound("Charactor not found, invalid drop " ^ name))
        else if (not (VarMap.mem (subname,Item) symt) ) then
          raise ( NotFound("Item not found, invalid drop " ^ subname))
        else ()
  | Show (name, subname) ->
      if (not (VarMap.mem (name,Location) symt) ) then
        raise ( NotFound("Location not found, invalid show " ^ name))
        else if ( (not (VarMap.mem (subname,Item) symt)) &&
                        (not (VarMap.mem (subname,Character) symt)) ) then
            raise (NotFound("Item or Character not found, invalid show " ^ subname))
        else ()
  | Hide (name, subname) ->
      if (not (VarMap.mem (name,Location) symt) ) then
        raise ( NotFound("Location not found, invalid hide " ^ name))
        else if ( (not (VarMap.mem (subname,Item) symt)) &&
                  (not (VarMap.mem (subname,Character) symt)) ) then
            raise (NotFound("Item or Character not found, invalid hide " ^ subname))
        else ()
```

```
  | Atomstmt (exp) -> ignore (check_expr symt exp)
  | Cmpdstmt (blk) -> List.iter (check_stmt symt) blk
  | Nostmt (nonsense) -> ()
  | Print  (exp) ->
      if (not ((check_expr symt exp) = Integer)) &&
         (not ((check_expr symt exp) = String)) then
        raise (WrongType("Type does not match"))
        else ()

let pridec_tostr = function
   Strdec (name) -> name
 | Intdec (name) -> name
 | Strdecinit (name,initexpr) -> name
 | Intdecinit (name,initexpr) -> name

let check_pridec symt = function
      Strdec (name) ->
        (*print_endline("symt----------");
            print_symboltable symt;*)
        if (VarMap.mem (name, Location) symt) ||
             (VarMap.mem (name, Character) symt) ||
             (VarMap.mem (name, Item) symt) ||
             (VarMap.mem (name, String) symt) ||
             (VarMap.mem (name, Integer) symt) then
             (* *)
          (print_endline("error");
            raise ( DupVar("duplicated identifier " ^ name)))
        else
            VarMap.add (name, String) VarMap.empty symt

  | Intdec (name) ->
      (*print_endline("symt----------");
            print_symboltable symt;*)
        if (VarMap.mem (name, Location) symt) ||
             (VarMap.mem (name, Character) symt) ||
             (VarMap.mem (name, Item) symt) ||
             (VarMap.mem (name, String) symt) ||
             (VarMap.mem (name, Integer) symt) then
          (print_endline("error");
          (* *)
            raise ( DupVar("duplicated identifier " ^ name)))
        else
            VarMap.add (name, Integer) VarMap.empty symt

  | Strdecinit (name,initexpr) ->
      ignore (check_expr symt initexpr);
        (*print_endline("symt----------");
        print_symboltable symt;*)
        if (VarMap.mem (name, Location) symt) ||
             (VarMap.mem (name, Character) symt) ||
             (VarMap.mem (name, Item) symt) ||
             (VarMap.mem (name, String) symt) ||
             (VarMap.mem (name, Integer) symt) then
          (* *)
            (print_endline("error");
            raise ( DupVar("duplicated identifier " ^ name)))
        else
            VarMap.add (name, String) VarMap.empty symt
  | Intdecinit (name,initexpr) ->
      ignore (check_expr symt initexpr);
        (*print_symboltable symt;*)
        if (VarMap.mem (name, Location) symt) ||
             (VarMap.mem (name, Character) symt) ||
             (VarMap.mem (name, Item) symt) ||
             (VarMap.mem (name, String) symt) ||
             (VarMap.mem (name, Integer) symt) then
          (* *)
            (print_endline("error");
            raise ( DupVar("duplicated identifier " ^ name)))
        else
            VarMap.add (name, Integer) VarMap.empty symt
```

```
let rec check_membervarlist_intstr subsymt = function
    [] -> subsymt
  | member::tl ->
      match member with
            Primember (pridec) ->
              check_membervarlist_intstr (check_pridec subsymt pridec) tl
        | Varref (varname) -> raise (WrongType("Type does not match "^varname))

let rec check_membervarlist_item symt subsymt = function
    [] -> subsymt
  | member::tl ->
      match member with
            Primember (pridec) ->
              raise (WrongType("Type does not match " ^(pridec_tostr pridec)))
        | Varref (varname) ->
            if not (VarMap.mem (varname, Item) symt) then
                  raise ( NotFound("undefined variable " ^ varname))
              else
                  if (VarMap.mem (varname,Item) subsymt) then
                    raise ( DupVar("duplicated identifier " ^ varname))
                  else
                    check_membervarlist_item symt
                    (VarMap.add (varname,Item) VarMap.empty subsymt) tl

let rec check_membervarlist_chara symt subsymt = function
    [] -> subsymt
  | member::tl ->
      match member with
            Primember (pridec) ->
              raise (WrongType("Type does not match " ^(pridec_tostr pridec)))
        | Varref (varname) ->
            if not (VarMap.mem (varname, Character) symt) then
                raise ( NotFound("undefined variable " ^ varname))
              else
                  if (VarMap.mem (varname,Character) subsymt) then
                    raise ( DupVar("duplicated identifier " ^ varname))
                  else
                    check_membervarlist_chara symt
                    (VarMap.add (varname,Character) VarMap.empty subsymt) tl


let check_globaldec symt locmap = function
    IntStrdec (pridec) -> check_pridec symt pridec, locmap
  | Charadec (name, memberlist1, memberlist2) ->
      (*print_symboltable symt;*)
        if (VarMap.mem (name, Location) symt) ||
             (VarMap.mem (name, Character) symt) ||
             (VarMap.mem (name, Item) symt) ||
             (VarMap.mem (name, String) symt) ||
             (VarMap.mem (name, Integer) symt) then
          raise ( DupVar("duplicated identifier in character dec " ^ name))
        else
            let subsymt = VarMap.empty in
            let subsymt = check_membervarlist_intstr subsymt memberlist1 in
            let subsymt = check_membervarlist_item symt subsymt memberlist2 in
            (VarMap.add (name, Character) subsymt symt) , locmap
  | Itemdec (name, memberlist1) ->
      (*print_symboltable symt;*)
        if (VarMap.mem (name, Location) symt) ||
             (VarMap.mem (name, Character) symt) ||
             (VarMap.mem (name, Item) symt) ||
             (VarMap.mem (name, String) symt) ||
             (VarMap.mem (name, Integer) symt) then
            raise ( DupVar("duplicated identifier in item dec " ^ name))
        else
            let subsymt = VarMap.empty in
            let subsymt = check_membervarlist_intstr subsymt memberlist1 in
            (VarMap.add (name, Item) subsymt symt) , locmap
  | Locdec (name, memberlist1,memberlist2, memberlist3)->
      (*print_symboltable symt;*)
        if (VarMap.mem (name, Location) symt) ||
```

```
            (VarMap.mem (name, Character) symt) ||
            (VarMap.mem (name, Item) symt) ||
            (VarMap.mem (name, String) symt) ||
            (VarMap.mem (name, Integer) symt) then
          raise ( DupVar("duplicated identifier in location dec " ^ name))
        else
            let subsymt = VarMap.empty in
            let subsymt = check_membervarlist_intstr subsymt memberlist1 in
            let subsymt = check_membervarlist_item symt subsymt memberlist2 in
            let subsymt = check_membervarlist_chara symt subsymt memberlist3 in
            (VarMap.add (name, Location) subsymt symt), locmap
  | Startend (name, cond, logicstmt) ->
      (*print_symboltable symt;*)
        if not (VarMap.mem (name, Location) symt)  then
            raise ( NotFound("undefined variable in start stmt " ^ name))
        else
            ignore (check_expr symt cond);
            check_stmt symt logicstmt;
            symt, (StringMap.add name 1 locmap)



let rec check_program (symt,locmap) = function
    [] -> let match_loc tuple dc =
            let name = fst tuple in
            let t = snd tuple in
            if (t = Location) && (not (StringMap.mem name locmap)) then
              raise ( NotFound("No body definition for location " ^ name))
            else
              ()
        in
        VarMap.iter match_loc symt;
        symt
  | dec::tl -> check_program  (check_globaldec symt locmap dec) tl
```

```
(* checktype.mli by Xiao *)
type t =
      Integer
  | String
  | Character
  | Item
  | Location
  | Action (*hack for action name in choose stmt *)
  | Key    (*hack for key binded for an action *)
```

```
(* compile.ml by Everybody *)
open Ast
open Expression
open Declaration
open Action
open Selection
open Start
open Statement
open Check

module type COMPILE =
  sig
    exception CompileError of string
    val javacode :
      globaldec list -> ('a VarMap.t VarMap.t) -> string list * string list
    val stmt_to_java :
      ('a VarMap.t VarMap.t) -> string list * string list -> stmt
      -> string list * string list
    val global_dec_to_java :
      string list * string list -> globaldec
      -> ('a VarMap.t VarMap.t) -> string list * string list
  end

module Compile : COMPILE = struct

exception CompileError of string

let rec startend_stmt_check (expression:string) (statement:string list) =
  match statement with
    []->[]
  |hd::tl ->
    if (String.contains hd ';') then
      [hd] @ ["if (" ^expression ^")"; "endGame();"] @
        (startend_stmt_check expression tl)
      else [hd] @ (startend_stmt_check expression tl)

let rec actiondeclist_to_java list num = match list with
    [] -> []
  | hd::tail ->
      match hd with
        Unitaction(action, actionname, key) ->
          ["keysToActionName" ^ string_of_int num ^
           ".put(\"" ^ key ^ "\", \"" ^ action ^ "\"" ^ ");";
           "actionNameToOutput" ^ string_of_int num ^
           ".put(\"" ^ action ^ "\", \"" ^ actionname ^ "\");";
           "System.out.println(\"Type " ^ key ^
           " for " ^ actionname ^ "\");"] @ actiondeclist_to_java tail num

let rec prob_sum list = match list with
    [] -> 0
  | hd::tail ->
      match hd with Unitprob(i, stmt) -> i + (prob_sum tail)

let rec stmt_to_java tmap (playcode, startfns) stmt =
  match stmt with
    Ifelse (expr, stmt1, stmt2) ->
      let (expr_precode, expr_exp) =
          Expression.expr_to_java_boolean expr tmap in
      let (stmt1_playcode, stmt1_startfns) =
          stmt_to_java tmap ([], []) stmt1  in
      let (stmt2_playcode, stmt2_startfns) =
          stmt_to_java tmap ([], []) stmt2  in
      (playcode @ expr_precode @ ["if(" ^ expr_exp ^ ") {"] @
      stmt1_playcode @ ["}"; "else {"] @ stmt2_playcode @ ["}"],
      startfns @ stmt1_startfns @ stmt2_startfns)
  | Chwhen (actiondeclist, whenexprlist) ->
      let num = List.length(playcode) in
      let mapDecl = ["Map<String,String> keysToActionName" ^
                     string_of_int num ^ " = new HashMap<String, String>();";
                     "Map<String, String> actionNameToOutput" ^
                     string_of_int num ^
                     " = new HashMap<String, String>();"] in
```

```
      let actiondecs = ["System.out.println(\"CHOOSE AN ACTION:\");"]
                       @ actiondeclist_to_java actiondeclist num in
      let getinput = ["Scanner in" ^ string_of_int num ^
                      " = new Scanner(System.in);";
                      "String input" ^ string_of_int num ^
                      " = in" ^ string_of_int num ^ ".nextLine();";
                      "while(!keysToActionName" ^ string_of_int num ^
                      ".containsKey(input" ^ string_of_int num ^ ")) {";
                      "System.out.println(\"Invalid input, try again\");";
                      "input" ^ string_of_int num ^ " = in" ^
                      string_of_int num ^ ".nextLine();";
                      "}";
                      "System.out.println(\"You typed \" + input" ^
                      string_of_int num ^ ");";
                      "String action" ^ string_of_int num ^
                      " = keysToActionName" ^ string_of_int num ^
                      ".get(input" ^ string_of_int num ^ ");" ] in
      let whenexprs_playcode, whenexprs_startfns =
        whenexprs_to_java whenexprlist num tmap in
      (playcode @ mapDecl @ actiondecs @ getinput @
      whenexprs_playcode, startfns @ whenexprs_startfns)

  | Prob (probexprlist) ->
      let sum = prob_sum probexprlist in
      if sum != 100 then
        raise (CompileError "Probabilities did not sum to 100")
      else
        let randomcode = ["int num = r.nextInt(100);"] in
        let (probexprs_code, probexprs_startfns) =
          probexprs_to_java 0 probexprlist tmap in
      (playcode @ randomcode @ probexprs_code, startfns @ probexprs_startfns)

  | Kill (str) -> (playcode @ (Action.kill_to_java str), startfns)
  | Grab (str1, str2) ->
      (playcode @ (Action.grab_to_java str1 str2), startfns)
  | Drop (str1, str2) ->
      (playcode @ (Action.drop_to_java str1 str2), startfns)
  | Show (str1, str2) ->
      (playcode @ (Action.show_to_java str1 str2 tmap), startfns)
  | Hide (str1, str2) ->
      (playcode @ (Action.hide_to_java str1 str2 tmap), startfns)
  | Atomstmt (expr) ->
      (playcode @ ["dummy = (" ^(Expression.expr_to_java expr tmap)^");"],
      startfns)
  | Cmpdstmt (codeblock) ->
      let (blockcode, startfns) =
        List.fold_left (stmt_to_java tmap) ([], []) codeblock
      in
      (playcode @ ["{"] @ (blockcode) @ ["}"], startfns)


  | Nostmt (i) -> (playcode @ ["//Empty stmt"], startfns)
  | Print (str) ->
    (playcode @
    ["System.out.println(\"\"+ (" ^Expression.expr_to_java str tmap^"));"],
    startfns)


and whenexprs_to_java list num tmap = match list with
    [] -> ([], [])
  | hd :: tail ->
      let (hd_playcode, hd_startfns) = whenexpr_to_java hd num tmap in
      let (tail_playcode, tail_startfns) = whenexprs_to_java tail num tmap in
      (hd_playcode @ tail_playcode, hd_startfns @ tail_startfns)

and whenexpr_to_java whenexpr num tmap =
  match whenexpr with
    Unitwhen(action, stmt, loc) ->
      let (when_playcode, when_startfns) = stmt_to_java tmap ([], []) stmt in
      let nextcode = [loc ^ "();"] in
      (["if(action" ^ string_of_int num ^ ".equals(\"" ^ action ^   "\")) {"]
      @ when_playcode @ nextcode @ ["}"], when_startfns)
```

```
and probexprs_to_java start_num list tmap = match list with
    [] -> ([], [])
  | hd::tail ->
    let (hd_playcode, hd_startfns, curr_num) =
      probexpr_to_java hd start_num tmap in
    let (tail_playcode, tail_startfns) =
      probexprs_to_java curr_num tail tmap in
    (hd_playcode @ tail_playcode, hd_startfns @ tail_startfns)

and probexpr_to_java probexpr start_num tmap =
  match probexpr with
    Unitprob(i, stmt) ->
      let (prob_playcode, prob_startfns) = stmt_to_java tmap ([], []) stmt in
      (["if(num >= " ^ string_of_int start_num ^ " && num < "
      ^ string_of_int (start_num + i) ^ ") {"] @ prob_playcode @ ["}"],
      prob_startfns, start_num + i)

exception InvalidCode of string

(* TODO: FIX THIS *)
let global_dec_to_java (playcode, startfns) global_dec tmap =
  match global_dec with
    IntStrdec (pridec) ->
        let l = Declaration.intstrdec_to_java pridec tmap in
        (match l with
        []-> (playcode, startfns)
      | hd::hd2::tl -> (playcode @ tl, startfns @ [hd; hd2])
      | _::tl -> raise (InvalidCode("Invalid Code")))
  | Charadec (name, membervar1, membervar2) ->
      let l = Declaration.charadec_to_java name membervar1 membervar2 tmap in
        (match l with
        []-> (playcode, startfns)
      | hd::hd2::tl -> (playcode @ tl, startfns @ [hd; hd2])
      | _::tl -> raise (InvalidCode("Invalid Code")))
  | Itemdec (name, membervar) ->
      let l = Declaration.itemdec_to_java name membervar tmap in
      (match l with
        []-> (playcode, startfns)
      | hd::hd2::tl -> (playcode @ tl, startfns @ [hd; hd2])
      | _::tl -> raise (InvalidCode("Invalid Code")))
  | Locdec (name, membervar1, membervar2, membervar3) ->
      let l =
        Declaration.locdec_to_java name membervar1 membervar2 membervar3 tmap in
        (match l with
          []-> (playcode, startfns)
        | hd::hd2::tl -> (playcode @ tl, startfns @ [hd; hd2])
        | _::tl -> raise (InvalidCode("Invalid Code")))
  | Startend (name, expr, stmt) ->
      playcode @ ["//Location function call"; name ^ "();"],
      startfns @ ["//start funtion"; "public void " ^ name ^ "() {"] @
        (fst (Expression.expr_to_java_boolean expr tmap)) @
        ["currentLocation = \"" ^ name ^ "\";"]@
        ["if (" ^(snd (Expression.expr_to_java_boolean expr tmap)) ^")";
        "endGame();"] @
        ["while (!(" ^ (snd (Expression.expr_to_java_boolean expr tmap)) ^")){"] @
        (startend_stmt_check (snd (Expression.expr_to_java_boolean expr tmap))
        (fst (stmt_to_java tmap ([], []) stmt)) )
        @ (fst (Expression.expr_to_java_boolean expr tmap))@ ["}"  ; "}"]

let print_globaldec global_dec = match global_dec with
    IntStrdec (pridec) ->
      print_endline "pridec"
  | Charadec (name, membervar1, membervar2) ->
      print_endline ("char " ^ name)
  | Itemdec (name, membervar) ->
      print_endline ("item " ^ name)
  | Locdec (name, membervar1, membervar2, membervar3) ->
      print_endline ("loc " ^ name)
  | Startend (name, expr, stmt) -> print_endline ("Start "^name)

let rec javacode program symt = match program with
```

```
    [] -> ([], [])
  | hd::tl ->
    let tuple = javacode tl symt in
    (fst (global_dec_to_java ([], []) hd symt))@ (fst tuple),
    (snd (global_dec_to_java ([], []) hd symt))@ (snd tuple)
end
```

```
(* declaration.ml by Pri: Danny, Sec: Ernesto, Morgan *)
open Ast
open Expression
open Check

module type DECLARATION = sig
   val intstrdec_to_java : pridec -> ('a VarMap.t VarMap.t) -> string list
   val itemdec_to_java : string -> membervarlist ->
                         ('a VarMap.t VarMap.t) -> string list
   val charadec_to_java : string -> membervarlist -> membervarlist ->
                          ('a VarMap.t VarMap.t) -> string list
   val locdec_to_java : string -> membervarlist -> membervarlist ->
                        membervarlist -> ('a VarMap.t VarMap.t)  -> string list
end


module Declaration : DECLARATION = struct

let attr_to_java tmap (var:string) (attr:membervar) : string list =
  match attr with
    Primember(pridec) ->
      (match pridec with
        Strdec (str) ->
          [var ^ ".addStrAttr(\"" ^ str ^ "\", \"\");";
          "types.put(\"" ^ str ^ "\", Type.STRING);"]
      | Intdec (str) ->
          [var ^ ".addIntAttr(\"" ^ str ^ "\", 0);";
          "types.put(\"" ^ str ^ "\", Type.INT);"]
      | Strdecinit (str, expr) ->
          [var ^ ".addStrAttr(\"" ^ str ^ "\"," ^
          (Expression.expr_to_java expr tmap) ^ ");";
          "types.put(\"" ^ str ^ "\", Type.STRING);"]
      | Intdecinit (str, expr) ->
          [ var ^ ".addIntAttr(\"" ^ str ^ "\", " ^
          (Expression.expr_to_java expr tmap) ^ ");";
          "types.put(\"" ^ str ^ "\", Type.INT);"])
  | Varref(str) -> ["//OOPS THIS WAS BAD!"]

let rec attrlist_to_java var attrlist tmap =
  match attrlist with
    [] -> []
  | hd::tl -> (attr_to_java tmap var hd) @ (attrlist_to_java var tl tmap)

let itemdec_to_java str attrlist tmap =
  ["//itemdec"; "Item " ^ str ^ " = new Item();";
  "items.put(\"" ^ str ^ "\"," ^ str ^ ");";
  "types.put(\"" ^ str ^ "\", Type.ITEM);"] @
  (attrlist_to_java str attrlist tmap)

let item_to_java (var:string) (item:membervar) : string list =
  match item with
    Varref(str) -> [var ^".addItem(\"" ^str^ "\");"]
  | Primember(pridec) ->
      ["//This was bad. Not a reference to a complex variable"]

let rec itemlist_to_java var itemlist =
  match itemlist with
    [] -> []
  | hd::tl -> (item_to_java var hd) @ (itemlist_to_java var tl)

let charadec_to_java str attrlist itemlist tmap =
  ["//charadec"; "Character " ^ str ^ " = new Character();";
  "characters.put(\"" ^ str ^ "\"," ^ str ^ ");";
  "types.put(\"" ^ str ^ "\", Type.CHARACTER);"]@
  (attrlist_to_java str attrlist tmap) @ (itemlist_to_java str itemlist)

let character_to_java (var:string) (character:membervar) : string list =
  match character with
    Varref(str) -> [var ^".showCharacter(\"" ^str^ "\");"]
  | Primember(pridec) ->
      ["//This was bad. Not a reference to a complex variable"]
```

```
let rec characterlist_to_java var characterlist =
  match characterlist with
    [] -> []
  | hd::tl -> (character_to_java var hd) @ (characterlist_to_java var tl)

let locdec_to_java str attrlist itemlist charlist tmap =
  ["//locdec"; "Location " ^ str ^ " = new Location();";
  "locations.put(\"" ^ str ^ "\"," ^ str ^ ");";
  "types.put(\"" ^ str ^ "\", Type.LOCATION);"]@
  (attrlist_to_java str attrlist tmap) @
  (itemlist_to_java str itemlist) @
  (characterlist_to_java str charlist)

let intstrdec_to_java pridec tmap = match pridec with
    Strdec(str) -> ["//strdec"; "String " ^ str ^ " = \"\";"]
  | Intdec(str) -> ["//intdec"; "int " ^ str ^ ";"]
  | Strdecinit(str, expr) ->
      ["//strdecinit"; "String " ^ str ^ ";";
      str ^ " = " ^ (Expression.expr_to_java expr tmap) ^ ";"]
  | Intdecinit(str, expr) ->
      ["//intdecinit"; "int " ^ str ^ ";";
      str ^ " = " ^ (Expression.expr_to_java expr tmap) ^ ";"]
end
```

```
(* expression.ml by Pri: Danny, Sec: Morgan, Ernesto *)
open Ast
open Checktype
open Check


module type EXPRESSION =
   sig
    val expr_to_java : expr -> ('a VarMap.t VarMap.t) -> string
    val expr_to_java_boolean :
      expr -> ('a VarMap.t VarMap.t) -> string list * string
      (* string list contains statements that need
      to happen before the string condition is checked *)
    val next_type_to_string : Checktype.t -> string
    val check_type_to_string : string -> ('a VarMap.t VarMap.t) -> string
   end

module Expression : EXPRESSION = struct

exception InvalidComparison of string

let next_type_to_string = function
      Integer -> "int"
  | String -> "string"
  | Character -> "character"
  | Item -> "item"
  | Location -> "location"
  | Action -> "action"
  | Key -> "key"

let check_type_to_string name  = function
  symt ->
    if (not (VarMap.mem (name,String) symt)) &&
       (not (VarMap.mem (name,Integer) symt)) &&
       (not (VarMap.mem (name,Location) symt)) &&
       (not (VarMap.mem (name,Character) symt)) &&
       (not (VarMap.mem (name,Item) symt)) then
      raise ( NotFound("undefined variable " ^ name))
    else if (VarMap.mem (name,String) symt) then "String"
      else if (VarMap.mem (name,Integer) symt) then "Int"
      else if (VarMap.mem (name,Location) symt) then "Location"
      else if(VarMap.mem (name,Character) symt) then "Character"
    else "Item"

let rec expr_to_java exp tmap =
  match exp with
    Binop (exp1, op, exp2) ->
      if op == Add then
        "(" ^ (expr_to_java exp1 tmap) ^ " + " ^ (expr_to_java exp2 tmap) ^ ")"
      else if op == Sub then
        "(" ^ (expr_to_java exp1 tmap) ^ " - " ^ (expr_to_java exp2 tmap) ^ ")"
      else if op == Mul then
        "(" ^ (expr_to_java exp1 tmap) ^ " * " ^ (expr_to_java exp2 tmap) ^ ")"
      else if op == Div then
        "(" ^ (expr_to_java exp1 tmap) ^ " / " ^ (expr_to_java exp2 tmap) ^ ")"
      else if op == Or then
        "boolToInt(isTrue(" ^ (expr_to_java exp1 tmap) ^
        ") || isTrue(" ^ (expr_to_java exp2 tmap) ^ "))"
      else if op == And then
        "boolToInt(isTrue(" ^ (expr_to_java exp1 tmap) ^
        ") && isTrue(" ^ (expr_to_java exp2 tmap) ^ "))"
      else if op == Eq then
        let t = check_expr tmap exp1 in
        (match t with
          String ->
            "boolToInt(" ^ (expr_to_java exp1 tmap) ^
            ".equals(" ^ (expr_to_java exp2 tmap) ^ "))"
        | Integer ->
            "boolToInt(" ^ (expr_to_java exp1 tmap) ^
            " == " ^ (expr_to_java exp2 tmap) ^ ")"
        | _ -> raise (InvalidComparison("Invalid Comparison")))
      else if op == Lt then
```

```
        "boolToInt (" ^ (expr_to_java exp1 tmap) ^
        " < " ^ (expr_to_java exp2 tmap) ^ ")"
      else if op == Gt then
        "boolToInt (" ^ (expr_to_java exp1 tmap) ^
        " > " ^ (expr_to_java exp2 tmap) ^ ")"
      else if op == Leq then
        "boolToInt (" ^ (expr_to_java exp1 tmap) ^
        " <= " ^ (expr_to_java exp2 tmap) ^ ")"
      else if op == Geq then
        "boolToInt (" ^ (expr_to_java exp1 tmap) ^
        " >= " ^ (expr_to_java exp2 tmap) ^ ")"
      else if op == Neq then
        "boolToInt (" ^ (expr_to_java exp1 tmap) ^
        " != " ^ (expr_to_java exp2 tmap) ^ ")"
      else raise (InvalidComparison("Invalid Comparison"))
  | Asn (id, exp) ->
      let t = check_id tmap id in
      (match id with
        Var(str) -> str ^ " = " ^ (expr_to_java exp tmap)
      | Has(name, subname) ->
          "entitySet" ^ (String.capitalize (next_type_to_string t)) ^
          "(\"" ^ name ^ "\", Type." ^
          (String.uppercase (check_type_to_string name tmap)) ^
          ", \"" ^ subname ^ "\", " ^ (expr_to_java exp tmap) ^ ")")
  | Lit (i) -> "(" ^ (string_of_int i) ^ ")"
  | LitS (str) -> "\"" ^ str ^ "\""
  | Exists (str1, str2) ->
      let t1 = check_id tmap (Var(str1)) in
      let t2 = check_id tmap (Var(str2)) in
      (match t2 with
        Item ->
          "entityExistsItem(\"" ^ str1 ^ "\", Type." ^
          (String.uppercase (next_type_to_string t1)) ^
          ", \"" ^ str2 ^ "\")"
      | Character ->
          "entityExistsCharacter(\"" ^ str1 ^ "\", Type." ^
          (String.uppercase (next_type_to_string t1))
          ^ ", \"" ^ str2 ^ "\")"
      | _ -> raise (InvalidComparison("Invalid Comparison")))
  | Ident (id) ->
      let t = check_id tmap id in
      (match id with
        Var(name) -> name
      | Has(name, subname) ->
          "entityHas" ^ (String.capitalize (next_type_to_string t)) ^
          "(\"" ^ name ^ "\", Type." ^
          (String.uppercase (check_type_to_string name tmap))
          ^ ", \"" ^ subname ^ "\")")
  | Neg (exp) -> "(-" ^ (expr_to_java exp tmap) ^ ")"
  | Not (exp) -> "boolToInt(!" ^ "isTrue(" ^ (expr_to_java exp tmap) ^ ") )"


let rec expr_to_java_boolean exp tmap =
  match exp with
      Binop (exp1, op, exp2) ->
        if op == Add then
          ([], "(" ^ (expr_to_java exp1 tmap) ^ " + "
          ^ (expr_to_java exp2 tmap) ^ ") != 0" )
        else if op == Sub then
          ([], "(" ^ (expr_to_java exp1 tmap) ^ " - " ^
          (expr_to_java exp2 tmap) ^ ") != 0")
        else if op == Mul then
          ([], "(" ^ (expr_to_java exp1 tmap) ^ " * "
          ^ (expr_to_java exp2 tmap) ^ ") != 0")
        else if op == Div then
          ([], "(" ^ (expr_to_java exp1 tmap) ^ " / "
          ^ (expr_to_java exp2 tmap) ^ ") != 0")
        else if op == Eq then
          let t = check_expr tmap exp1 in
        (match t with
          String ->
            ([], (expr_to_java exp1 tmap) ^
```

```
            ".equals(" ^ (expr_to_java exp2 tmap) ^ ")")
        | Integer ->
            ([], (expr_to_java exp1 tmap) ^ " == " ^ (expr_to_java exp2 tmap))
        | _ -> raise (InvalidComparison("Invalid Comparison")))
        else if op == Lt then
          ([], "(" ^ (expr_to_java exp1 tmap) ^
          " < " ^ (expr_to_java exp2 tmap) ^ ")")
      else if op == Gt then
        ([], "(" ^ (expr_to_java exp1 tmap) ^
        " > " ^ (expr_to_java exp2 tmap) ^ ")")
        else if op == Neq then
          ([], "(" ^ (expr_to_java exp1 tmap) ^
          " != " ^ (expr_to_java exp2 tmap) ^ ")")
        else if op == Leq then
          ([], "(" ^ (expr_to_java exp1 tmap) ^
          " <= " ^ (expr_to_java exp2 tmap) ^ ")")
      else if op == Geq then
        ([], "(" ^ (expr_to_java exp1 tmap) ^
        " >= " ^ (expr_to_java exp2 tmap) ^ ")")
        else if op == Or then
          (fst (expr_to_java_boolean exp1 tmap) @
          fst (expr_to_java_boolean exp2 tmap),
          (snd (expr_to_java_boolean exp1 tmap)) ^
          " || " ^ (snd (expr_to_java_boolean exp2 tmap)))
      else if op == And then
        (fst (expr_to_java_boolean exp1 tmap) @
        fst (expr_to_java_boolean exp2 tmap),
        (snd (expr_to_java_boolean exp1 tmap)) ^
        " && " ^ (snd (expr_to_java_boolean exp2 tmap)))
        else ([], "false")
  | Asn (id, exp) ->
      let t = check_id tmap id in
      (match id with
        Var(str) ->
          ([str ^ " = " ^ (expr_to_java exp tmap)],
          "(" ^ (expr_to_java exp tmap) ^ ") != 0 ")
      | Has(name, subname) ->
          (["entitySet" ^ (String.capitalize (next_type_to_string t)) ^
          "(\"" ^ name ^ "\", Type." ^
          (String.uppercase (check_type_to_string name tmap)) ^
          ", \"" ^ subname ^ "\", " ^ (expr_to_java exp tmap) ^ ")"],
          "(" ^ (expr_to_java exp tmap) ^ ") != 0 "))
  | Lit (i) -> ([], "isTrue(" ^ (string_of_int i) ^ ")")
  | LitS (str) -> ([], "isTrue(" ^ str ^ ")")
  | Exists (str1, str2) ->
      let t1 = check_id tmap (Var(str1)) in
      let t2 = check_id tmap (Var(str2)) in
      (match t2 with
        Item ->
          ([], "isTrue(entityExistsItem(\"" ^ str1 ^
          "\", Type." ^ (String.uppercase (next_type_to_string t1)) ^
          ", \"" ^ str2 ^ "\"))")
      | Character ->
        ([], "isTrue(entityExistsCharacter(\"" ^ str1 ^
        "\", Type." ^ (String.uppercase (next_type_to_string t1)) ^
        ", \"" ^ str2 ^ "\"))")
      | _ -> raise (InvalidComparison("Invalid Comparison")))
  | Ident (id) -> ([], "isTrue(" ^ (expr_to_java exp tmap) ^ ")")
  | Neg (exp) -> ([], (expr_to_java exp tmap) ^ " != 0 ")
  | Not (exp) -> ([], "!isTrue(" ^ (expr_to_java exp tmap) ^ ")")

end
```

```
#Makefile
TARFILES = Makefile scanner.mll parser.mly ast.mli check.ml checktype.mli expression.ml
declaration.ml

action.ml selection.ml start.ml statement.ml compile.ml next.ml

OBJS = parser.cmo scanner.cmo check.cmo expression.cmo declaration.cmo action.cmo
selection.cmo

start.cmo statement.cmo compile.cmo next.cmo
LIBPATH = -I +sdl

next : $(OBJS)
	ocamlc -o next $(OBJS)

scanner.ml : scanner.mll
	ocamllex scanner.mll

parser.ml parser.mli : parser.mly
	ocamlyacc -v parser.mly

%.cmo : %.ml
	ocamlc -c $(LIBPATH) $<

%.cmi : %.mli
	ocamlc -c $(LIBPATH) $<

next.tar.gz : $(TARFILES)
	cd .. && tar zcf next/next.tar.gz $(TARFILES:%=next/%)

.PHONY : clean
clean :
	rm -f next parser.ml parser.mli scanner.ml *.cmo *.cmi *.class

# Generated by ocamldep *.ml *.mli
action.cmo:
action.cmx:
check.cmo: checktype.cmi ast.cmi
check.cmx: checktype.cmi ast.cmi
compile.cmo: statement.cmo start.cmo selection.cmo expression.cmo \
    declaration.cmo ast.cmi action.cmo
compile.cmx: statement.cmx start.cmx selection.cmx expression.cmx \
    declaration.cmx ast.cmi action.cmx
declaration.cmo: expression.cmo ast.cmi
declaration.cmx: expression.cmx ast.cmi
expression.cmo: ast.cmi
expression.cmx: ast.cmi
next.cmo: scanner.cmo parser.cmi compile.cmo check.cmo ast.cmi
next.cmx: scanner.cmx parser.cmx compile.cmx check.cmx ast.cmi
parser.cmo: ast.cmi parser.cmi
parser.cmx: ast.cmi parser.cmi
scanner.cmo: parser.cmi
scanner.cmx: parser.cmx
selection.cmo: ast.cmi
selection.cmx: ast.cmi
start.cmo: ast.cmi
start.cmx: ast.cmi
statement.cmo: ast.cmi
statement.cmx: ast.cmi
ast.cmi:
checktype.cmi:
parser.cmi: ast.cmi
```

```
(* next.ml by Pri: Morgan, Sec: Everybody *)
open Ast
open Compile
open Check

let java_of_prog program symt =
let (playcode, startfns) = Compile.javacode program symt in
"
import java.util.*;

public class Next {
   enum Type {INT, STRING, CHARACTER, ITEM, LOCATION}

   static Random r = new Random();
   Object dummy;
   String currentLocation;
   Map<String, Location> locations = new HashMap<String, Location>();
   Map<String, Character> characters = new HashMap<String, Character>();
   Map<String, Item> items = new HashMap<String, Item>();
   Map<String, Type> types = new HashMap<String, Type>();

   public static void main(String[] args) {
    (new Next()).play();
   }

   public int boolToInt(boolean value) {
        if(value) {
            return 1;
        }
        else {
            return 0;
        }
   }

   public String entitySetString(String key1, Type type1, String key2, String value) {
    boolean valueSet = false;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  loc.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.ITEM) {
            Item item = items.get(key1);
            if(item != null) {
                  item.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }

      if(!valueSet) {
            throw new RuntimeException();
      }

      return value;
   }

   public int entitySetInt(String key1, Type type1, String key2, int value) {
      boolean foundReturnValue = false;

      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
            loc.intAttrs.put(key2, value);
```

```
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.CHARACTER) {
          Character character = characters.get(key1);
          if(character != null) {
                character.intAttrs.put(key2, value);
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.ITEM) {
          Item item = items.get(key1);
          if(item != null) {
                item.intAttrs.put(key2, value);
                foundReturnValue = true;
          }
    }

    if(foundReturnValue == false) {
          throw new RuntimeException();
    }

    return value;
}

    public boolean isTrue(Object object) {
        if(object instanceof String) {
           if(((String)object).isEmpty()) {
               return false;
           }
        }
        else if(object instanceof Integer) {
           if((Integer)object == 0) {
               return false;
           }
        }
        else {
            if(object == null) {
                return false;
            }
        }

        return true;
    }

  public void killFunction(String varName){
    if (characters.containsKey(varName)){
          characters.remove(varName);
          for (String key: locations.keySet()){
                      locations.get(key).hideCharacter(varName);
          }
    }
    else if (items.containsKey(varName)){
          items.remove(varName);
          for (String key:locations.keySet()){
                locations.get(key).removeItem(varName);
                }
          for (String key : characters.keySet()){
                characters.get(key).removeItem(varName);
          }
    }
  }

  public int entityHasInt(String key1, Type type1, String key2) {
    int returnValue = 0;
    boolean foundReturnValue = false;

    if(type1 == Type.LOCATION) {
          Location loc = locations.get(key1);
          if(loc != null) {
                returnValue = loc.intAttrs.get(key2);
                foundReturnValue = true;
```

```
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			returnValue = character.intAttrs.get(key2);
			foundReturnValue = true;
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.intAttrs.get(key2);
			foundReturnValue = true;
		}
	}

	if(foundReturnValue == false) {
		throw new RuntimeException();
	}

	return returnValue;
}

public String entityHasString(String key1, Type type1, String key2) {
	String returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			returnValue = loc.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			returnValue = character.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.strAttrs.get(key2);
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Item entityHasItem(String key1, Type type1, String key2) {
	Item returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			if(loc.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			if(character.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}

	if(returnValue == null) {
```

```
            throw new RuntimeException();
      }

      return returnValue;
   }

   public Character entityHasCharacter(String key1, Type type1, String key2) {
      Character returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            throw new RuntimeException();
      }

      return returnValue;
   }

   public int entityExistsItem(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  if(character.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
   }

   public int entityExistsCharacter(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
   }

  public void endGame() {
      System.exit(0);
  }
```

```
   public void play() {
"
 ^ (String.concat "\n" playcode) ^ "
   endGame();
   } \n"

 ^ (String.concat "\n" startfns) ^ "
}

abstract class Entity {

   Map<String, Integer> intAttrs = new HashMap<String, Integer>();
   Map<String, String> strAttrs = new HashMap<String, String>();

   public void addIntAttr(String name, int value) {
      intAttrs.put(name, value);
   }

   public void addStrAttr(String name, String value) {
      strAttrs.put(name, value);
   }
}

class Location extends Entity {
   Set<String> characters = new HashSet<String>();
   Set<String> items = new HashSet<String>();

   public void addItem(String name, Map<String, Item> itemses) {
      if(itemses.containsKey(name))
                  items.add(name);
      else
            System.out.println(\"Error: The item you attempted to add no longer
exists\");
   }

   public void addItem(String name){
      items.add(name);
      }

   public void removeItem(String name) {
      items.remove(name);
   }

   public void showCharacter(String name, Map<String, Character> characterses) {
      if(characterses.containsKey(name))
      characters.add(name);
      else
            System.out.println(\"Error: The character you attempted to use no longer
exists\");
   }
   public void showCharacter(String name){

      characters.add(name);
      }

   public void hideCharacter(String name) {
      characters.remove(name);
   }
}

class Character extends Entity {
   Set<String> items = new HashSet<String>();

   public void addItem(String name, String locationNow, Map<String, Location> locations){

      if(locations.get(locationNow).items.contains(name)){
            locations.get(locationNow).removeItem(name);
            items.add(name);
      }
      else
            System.out.println(\"Error: The item you attempted to grab is not in this
location\");
```

```
}

public void removeItem(String name, String locationNow, Map<String, Location> locations)
{
            if (items.contains(name)){
            items.remove(name);
            locations.get(locationNow).addItem(name);
   }
      else
            System.out.println(\"Error: The character does not have the item you
attempted to

drop\");
}

   public void addItem(String name) {
      items.add(name);
   }

   public void removeItem(String name) {
      items.remove(name);
   }
}

class Item extends Entity {
}"

let _ =
  let lexbuf = Lexing.from_channel stdin in
  let program = Parser.program Scanner.token lexbuf in
  let symt = check_program (VarMap.empty, StringMap.empty) program in
  let java = java_of_prog program symt in
  print_endline java
```

```
/* Parser.mly by Xiao */
%{ open Ast %}


%token PLUS MINUS TIMES DIVIDE LESSTHAN GREATERTHAN EQUAL NEQUAL LOGICAND LOGICOR ASSIGN
LOGICNOT

LEQTHAN GEQTHAN
%token COMMA SEMICOLON DOT
%token LBRACKET RBRACKET LPAREN RPAREN RPROBBLOCK LPROBBLOCK
%token IF THEN ELSE START END PROB WHEN NEXT CHOOSE KILL GRAB HIDE EXISTS DROP SHOW
%token CHARACTER LOCATION ACTION OUTPUT ITEM INT STRING
%token EOF
%token <int> LITERAL
%token <string> VARIABLE
%token <string> STRINGLIT


%nonassoc IF THEN ELSE START END PROB
%left SEMICOLON
%nonassoc OUTPUT
%left COMMA
%right ASSIGN
%left LOGICOR
%left LOGICAND
%left LOGICNOT
%left EQUAL NEQUAL
%left LESSTHAN GREATERTHAN LEQTHAN GEQTHAN
%left PLUS MINUS
%left TIMES DIVIDE
%left NEG
%nonassoc EXISTS
%left DOT
%nonassoc LPAREN RPAREN


%start program
%type < Ast.globaldecs> program
%type < Ast.globaldecs> file
%type < Ast.block> block
%type < Ast.expr> expr
%type < Ast.stmt> stmt
%type < Ast.pridec> pridec
%type < Ast.membervar> membervar
%type < Ast.probexpr> probexpr
%type < Ast.globaldec> globaldec

%%
program:
  file EOF {$1}
;

file:
            {[]}
| globaldecs {List.rev $1}
;

stmt:
  IF expr THEN stmt ELSE stmt {Ifelse($2,$4,$6)}
| KILL VARIABLE SEMICOLON                       {Kill($2)}
| GRAB VARIABLE DOT VARIABLE SEMICOLON                      {Grab($2,$4)}
| DROP VARIABLE DOT VARIABLE SEMICOLON                      {Drop($2,$4)}
| HIDE VARIABLE DOT VARIABLE SEMICOLON                      {Hide($2,$4)}
| SHOW VARIABLE DOT VARIABLE SEMICOLON                      {Show($2,$4)}
| RPROBBLOCK probexprlist LPROBBLOCK {Prob(List.rev $2)}
| expr SEMICOLON {Atomstmt ($1) }
| LBRACKET block RBRACKET {Cmpdstmt (List.rev $2) }
| LBRACKET RBRACKET { Nostmt (0) }
| SEMICOLON { Nostmt (0) }
| CHOOSE actiondeclist LBRACKET whenexprlist RBRACKET {Chwhen (List.rev $2, List.rev $4)}
| OUTPUT expr SEMICOLON { Print($2)}
;
```

```
globaldec:
  pridec SEMICOLON {IntStrdec($1)}
| CHARACTER VARIABLE LBRACKET LPAREN membervarlist RPAREN COMMA LPAREN membervarlist
RPAREN RBRACKET

{Charadec($2, List.rev $5, List.rev $9)}
| LOCATION VARIABLE LBRACKET LPAREN membervarlist RPAREN COMMA LPAREN membervarlist
RPAREN COMMA

LPAREN membervarlist RPAREN RBRACKET {Locdec($2,List.rev $5,List.rev $9, List.rev $13)}
| ITEM VARIABLE LBRACKET LPAREN membervarlist RPAREN RBRACKET {Itemdec($2, List.rev $5)}
| START VARIABLE END LPAREN expr RPAREN stmt {Startend ($2, $5, $7)}
;

globaldecs:
 globaldec {[$1]}
| globaldecs globaldec {$2::$1}
;

block:
  stmt { [$1] }
| block stmt { $2::$1 }
;

expr:
  expr PLUS expr               { Binop($1, Add, $3) }
| LOGICNOT expr                      { Not ($2)}
| expr MINUS  expr                   { Binop($1, Sub, $3) }
| expr TIMES  expr                   { Binop($1, Mul, $3) }
| expr DIVIDE expr                   { Binop($1, Div, $3) }
| expr LESSTHAN expr           { Binop($1, Lt, $3) }
| expr GREATERTHAN expr        { Binop($1, Gt ,$3) }
| expr LEQTHAN expr                  { Binop($1, Leq, $3)}
| expr GEQTHAN expr                  { Binop($1, Geq, $3)}
| expr EQUAL expr                    { Binop($1, Eq ,$3) }
| expr NEQUAL expr                   { Binop($1, Neq ,$3) }
| expr LOGICAND expr           { Binop($1, And ,$3) }
| expr LOGICOR expr            { Binop($1, Or ,$3) }
| LPAREN expr RPAREN         { $2 }
| id ASSIGN expr               { Asn($1, $3) }
| LITERAL                            { Lit($1) }
| STRINGLIT                          { LitS($1) }
| EXISTS VARIABLE DOT VARIABLE       { Exists($2,$4)}
| MINUS expr %prec NEG         { Neg($2)}
| id                                 { Ident($1)}
;

id:
  VARIABLE DOT VARIABLE        { Has($1,$3)}
| VARIABLE                           { Var($1)}

membervarlist:
  {[]}
|  membervar {[$1]}
| membervarlist COMMA membervar {$3::$1}
;

membervar:
  pridec {Primember($1)}
| VARIABLE {Varref($1)}
;

pridec:
  STRING VARIABLE ASSIGN expr {Strdecinit($2,$4)}
| INT VARIABLE ASSIGN expr {Intdecinit($2,$4)}
| STRING VARIABLE {Strdec($2)}
| INT VARIABLE {Intdec($2)}
;


probexprlist:
```

```
  probexpr {[$1]}
| probexprlist probexpr {$2::$1}
;

probexpr:
  PROB LITERAL stmt { Unitprob ($2,$3)}
;

actiondeclist:
  actiondec {[$1]}
| actiondeclist actiondec {$2::$1}
;

actiondec:
  LPAREN VARIABLE COMMA STRINGLIT COMMA STRINGLIT RPAREN { Unitaction ($2,$4,$6)}
;

whenexprlist:
  whenexpr  {[$1]}
| whenexprlist whenexpr {$2::$1}
;

whenexpr:
  WHEN VARIABLE stmt NEXT VARIABLE { Unitwhen($2,$3,$5)}
;
```

```
(* scanner.mll by Ernesto *)
{ open Parser }


rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf }
| "/*" {comment lexbuf}
| '"' [^ '"']* '"' as s { STRINGLIT( String.sub s 1 (String.length s - 2)) }
| '+' { PLUS }
| '-' { MINUS }
| '*' { TIMES }
| '/' { DIVIDE }
| '<' { LESSTHAN }
| "<=" {LEQTHAN}
| ">=" {GEQTHAN}
| '>' { GREATERTHAN }
| "==" { EQUAL }
| "!=" { NEQUAL }
| "and" {LOGICAND}
| "or" {LOGICOR}
| ';' {SEMICOLON}
| '=' {ASSIGN}
| "if" {IF}
| "then" {THEN}
| "else" {ELSE}
| '{' {LBRACKET}
| '}' {RBRACKET}
| '(' {LPAREN}
| ')' {RPAREN}
| ',' {COMMA}
| "output" {OUTPUT}
| "not" {LOGICNOT}
|'.' {DOT}
|"start" {START}
|"end" {END}
|"prob" {PROB}
|"[?" {RPROBBLOCK}
|"?]" {LPROBBLOCK}
|"when" {WHEN}
|"next" {NEXT}
|"choose" {CHOOSE}
|"kill" {KILL}
|"grab" {GRAB}
|"drop" {DROP}
|"show" {SHOW}
|"hide" {HIDE}
|"exists" {EXISTS}
|"character" {CHARACTER}
|"location" {LOCATION}
|"item" {ITEM}
|"int" {INT}
|"string" {STRING}
| ['0'-'9']+ as lit { LITERAL(int_of_string lit) }
| ['A'-'Z' 'a'-'z'] ['A'-'Z' 'a'-'z' '0'-'9' '_']* as var { VARIABLE ("_" ^ var) }
| eof { EOF }

and comment = parse
| "*/" {token lexbuf }
| _ {comment lexbuf}
```

Sample Program Files

All sample .next programs are followed by their Next.java files they translate to.

```
/*LRMSample.next by Ernesto Arreguin */

int count;

item the_greatest_sword_ever {(int damage = 100000000)}

character xiaowei_the_greatest_man_ever {(int life = 100000000,
int level=99999, string haha="hahahahaha"), (the_greatest_sword_ever)}

location where_is_this_place {(int sizex = 10000, int sizey=9283), (),
(xiaowei_the_greatest_man_ever)}

start where_is_this_place end (xiaowei_the_greatest_man_ever.life < 0) {
      choose (attack,"hia!","a") (up,"up","u") (fin, "end", "f") {
            when attack
            {
                  xiaowei_the_greatest_man_ever.life+1;

            } next where_is_this_place

            when up
            {
                  [? prob 40 {
                        count = count-1;
                        output "count:";
                        output count;
                  }
                  prob 60 {
                        count = count+1;
                        output "count:";
                        output count;
                  }
                  ?]
            } next where_is_this_place

            when fin
            {
                  xiaowei_the_greatest_man_ever.life = -1;
            } next where_is_this_place


      }
}
```

```
//Generated Next.java file for LRMSample.Next
import java.util.*;

public class Next {
   enum Type {INT, STRING, CHARACTER, ITEM, LOCATION}

   static Random r = new Random();
   Object dummy;
   String currentLocation;
   Map<String, Location> locations = new HashMap<String, Location>();
   Map<String, Character> characters = new HashMap<String, Character>();
   Map<String, Item> items = new HashMap<String, Item>();
   Map<String, Type> types = new HashMap<String, Type>();

   public static void main(String[] args) {
    (new Next()).play();
   }

   public int boolToInt(boolean value) {
        if(value) {
            return 1;
        }
        else {
            return 0;
        }
   }

   public String entitySetString(String key1, Type type1, String key2, String value) {
    boolean valueSet = false;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  loc.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.ITEM) {
            Item item = items.get(key1);
            if(item != null) {
                  item.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }

      if(!valueSet) {
            throw new RuntimeException();
      }

      return value;
   }

   public int entitySetInt(String key1, Type type1, String key2, int value) {
      boolean foundReturnValue = false;

      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
            loc.intAttrs.put(key2, value);
                  foundReturnValue = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.intAttrs.put(key2, value);
```

```
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.ITEM) {
          Item item = items.get(key1);
          if(item != null) {
                item.intAttrs.put(key2, value);
                foundReturnValue = true;
          }
    }

    if(foundReturnValue == false) {
          throw new RuntimeException();
    }

    return value;
}

    public boolean isTrue(Object object) {
        if(object instanceof String) {
           if(((String)object).isEmpty()) {
               return false;
           }
        }
        else if(object instanceof Integer) {
           if((Integer)object == 0) {
               return false;
           }
        }
        else {
            if(object == null) {
                return false;
            }
        }

        return true;
    }

  public void killFunction(String varName){
    if (characters.containsKey(varName)){
          characters.remove(varName);
          for (String key: locations.keySet()){
                      locations.get(key).hideCharacter(varName);
          }
    }
    else if (items.containsKey(varName)){
          items.remove(varName);
          for (String key:locations.keySet()){
                locations.get(key).removeItem(varName);
                }
          for (String key : characters.keySet()){
                characters.get(key).removeItem(varName);
          }
    }
  }

  public int entityHasInt(String key1, Type type1, String key2) {
    int returnValue = 0;
    boolean foundReturnValue = false;

    if(type1 == Type.LOCATION) {
          Location loc = locations.get(key1);
          if(loc != null) {
                returnValue = loc.intAttrs.get(key2);
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.CHARACTER) {
          Character character = characters.get(key1);
          if(character != null) {
                returnValue = character.intAttrs.get(key2);
                foundReturnValue = true;
```

```
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.intAttrs.get(key2);
			foundReturnValue = true;
		}
	}

	if(foundReturnValue == false) {
		throw new RuntimeException();
	}

	return returnValue;
}

public String entityHasString(String key1, Type type1, String key2) {
	String returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			returnValue = loc.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			returnValue = character.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.strAttrs.get(key2);
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Item entityHasItem(String key1, Type type1, String key2) {
	Item returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			if(loc.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			if(character.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Character entityHasCharacter(String key1, Type type1, String key2) {
```

```
      Character returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            throw new RuntimeException();
      }

      return returnValue;
    }

    public int entityExistsItem(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  if(character.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
    }

    public int entityExistsCharacter(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
    }

   public void endGame() {
      System.exit(0);
   }

   public void play() {
items.put("_the_greatest_sword_ever",_the_greatest_sword_ever);
types.put("_the_greatest_sword_ever", Type.ITEM);
_the_greatest_sword_ever.addIntAttr("_damage", (100000000));
types.put("_damage", Type.INT);
characters.put("_xiaowei_the_greatest_man_ever",_xiaowei_the_greatest_man_ever);
types.put("_xiaowei_the_greatest_man_ever", Type.CHARACTER);
```

```
_xiaowei_the_greatest_man_ever.addIntAttr("_life", (100000000));
types.put("_life", Type.INT);
_xiaowei_the_greatest_man_ever.addIntAttr("_level", (99999));
types.put("_level", Type.INT);
_xiaowei_the_greatest_man_ever.addStrAttr("_haha","hahahahaha");
types.put("_haha", Type.STRING);
_xiaowei_the_greatest_man_ever.addItem("_the_greatest_sword_ever");
locations.put("_where_is_this_place",_where_is_this_place);
types.put("_where_is_this_place", Type.LOCATION);
_where_is_this_place.addIntAttr("_sizex", (10000));
types.put("_sizex", Type.INT);
_where_is_this_place.addIntAttr("_sizey", (9283));
types.put("_sizey", Type.INT);
_where_is_this_place.showCharacter("_xiaowei_the_greatest_man_ever");
//Location function call
_where_is_this_place();
   endGame();
   }
//intdec
int _count;
//itemdec
Item _the_greatest_sword_ever = new Item();
//charadec
Character _xiaowei_the_greatest_man_ever = new Character();
//locdec
Location _where_is_this_place = new Location();
//start funtion
public void _where_is_this_place() {
currentLocation = "_where_is_this_place";
while (!((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") <
(0)))){
{
Map<String,String> keysToActionName0 = new HashMap<String, String>();
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
Map<String, String> actionNameToOutput0 = new HashMap<String, String>();
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
System.out.println("CHOOSE AN ACTION:");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
keysToActionName0.put("a", "_attack");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
actionNameToOutput0.put("_attack", "hia!");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
System.out.println("Type a for hia!");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
keysToActionName0.put("u", "_up");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
actionNameToOutput0.put("_up", "up");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
System.out.println("Type u for up");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
keysToActionName0.put("f", "_fin");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
actionNameToOutput0.put("_fin", "end");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
System.out.println("Type f for end");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
Scanner in0 = new Scanner(System.in);
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
String input0 = in0.nextLine();
```

```
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
while(!keysToActionName0.containsKey(input0)) {
System.out.println("Invalid input, try again");
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
input0 = in0.nextLine();
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
}
System.out.println("You typed " + input0);
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
String action0 = keysToActionName0.get(input0);
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
if(action0.equals("_attack")) {
{
dummy = ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") +
(1)));
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
}
_where_is_this_place();
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
}
if(action0.equals("_up")) {
{
int num = r.nextInt(100);
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
if(num >= 0 && num < 40) {
{
dummy = (_count = (_count - (1)));
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
System.out.println(""+ ("count:"));
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
System.out.println(""+ (_count));
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
}
}
if(num >= 40 && num < 100) {
{
dummy = (_count = (_count + (1)));
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
System.out.println(""+ ("count:"));
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
System.out.println(""+ (_count));
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
}
}
}
_where_is_this_place();
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
}
if(action0.equals("_fin")) {
{
dummy = (entitySetInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life", (-
(1))));
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
endGame();
}
_where_is_this_place();
if ((entityHasInt("_xiaowei_the_greatest_man_ever", Type.CHARACTER, "_life") < (0)))
```

```
endGame();
}
}
}
endGame();
}
}

abstract class Entity {

   Map<String, Integer> intAttrs = new HashMap<String, Integer>();
   Map<String, String> strAttrs = new HashMap<String, String>();

   public void addIntAttr(String name, int value) {
      intAttrs.put(name, value);
   }

   public void addStrAttr(String name, String value) {
      strAttrs.put(name, value);
   }
}

class Location extends Entity {
   Set<String> characters = new HashSet<String>();
   Set<String> items = new HashSet<String>();

   public void addItem(String name, Map<String, Item> itemses) {
      if(itemses.containsKey(name))
                  items.add(name);
      else
            System.out.println("Error: The item you attempted to add no longer exists");
   }

   public void addItem(String name){
      items.add(name);
      }

   public void removeItem(String name) {
      items.remove(name);
   }

   public void showCharacter(String name, Map<String, Character> characterses) {
      if(characterses.containsKey(name))
      characters.add(name);
      else
            System.out.println("Error: The character you attempted to use no longer
exists");
   }
   public void showCharacter(String name){

      characters.add(name);
      }

   public void hideCharacter(String name) {
      characters.remove(name);
   }
}

class Character extends Entity {
   Set<String> items = new HashSet<String>();

   public void addItem(String name, String locationNow, Map<String, Location> locations){

      if(locations.get(locationNow).items.contains(name)){
            locations.get(locationNow).removeItem(name);
            items.add(name);
      }
      else
            System.out.println("Error: The item you attempted to grab is not in this
location");
}
```

```
public void removeItem(String name, String locationNow, Map<String, Location> locations)
{
            if (items.contains(name)){
            items.remove(name);
            locations.get(locationNow).addItem(name);
   }
      else
            System.out.println("Error: The character does not have the item you attempted
to drop");
}

   public void addItem(String name) {
      items.add(name);
   }

   public void removeItem(String name) {
      items.remove(name);
   }
}

class Item extends Entity {
}
```

```
/*RockPaperScissors.next by Ernesto */

int won;
int lost;


location game {(int outcome),(),()}


start game end (game.outcome == 4) {
	output "Games won:";
	output won;
	output "";
	output "Games lost:";
	output lost;
	output "";
	output "rock paper scissors says shoot!";

	[?
		prob 33 game.outcome = 1;
		prob 33 game.outcome = 2;
		prob 34 game.outcome = 3;
	?]



	choose (scissors, "scissors", "s")
		(paper, "paper", "p")
		(rock, "rock", "r")
		(exit, "exit", "e")
	{
		when scissors{
			if(game.outcome == 1) then{
				output "opponent picked scissors!";
				output "You tied";
			}
			else if ( game.outcome == 2) then{
				output "opponent picked paper!";
				output "You won!!";
				won = won + 1;
			}
			else{
				output "opponent picked rock!";
				output"You lost";
				lost = lost + 1;
			}
		}
		next game

		when paper{
			if(game.outcome == 2) then{
				output "opponent picked paper!";
				output "You tied";
			}
			else if ( game.outcome == 3) then{
				output "opponent picked rock!";
				output "You won!!";
				won = won + 1;
			}
			else{
				output "opponent picked scissors!";
				output"You lost";
				lost = lost + 1;
			}
		}
		next game

		when rock{
			if(game.outcome == 3) then{
				output "opponent picked rock!";
```

```
                        output "You tied";
                  }
                  else if ( game.outcome == 1) then{
                        output "opponent picked scissors!";
                        output "You won!!";
                        won = won + 1;
                  }
                  else{
                        output "opponent picked paper!";
                        output"You lost";
                        lost = lost + 1;
                  }
            }
            next game


            when exit
                  game.outcome = 4;
            next game
      }

}
```

```
//Generated Next.java file for RockPaperScissors.Next
import java.util.*;

public class Next {
   enum Type {INT, STRING, CHARACTER, ITEM, LOCATION}

   static Random r = new Random();
   Object dummy;
   String currentLocation;
   Map<String, Location> locations = new HashMap<String, Location>();
   Map<String, Character> characters = new HashMap<String, Character>();
   Map<String, Item> items = new HashMap<String, Item>();
   Map<String, Type> types = new HashMap<String, Type>();

   public static void main(String[] args) {
    (new Next()).play();
   }

   public int boolToInt(boolean value) {
        if(value) {
            return 1;
        }
        else {
            return 0;
        }
   }

   public String entitySetString(String key1, Type type1, String key2, String value) {
    boolean valueSet = false;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  loc.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.ITEM) {
            Item item = items.get(key1);
            if(item != null) {
                  item.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }

      if(!valueSet) {
            throw new RuntimeException();
      }

      return value;
   }

   public int entitySetInt(String key1, Type type1, String key2, int value) {
      boolean foundReturnValue = false;

      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
            loc.intAttrs.put(key2, value);
                  foundReturnValue = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.intAttrs.put(key2, value);
```

```
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.ITEM) {
          Item item = items.get(key1);
          if(item != null) {
                item.intAttrs.put(key2, value);
                foundReturnValue = true;
          }
    }

    if(foundReturnValue == false) {
          throw new RuntimeException();
    }

    return value;
}

    public boolean isTrue(Object object) {
        if(object instanceof String) {
           if(((String)object).isEmpty()) {
               return false;
           }
        }
        else if(object instanceof Integer) {
           if((Integer)object == 0) {
               return false;
           }
        }
        else {
            if(object == null) {
                return false;
            }
        }

        return true;
    }

  public void killFunction(String varName){
    if (characters.containsKey(varName)){
          characters.remove(varName);
          for (String key: locations.keySet()){
                      locations.get(key).hideCharacter(varName);
          }
    }
    else if (items.containsKey(varName)){
          items.remove(varName);
          for (String key:locations.keySet()){
                locations.get(key).removeItem(varName);
                }
          for (String key : characters.keySet()){
                characters.get(key).removeItem(varName);
          }
    }
  }

  public int entityHasInt(String key1, Type type1, String key2) {
    int returnValue = 0;
    boolean foundReturnValue = false;

    if(type1 == Type.LOCATION) {
          Location loc = locations.get(key1);
          if(loc != null) {
                returnValue = loc.intAttrs.get(key2);
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.CHARACTER) {
          Character character = characters.get(key1);
          if(character != null) {
                returnValue = character.intAttrs.get(key2);
                foundReturnValue = true;
```

```
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.intAttrs.get(key2);
			foundReturnValue = true;
		}
	}

	if(foundReturnValue == false) {
		throw new RuntimeException();
	}

	return returnValue;
}

public String entityHasString(String key1, Type type1, String key2) {
	String returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			returnValue = loc.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			returnValue = character.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.strAttrs.get(key2);
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Item entityHasItem(String key1, Type type1, String key2) {
	Item returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			if(loc.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			if(character.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Character entityHasCharacter(String key1, Type type1, String key2) {
```

```
      Character returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            throw new RuntimeException();
      }

      return returnValue;
    }

    public int entityExistsItem(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  if(character.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
    }

    public int entityExistsCharacter(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
    }

   public void endGame() {
      System.exit(0);
   }

   public void play() {
locations.put("_game",_game);
types.put("_game", Type.LOCATION);
_game.addIntAttr("_outcome", 0);
types.put("_outcome", Type.INT);
//Location function call
_game();
```

```
   endGame();
   }
//intdec
int _won;
//intdec
int _lost;
//locdec
Location _game = new Location();
//start funtion
public

void _game() {
currentLocation = "_game";
while (!(entityHasInt("_game", Type.LOCATION, "_outcome") ==

(4))){
{
System.out.println(""+ ("Games won:"));
if (entityHasInt("_game", Type.LOCATION, "_outcome") ==

(4))
endGame();
System.out.println(""+ (_won));
if (entityHasInt("_game", Type.LOCATION, "_outcome") ==

(4))
endGame();
System.out.println(""+ (""));
if (entityHasInt("_game", Type.LOCATION, "_outcome") ==

(4))
endGame();
System.out.println(""+ ("Games lost:"));
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
System.out.println(""+ (_lost));
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
System.out.println(""+ (""));
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
System.out.println(""+ ("rock paper scissors says shoot!"));
if

(entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
int num = r.nextInt(100);
if

(entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
if(num >= 0 && num < 33) {
dummy =

(entitySetInt("_game", Type.LOCATION, "_outcome", (1)));
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
}
if(num >= 33 && num < 66) {
dummy = (entitySetInt("_game", Type.LOCATION,

"_outcome", (2)));
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
```

```
}
if(num >= 66

&& num < 100) {
dummy = (entitySetInt("_game", Type.LOCATION, "_outcome", (3)));
if (entityHasInt

("_game", Type.LOCATION, "_outcome") == (4))
endGame();
}
Map<String,String> keysToActionName17 = new

HashMap<String, String>();
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
Map<String, String> actionNameToOutput17 = new HashMap<String, String>();
if (entityHasInt("_game",

Type.LOCATION, "_outcome") == (4))
endGame();
System.out.println("CHOOSE AN ACTION:");
if (entityHasInt

("_game", Type.LOCATION, "_outcome") == (4))
endGame();
keysToActionName17.put("s", "_scissors");
if

(entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
actionNameToOutput17.put

("_scissors", "scissors");
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
System.out.println("Type s for scissors");
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
keysToActionName17.put("p", "_paper");
if (entityHasInt("_game", Type.LOCATION, "_outcome")

== (4))
endGame();
actionNameToOutput17.put("_paper", "paper");
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
System.out.println("Type p for paper");
if (entityHasInt("_game",

Type.LOCATION, "_outcome") == (4))
endGame();
keysToActionName17.put("r", "_rock");
if (entityHasInt

("_game", Type.LOCATION, "_outcome") == (4))
endGame();
actionNameToOutput17.put("_rock", "rock");
if

(entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
System.out.println("Type r for

rock");
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
keysToActionName17.put

("e", "_exit");
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
```

```
endGame();
actionNameToOutput17.put("_exit", "exit");
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
System.out.println("Type e for exit");
if (entityHasInt("_game", Type.LOCATION, "_outcome")

== (4))
endGame();
Scanner in17 = new Scanner(System.in);
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
String input17 = in17.nextLine();
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
while(!keysToActionName17.containsKey(input17)) {
System.out.println

("Invalid input, try again");
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
input17 = in17.nextLine();
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
}
System.out.println("You typed " + input17);
if (entityHasInt("_game", Type.LOCATION, "_outcome") ==

(4))
endGame();
String action17 = keysToActionName17.get(input17);
if (entityHasInt("_game",

Type.LOCATION, "_outcome") == (4))
endGame();
if(action17.equals("_scissors")) {
{
if(entityHasInt

("_game", Type.LOCATION, "_outcome") == (1)) {
{
System.out.println(""+ ("opponent picked scissors!"));
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
System.out.println(""+ ("You

tied"));
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
}
}
else {
if(entityHasInt

("_game", Type.LOCATION, "_outcome") == (2)) {
{
System.out.println(""+ ("opponent picked paper!"));
if

(entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
System.out.println(""+ ("You

won!!"));
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
dummy = (_won = (_won +
```

```
(1)));
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
}
}
else {
{
System.out.println(""+ ("opponent picked rock!"));
if (entityHasInt("_game", Type.LOCATION, "_outcome")

== (4))
endGame();
System.out.println(""+ ("You lost"));
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
dummy = (_lost = (_lost + (1)));
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
}
}
}
}
_game();
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
}
if(action17.equals("_paper")) {
{
if(entityHasInt("_game", Type.LOCATION, "_outcome") == (2))

{
{
System.out.println(""+ ("opponent picked paper!"));
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
System.out.println(""+ ("You tied"));
if (entityHasInt("_game",

Type.LOCATION, "_outcome") == (4))
endGame();
}
}
else {
if(entityHasInt("_game", Type.LOCATION,

"_outcome") == (3)) {
{
System.out.println(""+ ("opponent picked rock!"));
if (entityHasInt("_game",

Type.LOCATION, "_outcome") == (4))
endGame();
System.out.println(""+ ("You won!!"));
if (entityHasInt

("_game", Type.LOCATION, "_outcome") == (4))
endGame();
dummy = (_won = (_won + (1)));
if (entityHasInt

("_game", Type.LOCATION, "_outcome") == (4))
endGame();
}
}
else {
{
```

```
System.out.println(""+ ("opponent

picked scissors!"));
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
System.out.println(""+ ("You lost"));
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
dummy = (_lost = (_lost + (1)));
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
}
}
}
}
_game();
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
}
if

(action17.equals("_rock")) {
{
if(entityHasInt("_game", Type.LOCATION, "_outcome") == (3)) {
{
System.out.println(""+ ("opponent picked rock!"));
if (entityHasInt("_game", Type.LOCATION, "_outcome")

== (4))
endGame();
System.out.println(""+ ("You tied"));
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
}
}
else {
if(entityHasInt("_game", Type.LOCATION, "_outcome") == (1)) {
{
System.out.println(""+ ("opponent picked scissors!"));
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
System.out.println(""+ ("You won!!"));
if (entityHasInt("_game",

Type.LOCATION, "_outcome") == (4))
endGame();
dummy = (_won = (_won + (1)));
if (entityHasInt("_game",

Type.LOCATION, "_outcome") == (4))
endGame();
}
}
else {
{
System.out.println(""+ ("opponent picked

paper!"));
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
System.out.println(""+

("You lost"));
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
dummy = (_lost =

(_lost + (1)));
```

```
if (entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
}
}
}
}
_game();
if

(entityHasInt("_game", Type.LOCATION, "_outcome") == (4))
endGame();
}
if(action17.equals("_exit")) {
dummy = (entitySetInt("_game", Type.LOCATION, "_outcome", (4)));
if (entityHasInt("_game",

Type.LOCATION, "_outcome") == (4))
endGame();
_game();
if (entityHasInt("_game", Type.LOCATION,

"_outcome") == (4))
endGame();
}
}
}
endGame();
}
}

abstract class Entity {

   Map<String, Integer> intAttrs = new HashMap<String, Integer>();
   Map<String, String> strAttrs = new HashMap<String, String>();

   public void addIntAttr(String name, int value) {
      intAttrs.put(name, value);
   }

   public void addStrAttr(String name, String value) {
      strAttrs.put(name, value);
   }
}

class Location extends Entity {
   Set<String> characters = new HashSet<String>();
   Set<String> items = new HashSet<String>();

   public void addItem(String name, Map<String, Item> itemses) {
      if(itemses.containsKey(name))
                  items.add(name);
      else
            System.out.println("Error: The item you attempted to add no longer exists");
   }

   public void addItem(String name){
      items.add(name);
      }

   public void removeItem(String name) {
      items.remove(name);
   }

   public void showCharacter(String name, Map<String, Character> characterses) {
      if(characterses.containsKey(name))
      characters.add(name);
      else
            System.out.println("Error: The character you attempted to use no longer
exists");
   }
   public void showCharacter(String name){
```

```
      characters.add(name);
      }

   public void hideCharacter(String name) {
      characters.remove(name);
   }
}

class Character extends Entity {
   Set<String> items = new HashSet<String>();

   public void addItem(String name, String locationNow, Map<String, Location> locations){

      if(locations.get(locationNow).items.contains(name)){
            locations.get(locationNow).removeItem(name);
            items.add(name);
      }
      else
            System.out.println("Error: The item you attempted to grab is not in this
location");
}

public void removeItem(String name, String locationNow, Map<String, Location> locations)
{
            if (items.contains(name)){
            items.remove(name);
            locations.get(locationNow).addItem(name);
   }
      else
            System.out.println("Error: The character does not have the item you attempted
to

drop");
}

   public void addItem(String name) {
      items.add(name);
   }

   public void removeItem(String name) {
      items.remove(name);
   }
}

class Item extends Entity {
}
```

```
/*Tutorial1.next by Ernesto */
int num = 0;
string announce = "not in the world:";
character you {(string slogan = "Hello world!"),()}
location here {(),(),(you)}

start here end (num == 1) {
      if (exists here.you) then
            output you.slogan;
      else
            output announce;
      num = 1;

}
```

```
//Next.java file generated for Tutorial1.Next
import java.util.*;

public class Next {
   enum Type {INT, STRING, CHARACTER, ITEM, LOCATION}

   static Random r = new Random();
   Object dummy;
   String currentLocation;
   Map<String, Location> locations = new HashMap<String, Location>();
   Map<String, Character> characters = new HashMap<String, Character>();
   Map<String, Item> items = new HashMap<String, Item>();
   Map<String, Type> types = new HashMap<String, Type>();

   public static void main(String[] args) {
    (new Next()).play();
   }

   public int boolToInt(boolean value) {
        if(value) {
            return 1;
        }
        else {
            return 0;
        }
   }

   public String entitySetString(String key1, Type type1, String key2, String value) {
    boolean valueSet = false;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  loc.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.ITEM) {
            Item item = items.get(key1);
            if(item != null) {
                  item.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }

      if(!valueSet) {
            throw new RuntimeException();
      }

      return value;
   }

   public int entitySetInt(String key1, Type type1, String key2, int value) {
      boolean foundReturnValue = false;

      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
            loc.intAttrs.put(key2, value);
                  foundReturnValue = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.intAttrs.put(key2, value);
```

```
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.ITEM) {
          Item item = items.get(key1);
          if(item != null) {
                item.intAttrs.put(key2, value);
                foundReturnValue = true;
          }
    }

    if(foundReturnValue == false) {
          throw new RuntimeException();
    }

    return value;
}

    public boolean isTrue(Object object) {
        if(object instanceof String) {
           if(((String)object).isEmpty()) {
               return false;
           }
        }
        else if(object instanceof Integer) {
           if((Integer)object == 0) {
               return false;
           }
        }
        else {
            if(object == null) {
                return false;
            }
        }

        return true;
    }

  public void killFunction(String varName){
    if (characters.containsKey(varName)){
          characters.remove(varName);
          for (String key: locations.keySet()){
                      locations.get(key).hideCharacter(varName);
          }
    }
    else if (items.containsKey(varName)){
          items.remove(varName);
          for (String key:locations.keySet()){
                locations.get(key).removeItem(varName);
                }
          for (String key : characters.keySet()){
                characters.get(key).removeItem(varName);
          }
    }
  }

  public int entityHasInt(String key1, Type type1, String key2) {
    int returnValue = 0;
    boolean foundReturnValue = false;

    if(type1 == Type.LOCATION) {
          Location loc = locations.get(key1);
          if(loc != null) {
                returnValue = loc.intAttrs.get(key2);
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.CHARACTER) {
          Character character = characters.get(key1);
          if(character != null) {
                returnValue = character.intAttrs.get(key2);
                foundReturnValue = true;
```

```
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.intAttrs.get(key2);
			foundReturnValue = true;
		}
	}

	if(foundReturnValue == false) {
		throw new RuntimeException();
	}

	return returnValue;
}

public String entityHasString(String key1, Type type1, String key2) {
	String returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			returnValue = loc.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			returnValue = character.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.strAttrs.get(key2);
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Item entityHasItem(String key1, Type type1, String key2) {
	Item returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			if(loc.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			if(character.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Character entityHasCharacter(String key1, Type type1, String key2) {
```

```
      Character returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            throw new RuntimeException();
      }

      return returnValue;
    }

    public int entityExistsItem(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  if(character.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
    }

    public int entityExistsCharacter(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
    }

   public void endGame() {
      System.exit(0);
   }

   public void play() {
_num = (0);
_announce = "not in the world:";
characters.put("_you",_you);
types.put("_you",

Type.CHARACTER);
```

```
_you.addStrAttr("_slogan","Hello world!");
types.put("_slogan", Type.STRING);
locations.put("_here",_here);
types.put("_here", Type.LOCATION);
_here.showCharacter("_you");
//Location

function call
_here();
   endGame();
   }
//intdecinit
int _num;
//strdecinit
String _announce;
//charadec
Character _you = new Character();
//locdec
Location _here = new Location();
//start funtion
public void _here() {
currentLocation = "_here";
while (!(_num == (1))){
{
if(isTrue(entityExistsCharacter("_here", Type.LOCATION, "_you"))) {
System.out.println(""+ (entityHasString("_you", Type.CHARACTER, "_slogan")));
if (_num == (1))
endGame

();
}
else {
System.out.println(""+ (_announce));
if (_num == (1))
endGame();
}
dummy = (_num = (1));
if (_num

== (1))
endGame();
}
}
endGame();
}
}

abstract class Entity {

   Map<String, Integer> intAttrs = new HashMap<String, Integer>();
   Map<String, String> strAttrs = new HashMap<String, String>();

   public void addIntAttr(String name, int value) {
      intAttrs.put(name, value);
   }

   public void addStrAttr(String name, String value) {
      strAttrs.put(name, value);
   }
}

class Location extends Entity {
   Set<String> characters = new HashSet<String>();
   Set<String> items = new HashSet<String>();

   public void addItem(String name, Map<String, Item> itemses) {
      if(itemses.containsKey(name))
                  items.add(name);
      else
            System.out.println("Error: The item you attempted to add no longer exists");
   }
```

```
   public void addItem(String name){
      items.add(name);
      }

   public void removeItem(String name) {
      items.remove(name);
   }

   public void showCharacter(String name, Map<String, Character> characterses) {
      if(characterses.containsKey(name))
      characters.add(name);
      else
            System.out.println("Error: The character you attempted to use no longer
exists");
   }
   public void showCharacter(String name){

      characters.add(name);
      }

   public void hideCharacter(String name) {
      characters.remove(name);
   }
}

class Character extends Entity {
   Set<String> items = new HashSet<String>();

   public void addItem(String name, String locationNow, Map<String, Location> locations){

      if(locations.get(locationNow).items.contains(name)){
            locations.get(locationNow).removeItem(name);
            items.add(name);
      }
      else
            System.out.println("Error: The item you attempted to grab is not in this
location");
}

public void removeItem(String name, String locationNow, Map<String, Location> locations)
{
            if (items.contains(name)){
            items.remove(name);
            locations.get(locationNow).addItem(name);
   }
      else
            System.out.println("Error: The character does not have the item you attempted
to

drop");
}

   public void addItem(String name) {
      items.add(name);
   }

   public void removeItem(String name) {
      items.remove(name);
   }
}

class Item extends Entity {
}
```

```
/*Tutorial2.next by Ernesto */
int num;

character person {(),()}
location here {(),(),()}
item object {()}

start here end (num == 1) {

	[?
		prob 50 output "This is line 1 of a possible 2";
		prob 50 output "This is line 2 of a possible 2";
	?]

	if (exists person.object) then
		output "The person has the object";
	else
		output "The person does not have the object";

	if(exists here.object) then
		output "The object is in the location";
	else
		output "The object is not in the location";


	choose (grabItem, "character grab item", "g")
		(dropItem, "character drop item", "d")
		(showItem, "show item in location", "s")
		(hideItem, "hide item from location", "h")
		(exit, "exit", "e")
	{
		when grabItem
			grab person.object;
		next here

		when dropItem
			drop person.object;
		next here

		when showItem
			show here.object;
		next here

		when hideItem
			hide here.object;
		next here

		when exit
			num = 1;
		next here
	}

}
```

```
//Next.java file generated for Tutorial2.next
import java.util.*;

public class Next {
   enum Type {INT, STRING, CHARACTER, ITEM, LOCATION}

   static Random r = new Random();
   Object dummy;
   String currentLocation;
   Map<String, Location> locations = new HashMap<String, Location>();
   Map<String, Character> characters = new HashMap<String, Character>();
   Map<String, Item> items = new HashMap<String, Item>();
   Map<String, Type> types = new HashMap<String, Type>();

   public static void main(String[] args) {
    (new Next()).play();
   }

   public int boolToInt(boolean value) {
        if(value) {
            return 1;
        }
        else {
            return 0;
        }
   }

   public String entitySetString(String key1, Type type1, String key2, String value) {
    boolean valueSet = false;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  loc.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.ITEM) {
            Item item = items.get(key1);
            if(item != null) {
                  item.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }

      if(!valueSet) {
            throw new RuntimeException();
      }

      return value;
   }

   public int entitySetInt(String key1, Type type1, String key2, int value) {
      boolean foundReturnValue = false;

      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
            loc.intAttrs.put(key2, value);
                  foundReturnValue = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.intAttrs.put(key2, value);
```

```
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.ITEM) {
          Item item = items.get(key1);
          if(item != null) {
                item.intAttrs.put(key2, value);
                foundReturnValue = true;
          }
    }

    if(foundReturnValue == false) {
          throw new RuntimeException();
    }

    return value;
}

    public boolean isTrue(Object object) {
        if(object instanceof String) {
           if(((String)object).isEmpty()) {
               return false;
           }
        }
        else if(object instanceof Integer) {
           if((Integer)object == 0) {
               return false;
           }
        }
        else {
            if(object == null) {
                return false;
            }
        }

        return true;
    }

  public void killFunction(String varName){
    if (characters.containsKey(varName)){
          characters.remove(varName);
          for (String key: locations.keySet()){
                      locations.get(key).hideCharacter(varName);
          }
    }
    else if (items.containsKey(varName)){
          items.remove(varName);
          for (String key:locations.keySet()){
                locations.get(key).removeItem(varName);
                }
          for (String key : characters.keySet()){
                characters.get(key).removeItem(varName);
          }
    }
  }

  public int entityHasInt(String key1, Type type1, String key2) {
    int returnValue = 0;
    boolean foundReturnValue = false;

    if(type1 == Type.LOCATION) {
          Location loc = locations.get(key1);
          if(loc != null) {
                returnValue = loc.intAttrs.get(key2);
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.CHARACTER) {
          Character character = characters.get(key1);
          if(character != null) {
                returnValue = character.intAttrs.get(key2);
                foundReturnValue = true;
```

```
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.intAttrs.get(key2);
			foundReturnValue = true;
		}
	}

	if(foundReturnValue == false) {
		throw new RuntimeException();
	}

	return returnValue;
}

public String entityHasString(String key1, Type type1, String key2) {
	String returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			returnValue = loc.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			returnValue = character.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.strAttrs.get(key2);
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Item entityHasItem(String key1, Type type1, String key2) {
	Item returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			if(loc.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			if(character.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Character entityHasCharacter(String key1, Type type1, String key2) {
```

```
      Character returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            throw new RuntimeException();
      }

      return returnValue;
    }

    public int entityExistsItem(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  if(character.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
    }

    public int entityExistsCharacter(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
    }

   public void endGame() {
      System.exit(0);
   }

   public void play() {
characters.put("_person",_person);
types.put("_person", Type.CHARACTER);
locations.put("_here",_here);
types.put("_here", Type.LOCATION);
items.put("_object",_object);
types.put("_object", Type.ITEM);
```

```
//Location function call
_here();
   endGame();
   }
//intdec
int _num;
//charadec
Character _person = new Character();
//locdec
Location _here = new

Location();
//itemdec
Item _object = new Item();
//start funtion
public void _here() {
currentLocation =

"_here";
while (!(_num == (1))){
{
int num = r.nextInt(100);
if (_num == (1))
endGame();
if(num >= 0 && num

< 50) {
System.out.println(""+ ("This is line 1 of a possible 2"));
if (_num == (1))
endGame();
}
if(num >=

50 && num < 100) {
System.out.println(""+ ("This is line 2 of a possible 2"));
if (_num == (1))
endGame

();
}
if(isTrue(entityExistsItem("_person", Type.CHARACTER, "_object"))) {
System.out.println(""+ ("The

person has the object"));
if (_num == (1))
endGame();
}
else {
System.out.println(""+ ("The person does not

have the object"));
if (_num == (1))
endGame();
}
if(isTrue(entityExistsItem("_here", Type.LOCATION,

"_object"))) {
System.out.println(""+ ("The object is in the location"));
if (_num == (1))
endGame();
}
else {
System.out.println(""+ ("The object is not in the location"));
if (_num == (1))
endGame();
}
Map<String,String> keysToActionName19 = new HashMap<String, String>();
if (_num == (1))
endGame();
Map<String, String> actionNameToOutput19 = new HashMap<String, String>();
if (_num == (1))
endGame();
```

```
System.out.println("CHOOSE AN ACTION:");
if (_num == (1))
endGame();
keysToActionName19.put("g",

"_grabItem");
if (_num == (1))
endGame();
actionNameToOutput19.put("_grabItem", "character grab item");
if

(_num == (1))
endGame();
System.out.println("Type g for character grab item");
if (_num == (1))
endGame();
keysToActionName19.put("d", "_dropItem");
if (_num == (1))
endGame();
actionNameToOutput19.put

("_dropItem", "character drop item");
if (_num == (1))
endGame();
System.out.println("Type d for

character drop item");
if (_num == (1))
endGame();
keysToActionName19.put("s", "_showItem");
if (_num ==

(1))
endGame();
actionNameToOutput19.put("_showItem", "show item in location");
if (_num == (1))
endGame

();
System.out.println("Type s for show item in location");
if (_num == (1))
endGame();
keysToActionName19.put("h", "_hideItem");
if (_num == (1))
endGame();
actionNameToOutput19.put

("_hideItem", "hide item from location");
if (_num == (1))
endGame();
System.out.println("Type h for hide

item from location");
if (_num == (1))
endGame();
keysToActionName19.put("e", "_exit");
if (_num == (1))
endGame();
actionNameToOutput19.put("_exit", "exit");
if (_num == (1))
endGame();
System.out.println("Type

e for exit");
if (_num == (1))
endGame();
Scanner in19 = new Scanner(System.in);
if (_num == (1))
endGame

();
String input19 = in19.nextLine();
```

```
if (_num == (1))
endGame();
while(!keysToActionName19.containsKey

(input19)) {
System.out.println("Invalid input, try again");
if (_num == (1))
endGame();
input19 =

in19.nextLine();
if (_num == (1))
endGame();
}
System.out.println("You typed " + input19);
if (_num == (1))
endGame();
String action19 = keysToActionName19.get(input19);
if (_num == (1))
endGame();
if

(action19.equals("_grabItem")) {
_person.addItem ("_object", currentLocation, locations);
if (_num ==

(1))
endGame();
_here();
if (_num == (1))
endGame();
}
if(action19.equals("_dropItem")) {
_person.removeItem

("_object", currentLocation, locations);
if (_num == (1))
endGame();
_here();
if (_num == (1))
endGame();
}
if(action19.equals("_showItem")) {
_here.addItem ("_object",items);
if (_num == (1))
endGame();
_here();
if

(_num == (1))
endGame();
}
if(action19.equals("_hideItem")) {
_here.removeItem ("_object");
if (_num ==

(1))
endGame();
_here();
if (_num == (1))
endGame();
}
if(action19.equals("_exit")) {
dummy = (_num = (1));
if

(_num == (1))
endGame();
_here();
if (_num == (1))
endGame();
}
```

```
}
}
endGame();
}
}

abstract class Entity {

   Map<String, Integer> intAttrs = new HashMap<String, Integer>();
   Map<String, String> strAttrs = new HashMap<String, String>();

   public void addIntAttr(String name, int value) {
      intAttrs.put(name, value);
   }

   public void addStrAttr(String name, String value) {
      strAttrs.put(name, value);
   }
}

class Location extends Entity {
   Set<String> characters = new HashSet<String>();
   Set<String> items = new HashSet<String>();

   public void addItem(String name, Map<String, Item> itemses) {
      if(itemses.containsKey(name))
                  items.add(name);
      else
            System.out.println("Error: The item you attempted to add no longer exists");
   }

   public void addItem(String name){
      items.add(name);
      }

   public void removeItem(String name) {
      items.remove(name);
   }

   public void showCharacter(String name, Map<String, Character> characterses) {
      if(characterses.containsKey(name))
      characters.add(name);
      else
            System.out.println("Error: The character you attempted to use no longer
exists");
   }
   public void showCharacter(String name){

      characters.add(name);
      }

   public void hideCharacter(String name) {
      characters.remove(name);
   }
}

class Character extends Entity {
   Set<String> items = new HashSet<String>();

   public void addItem(String name, String locationNow, Map<String, Location> locations){

      if(locations.get(locationNow).items.contains(name)){
            locations.get(locationNow).removeItem(name);
            items.add(name);
      }
      else
            System.out.println("Error: The item you attempted to grab is not in this
location");
}

public void removeItem(String name, String locationNow, Map<String, Location> locations)
{
```

```
            if (items.contains(name)){
            items.remove(name);
            locations.get(locationNow).addItem(name);
   }
      else
            System.out.println("Error: The character does not have the item you attempted
to

drop");
}

   public void addItem(String name) {
      items.add(name);
   }

   public void removeItem(String name) {
      items.remove(name);
   }
}

class Item extends Entity {
}
```

```
/*ProfEdwardsAdventure.next by Ernesto */

item laser{(string color = "Green", int damage = 10)}
item PDP_8{(int memory = 10)}
item not_windows_computer {(string OS = "Not-Windows")}
item fish_sandwich {(int portion_left = 100, string taste = "yummy")}

character prof_edwards {(int life = 10000, int awesomeness = 300, string slogan = "Add
another layer

of indirection!"), ()}

item dragon_book {(int damage = 5)}
character prof_aho{(int life = 20, int awesomeness = 20, string slogan = "I believe thats
on page 42

of the Dragon Book!"), (dragon_book)}


character godzilla{(int life = 100), ()}
character ronald{(),()}

character plt_mob {(int life = 3, int damage = 6), ()}


character bill_gates {(int life = 2, string money = "$$$54 billion$$$", string slogan =
"It just

works"), ()}


location dragon_cave {(int depth = 500), (), (prof_edwards, prof_aho)}
location ucb {(), (not_windows_computer), (bill_gates, plt_mob)}
location mcdonalds {(), (fish_sandwich), (ronald)}
location taiwan {(), (PDP_8), ()}

int itemNum = 0;
int enemy;
int intro = 0;

location columbia{(), (), (prof_edwards)}

start columbia end(prof_edwards.life < 0 or itemNum == 4) {
      if intro == 0 then{
            output "Professor Edwards has woken up this morning from a dream!";
            output "In the dream a GCD function tells him to go forth and create the
greatest

compiler ever created!";
            output "It is a sign and he knows he must do it. He will go forth and create
the most

premiums compiler eeeeeeever";
            output "There is a problem, however...";
            output "His items of compiler power have been stolen. He must go forth and
find them";
            intro = 1;
      }
      else ;

      enemy = 0;
      output "You are in columbia";
      output "Where to next?";

      choose (dcave, "The dragon cave", "d")
               (tai, "Taiwan", "t")
               (u, " UC Berkeley", "u")
               (mcd, " McDonalds", "m")
      {

            when dcave {
                        output "Off to the Dragon Cave to battle Professor Aho";
```

```
            } next dragon_cave

            when tai {
                        output "Off to Taiwan in search of the powerful PDP-8";
            } next taiwan

            when u {
                        output "Off to UC Berkeley! ... for some reason.";
            } next ucb

            when mcd {
                        output "Off to McDonalds for a fish burger";
            } next mcdonalds
      }
}



int moment = 0;

start dragon_cave end (prof_edwards.life <= 0  or itemNum == 4) {
      if(moment == 0) then {
            output "You are in Prof. Aho's dragon cave!!";
            output "Aho is ready to fight you.. You must fight him to get his dragon
book!!";
            output "... or maybe not?...";
            moment = 1;
      }
      else ;
      choose (fight, "fight", "f")
            (runaway, "run away", "r")
            (ask, " ask nicely", "a") {

            when fight{
                  [? prob 70 {
                        output "You say: ";
                        output "behold Aho, my shiny laser pointer";
                        prof_aho.life = prof_aho.life - laser.damage;
                        output "You caused Prof. Aho a damage of:";
                        output laser.damage;

                        if (prof_aho.life <= 1) then{
                              output "You have defeated Aho... well done!!";
                              drop prof_aho.dragon_book;
                        }
                        else ;
                  }
                  prob 20 {
                        output "As he hits you Prof. Aho lets you know:";
                        output prof_aho.slogan;
                        prof_edwards.life = prof_edwards.life - dragon_book.damage;
                        output "Prof Aho hurt you this much:";
                        output dragon_book.damage;
                  }
                  prob 10{
                        output "wuah wuah wuah";
                        output "Aho surrenders.";
                        output "YOu have won the dragon book";
                        drop prof_aho.dragon_book;

                  }?]

                  if(exists dragon_cave.dragon_book) then{
                        output "In victory you say your famous slogan:";
                        output prof_edwards.slogan;
                        output "Would you like to pick up the dragon book?";
                        choose(grabbook, "grab", "g")
                              (notgrab, "not grab", "n"){

                              when grabbook { output "You have the dragon book!!";
                                    grab prof_edwards.dragon_book;
                                    if (itemNum == 3) then{
```

```
                                          output "You did it!!";
                                          output "You have collected al four items";
                                          output "With them you build the most premiums

compiler eveeeeer!!";
                                    }
                                    else ;

                                    itemNum = itemNum + 1;
                                    output "Lets go back to columbia";
                              } next columbia

                              when notgrab {
                                    output "You dont have the dragon book!!!";
                                    grab prof_aho.dragon_book;
                              }
                              next dragon_cave
                        }
                  }
                  else ;
            } next dragon_cave

            when runaway {
                  output "You run away to columbia.. aaaawww";
                  moment = 0;

            }next columbia


            when ask {
                  if(exists prof_aho.dragon_book) then{
                        drop prof_aho.dragon_book;
                        grab prof_edwards.dragon_book;
                        output "He actually just gave you the dragon book!";
                        output "... after 3000 hours of telling you about Awk...";

                        if (itemNum == 3) then{
                              output "You did it!!";
                              output "You have collected all four items";
                              output "With them you build the most premiums compiler

eveeeeer!!";
                        }
                        else ;

                        itemNum = itemNum + 1;
                        output "Anyway, time to go back to columbia and choose the next

location";
                  }
                  else output "The professor does not have the dragon book!";

            }next columbia
      }



}

int attack;

start ucb end (prof_edwards.life <= 0 or itemNum == 4) {
      if (bill_gates.life <= 0 or plt_mob.life <= 0) then {
            if (exists ucb.not_windows_computer) then {
                  output "You have won!";
                  output "You have received the not-windows computer!";
                  grab prof_edwards.not_windows_computer;
                  if (itemNum == 3) then{
                        output "You did it!!";
                        output "You have collected all four items";
                        output "With them you build the most premiums compiler
eveeeeer!!";
```

```
                }
                else ;
                itemNum = itemNum + 1;
                output "Time to return to Columbia";
                      choose (col, "Columbia", "c"){
                            when col ;
                            next columbia
                      }
          }
          else{
                output "You have what you needed from here";
                output "Time to return to Columbia";
                choose (col, "Columbia", "c"){
                      when col{

                      } next columbia
                }
          }
    }
    else ;


    if(enemy == 0) then{
          [? prob 50 enemy = 1;
          prob 50 enemy = 2;
          ?]
    }
    else if (enemy == 1) then{

          output "Get ready to fight Bill Gates!!";
          [? prob 40 {
                attack = 1;
                output "Bill throws Microsoft Office at your head!";
          }
          prob 30 {
                 attack = 2;
                output "Bill throws Windows 7 at your feet!";
          }
          prob 30 {
                attack = 3;
                output "Bill is trying to tell you the new features of c#!";
          }

          ?]

          choose(duck, "duck", "d")
                (jump, "jump", "j")
                (cover, "cover your ears", "c")
                {
                when duck {
                      if (attack == 1) then {
                            output "You escaped the horror!";
                            output "Gates is hyperventilating!";
                            bill_gates.life = bill_gates.life - 1;
                      }
                      else if (attack == 2) then {
                            output "You got hit by windows.. you will never function

correctly again!";
                            output "You loose 5 life points!!";
                            prof_edwards.life = prof_edwards.life - 5;
                      }
                      else {
                            output "You heard it all... You are utterly confused!";
                            output "You loose 5 life points!!";
                            prof_edwards.life = prof_edwards.life - 5;
                      }
                } next  ucb
                when jump {
                      if (attack == 2) then {
                            output "You escaped that which makes many stumble and crack
```

```
their heards!";
                              output "Gates is sweating so much!";
                              bill_gates.life = bill_gates.life - 1;
                        }
                        else if (attack == 1) then {
                              output "You got hit by office square in the face... just
like

every user!!";
                              output "You loose 5 life points!!";
                              prof_edwards.life = prof_edwards.life - 5;

                        }
                        else {
                              output "You heard it all... You are utterly confused!";
                              output "You loose 5 life points!!";
                              prof_edwards.life = prof_edwards.life - 5;
                        }
                  } next ucb
                  when cover {
                        if (attack == 3) then {
                              output "Good! You would have never been able to make a good

compiler again!";
                              output "Gates looks dizzy!";
                              bill_gates.life = bill_gates.life - 1;
                        }
                        else if (attack == 1) then {
                              output "You got hit by office square in the face... just
like

every user!!";
                              output "You loose 5 life points!!";
                              prof_edwards.life = prof_edwards.life - 5;

                        }
                        else{
                              output "You got hit by windows.. you will never function

correctly again!";
                              output "You loose 5 life points!!";
                              prof_edwards.life = prof_edwards.life - 5;
                        }

                  } next ucb
            }


      }
      else{

            output "Get ready to fight a huge PLT mob!!";

            [? prob 40 {
                  attack = 1;
                  output "Someone throws a C Reference Manual at you!";
            }
            prob 30 {
                   attack = 2;
                  output "someone is trying to throw the Ocaml camel at your feet!!";
            }
            prob 30 {
                  attack = 3;
                  output "They are all screaming about a new language that is scanned

backwards!";
            }

            ?]
            choose(duck, "duck", "d")
                  (jump, "jump", "j")
                  (cover, "cover your ears", "c")
```

```
                  {
                  when duck {
                        if (attack == 1) then {
                              output "You escaped the horror!";
                              output "They are getting tired!";
                              plt_mob.life = plt_mob.life - 1;
                        }
                        else if (attack == 2) then {
                              output "You got hit by the camel.. you appreciate O'Caml a
little less!";
                              output "You loose 5 life points!!";
                              prof_edwards.life = prof_edwards.life - plt_mob.damage;
                        }
                        else {
                              output "You heard it all... You are utterly confused!";
                              output "You loose 5 life points!!";
                              prof_edwards.life = prof_edwards.life - plt_mob.damage;
                        }
                  } next  ucb
                  when jump {
                        if (attack == 2) then {
                              output "You escaped that which makes many stumble and crack
their heards!";
                              output "The mob is getting bored!";
                              plt_mob.life = plt_mob.life - 1;
                        }
                        else if (attack == 1) then {
                              output "You got hit by the manual square in the face...
your
brain is now a big null pointer!!";
                              output "You loose 5 life points!!";
                              prof_edwards.life = prof_edwards.life - plt_mob.damage;

                        }
                        else {
                              output "You heard it all... You are utterly confused!";
                              output "You loose 5 life points!!";
                              prof_edwards.life = prof_edwards.life - plt_mob.damage;
                        }
                  } next ucb
                  when cover {
                        if (attack == 3) then {
                              output "Good! You would have never been able to make a good
compiler again!";
                              output "Every head in that mob hurts!";
                              bill_gates.life = bill_gates.life - 1;
                        }
                        else if (attack == 1) then {
                              output "You got hit by the manual square in the face...
your
brain is now a big null pointer!!";
                              output "You loose 5 life points!!";
                              prof_edwards.life = prof_edwards.life - plt_mob.damage;

                        }
                        else{
                              output "You got hit by the camel.. you appreciate O'Caml a
little less!";
                              output "You loose 5 life points!!";
                              prof_edwards.life = prof_edwards.life - plt_mob.damage;
                        }

                  } next ucb
            }
      }
```

```
}

start mcdonalds end  (prof_edwards.life <= 0 or itemNum == 4) {
      if (exists mcdonalds.fish_sandwich) then{

            output "Ronald gives you a riddle!";
            output  "What Burger was featured in the greatest advertising campaign
ever?";
            choose (fish, " fish sandwitch", "f")
                  (burger, " Big Mac", "b")
                  (caesar, " Caesar Salad", "c"){

                  when fish {
                        output "Yes, you do believe in magic!";
                        output "YOu have earned a  fish Burger";
                        output "Lets go back to Columbia to keep on with the
adventure!!";
                        grab prof_edwards.fish_sandwich;
                        if (itemNum == 3) then{
                              output "You did it!!";
                              output "You have collected all four items";
                              output "With them you build the most premiums compiler

eveeeeer!!";
                        }
                        else ;
                        itemNum = itemNum + 1;
                  } next columbia

                  when burger
                        output "no... come on... you were there!!";
                  next mcdonalds

                  when caesar
                        output "Thats not even a burger!!";
                  next mcdonalds
            }
      }

      else{
            output "you have collected what you needed here";
            choose(col, "Columbia", "c"){
                  when col{} next columbia
            }
      }

}


int narrated = 0;

start taiwan end (prof_edwards.life <= 0 or itemNum == 4) {
      if (godzilla.life <= 0) then{
            output "...oh no wait, you beat Godzilla!! holy shhh... wow!";
            output "You have received the PDP-8!!";
            if( not exists prof_edwards.PDP_8) then{
                  grab prof_edwards.PDP_8;
                  if (itemNum == 3) then{
                        output "You did it!!";
                        output "You have collected all four items";
                        output "With them you build the most premiums compiler
eveeeeer!!";
                  }
                  else ;
            itemNum = itemNum + 1;
            }
            else ;
      }
      else ;
```

```
      if (exists taiwan.PDP_8) then{
            if (narrated == 0) then{
                  output "You have traveled to Taiwan to get the biggest PDP-8 in the
world";
                  output "You need it for its humongous stack!!";
                  output "The problem is, it belongs to Godzilla!!";
                  narrated = 1;
            }
            else ;

            output "Godzilla is about to step on you!!";

            choose (shoot,"shooting your laser pointer at his foot", "s")
                  (die, "dying squished between its toes", "d")
                  (tickle, "tickling the monster's foot", "t"){

                  when shoot {
                        output "You hit the monster in the foot";
                        godzilla.life = godzilla.life - laser.damage;
                        output " You did 10 points damage!!";
                        output "Keep fighting!!";

                  }next taiwan

                  when die{
                        output "You have died between its green toes";
                        prof_edwards.life = 0;
                  }next columbia

                  when tickle{
                        output "the monster fell back laughing";
                        output "You have done 50 damage!!";
                        godzilla.life = godzilla.life - 50;
                  }next taiwan
            }
      }
      else{
            output "You have the calculator. Time to return to columbia";
            choose(col, "Columbia", "c"){
                  when col; next columbia
            }
      }



}
```

```
//Next.java file generated for ProfEdwardsAdventure.next
import java.util.*;

public class Next {
   enum Type {INT, STRING, CHARACTER, ITEM, LOCATION}

   static Random r = new Random();
   Object dummy;
   String currentLocation;
   Map<String, Location> locations = new HashMap<String, Location>();
   Map<String, Character> characters = new HashMap<String, Character>();
   Map<String, Item> items = new HashMap<String, Item>();
   Map<String, Type> types = new HashMap<String, Type>();

   public static void main(String[] args) {
    (new Next()).play();
   }

   public int boolToInt(boolean value) {
        if(value) {
            return 1;
        }
        else {
            return 0;
        }
   }

   public String entitySetString(String key1, Type type1, String key2, String value) {
    boolean valueSet = false;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  loc.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }
      else if(type1 == Type.ITEM) {
            Item item = items.get(key1);
            if(item != null) {
                  item.strAttrs.put(key2, value);
                  valueSet = true;
            }
      }

      if(!valueSet) {
            throw new RuntimeException();
      }

      return value;
   }

   public int entitySetInt(String key1, Type type1, String key2, int value) {
      boolean foundReturnValue = false;

      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
            loc.intAttrs.put(key2, value);
                  foundReturnValue = true;
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  character.intAttrs.put(key2, value);
```

```
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.ITEM) {
          Item item = items.get(key1);
          if(item != null) {
                item.intAttrs.put(key2, value);
                foundReturnValue = true;
          }
    }

    if(foundReturnValue == false) {
          throw new RuntimeException();
    }

    return value;
}

    public boolean isTrue(Object object) {
        if(object instanceof String) {
           if(((String)object).isEmpty()) {
               return false;
           }
        }
        else if(object instanceof Integer) {
           if((Integer)object == 0) {
               return false;
           }
        }
        else {
            if(object == null) {
                return false;
            }
        }

        return true;
    }

  public void killFunction(String varName){
    if (characters.containsKey(varName)){
          characters.remove(varName);
          for (String key: locations.keySet()){
                      locations.get(key).hideCharacter(varName);
          }
    }
    else if (items.containsKey(varName)){
          items.remove(varName);
          for (String key:locations.keySet()){
                locations.get(key).removeItem(varName);
                }
          for (String key : characters.keySet()){
                characters.get(key).removeItem(varName);
          }
    }
  }

  public int entityHasInt(String key1, Type type1, String key2) {
    int returnValue = 0;
    boolean foundReturnValue = false;

    if(type1 == Type.LOCATION) {
          Location loc = locations.get(key1);
          if(loc != null) {
                returnValue = loc.intAttrs.get(key2);
                foundReturnValue = true;
          }
    }
    else if(type1 == Type.CHARACTER) {
          Character character = characters.get(key1);
          if(character != null) {
                returnValue = character.intAttrs.get(key2);
                foundReturnValue = true;
```

```
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.intAttrs.get(key2);
			foundReturnValue = true;
		}
	}

	if(foundReturnValue == false) {
		throw new RuntimeException();
	}

	return returnValue;
}

public String entityHasString(String key1, Type type1, String key2) {
	String returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			returnValue = loc.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			returnValue = character.strAttrs.get(key2);
		}
	}
	else if(type1 == Type.ITEM) {
		Item item = items.get(key1);
		if(item != null) {
			returnValue = item.strAttrs.get(key2);
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Item entityHasItem(String key1, Type type1, String key2) {
	Item returnValue = null;
	if(type1 == Type.LOCATION) {
		Location loc = locations.get(key1);
		if(loc != null) {
			if(loc.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}
	else if(type1 == Type.CHARACTER) {
		Character character = characters.get(key1);
		if(character != null) {
			if(character.items.contains(key2)) {
				returnValue = items.get(key2);
			}
		}
	}

	if(returnValue == null) {
		throw new RuntimeException();
	}

	return returnValue;
}

public Character entityHasCharacter(String key1, Type type1, String key2) {
```

```
      Character returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            throw new RuntimeException();
      }

      return returnValue;
    }

    public int entityExistsItem(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }
      else if(type1 == Type.CHARACTER) {
            Character character = characters.get(key1);
            if(character != null) {
                  if(character.items.contains(key2)) {
                        returnValue = items.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
    }

    public int entityExistsCharacter(String key1, Type type1, String key2) {
      Object returnValue = null;
      if(type1 == Type.LOCATION) {
            Location loc = locations.get(key1);
            if(loc != null) {
                  if(loc.characters.contains(key2)) {
                        returnValue = characters.get(key2);
                  }
            }
      }

      if(returnValue == null) {
            return 0;
      }

      return 1;
    }

   public void endGame() {
      System.exit(0);
   }

   public void play() {
items.put("_laser",_laser);
types.put("_laser", Type.ITEM);
_laser.addStrAttr("_color","Green");
types.put("_color", Type.STRING);
_laser.addIntAttr("_damage", (10));
types.put("_damage", Type.INT);
```

```
items.put("_PDP_8",_PDP_8);
types.put("_PDP_8", Type.ITEM);
_PDP_8.addIntAttr("_memory", (10));
types.put("_memory", Type.INT);
items.put("_not_windows_computer",_not_windows_computer);
types.put

("_not_windows_computer", Type.ITEM);
_not_windows_computer.addStrAttr("_OS","Not-Windows");
types.put

("_OS", Type.STRING);
items.put("_fish_sandwich",_fish_sandwich);
types.put("_fish_sandwich",

Type.ITEM);
_fish_sandwich.addIntAttr("_portion_left", (100));
types.put("_portion_left", Type.INT);
_fish_sandwich.addStrAttr("_taste","yummy");
types.put("_taste", Type.STRING);
characters.put

("_prof_edwards",_prof_edwards);
types.put("_prof_edwards", Type.CHARACTER);
_prof_edwards.addIntAttr

("_life", (10000));
types.put("_life", Type.INT);
_prof_edwards.addIntAttr("_awesomeness", (300));
types.put("_awesomeness", Type.INT);
_prof_edwards.addStrAttr("_slogan","Add another layer of

indirection!");
types.put("_slogan", Type.STRING);
items.put("_dragon_book",_dragon_book);
types.put

("_dragon_book", Type.ITEM);
_dragon_book.addIntAttr("_damage", (5));
types.put("_damage", Type.INT);
characters.put("_prof_aho",_prof_aho);
types.put("_prof_aho", Type.CHARACTER);
_prof_aho.addIntAttr

("_life", (20));
types.put("_life", Type.INT);
_prof_aho.addIntAttr("_awesomeness", (20));
types.put

("_awesomeness", Type.INT);
_prof_aho.addStrAttr("_slogan","I believe thats on page 42 of the Dragon

Book!");
types.put("_slogan", Type.STRING);
_prof_aho.addItem("_dragon_book");
characters.put

("_godzilla",_godzilla);
types.put("_godzilla", Type.CHARACTER);
_godzilla.addIntAttr("_life", (100));
types.put("_life", Type.INT);
characters.put("_ronald",_ronald);
types.put("_ronald", Type.CHARACTER);
characters.put("_plt_mob",_plt_mob);
types.put("_plt_mob", Type.CHARACTER);
_plt_mob.addIntAttr("_life",

(3));
types.put("_life", Type.INT);
_plt_mob.addIntAttr("_damage", (6));
types.put("_damage", Type.INT);
characters.put("_bill_gates",_bill_gates);
```

```
types.put("_bill_gates", Type.CHARACTER);
_bill_gates.addIntAttr("_life", (2));
types.put("_life", Type.INT);
_bill_gates.addStrAttr

("_money","$$$54 billion$$$");
types.put("_money", Type.STRING);
_bill_gates.addStrAttr("_slogan","It

just works");
types.put("_slogan", Type.STRING);
locations.put("_dragon_cave",_dragon_cave);
types.put

("_dragon_cave", Type.LOCATION);
_dragon_cave.addIntAttr("_depth", (500));
types.put("_depth",

Type.INT);
_dragon_cave.showCharacter("_prof_edwards");
_dragon_cave.showCharacter("_prof_aho");
locations.put("_ucb",_ucb);
types.put("_ucb", Type.LOCATION);
_ucb.addItem("_not_windows_computer");
_ucb.showCharacter("_bill_gates");
_ucb.showCharacter("_plt_mob");
locations.put

("_mcdonalds",_mcdonalds);
types.put("_mcdonalds", Type.LOCATION);
_mcdonalds.addItem("_fish_sandwich");
_mcdonalds.showCharacter("_ronald");
locations.put("_taiwan",_taiwan);
types.put("_taiwan",

Type.LOCATION);
_taiwan.addItem("_PDP_8");
_itemNum = (0);
_intro = (0);
locations.put

("_columbia",_columbia);
types.put("_columbia", Type.LOCATION);
_columbia.showCharacter

("_prof_edwards");
//Location function call
_columbia();
_moment = (0);
//Location function call
_dragon_cave();
//Location function call
_ucb();
//Location function call
_mcdonalds();
_narrated = (0);
//Location function call
_taiwan();
   endGame();
   }
//itemdec
Item _laser = new Item();
//itemdec
Item _PDP_8 = new Item();
//itemdec
Item

_not_windows_computer = new Item();
//itemdec
Item _fish_sandwich = new Item();
//charadec
```

```
Character

_prof_edwards = new Character();
//itemdec
Item _dragon_book = new Item();
//charadec
Character _prof_aho

= new Character();
//charadec
Character _godzilla = new Character();
//charadec
Character _ronald = new

Character();
//charadec
Character _plt_mob = new Character();
//charadec
Character _bill_gates = new

Character();
//locdec
Location _dragon_cave = new Location();
//locdec
Location _ucb = new Location();
//locdec
Location _mcdonalds = new Location();
//locdec
Location _taiwan = new Location();
//intdecinit
int

_itemNum;
//intdec
int _enemy;
//intdecinit
int _intro;
//locdec
Location _columbia = new Location();
//start

funtion
public void _columbia() {
currentLocation = "_columbia";
while (!((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") < (0)) || _itemNum == (4))){
{
if(_intro == (0)) {
{
System.out.println(""+

("Professor Edwards has woken up this morning from a dream!"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("In the dream a

GCD function tells him to go forth and create the greatest compiler ever created!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("It is a sign and he knows he must do it. He will go forth and
create the most

premiums compiler eeeeeeever"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) ||

_itemNum == (4))
endGame();
```

```
System.out.println(""+ ("There is a problem, however..."));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("His items of compiler power have been stolen. He must go forth
and find

them"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame

();
dummy = (_intro = (1));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) ||

_itemNum == (4))
endGame();
}
}
else {
//Empty stmt
}
dummy = (_enemy = (0));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
System.out.println(""+

("You are in columbia"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) ||

_itemNum == (4))
endGame();
System.out.println(""+ ("Where to next?"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
Map<String,String>

keysToActionName16 = new HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") < (0)) || _itemNum == (4))
endGame();
Map<String, String> actionNameToOutput16 = new

HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) ||

_itemNum == (4))
endGame();
System.out.println("CHOOSE AN ACTION:");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
keysToActionName16.put("d", "_dcave");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
actionNameToOutput16.put("_dcave", "The dragon cave");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
System.out.println("Type d for The dragon

cave");
```

```
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame

();
keysToActionName16.put("t", "_tai");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <

(0)) || _itemNum == (4))
endGame();
actionNameToOutput16.put("_tai", "Taiwan");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
System.out.println

("Type t for Taiwan");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum

== (4))
endGame();
keysToActionName16.put("u", "_u");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") < (0)) || _itemNum == (4))
endGame();
actionNameToOutput16.put("_u", " UC Berkeley");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
System.out.println("Type u for  UC Berkeley");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") < (0)) || _itemNum == (4))
endGame();
keysToActionName16.put("m", "_mcd");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
actionNameToOutput16.put("_mcd", " McDonalds");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") < (0)) || _itemNum == (4))
endGame();
System.out.println("Type m for  McDonalds");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
Scanner

in16 = new Scanner(System.in);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) ||

_itemNum == (4))
endGame();
String input16 = in16.nextLine();
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
while(!keysToActionName16.containsKey

(input16)) {
System.out.println("Invalid input, try again");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
input16 = in16.nextLine();
```

```
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
}
System.out.println("You typed " + input16);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

< (0)) || _itemNum == (4))
endGame();
String action16 = keysToActionName16.get(input16);
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame();
if

(action16.equals("_dcave")) {
{
System.out.println(""+ ("Off to the Dragon Cave to battle Professor

Aho"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum == (4))
endGame

();
}
_dragon_cave();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum ==

(4))
endGame();
}
if(action16.equals("_tai")) {
{
System.out.println(""+ ("Off to Taiwan in search of the

powerful PDP-8"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum ==

(4))
endGame();
}
_taiwan();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) ||

_itemNum == (4))
endGame();
}
if(action16.equals("_u")) {
{
System.out.println(""+ ("Off to UC Berkeley!

... for some reason."));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum

== (4))
endGame();
}
_ucb();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) ||

_itemNum == (4))
endGame();
}
if(action16.equals("_mcd")) {
{
System.out.println(""+ ("Off to McDonalds for

a fish burger"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) || _itemNum ==
```

```
(4))
endGame();
}
_mcdonalds();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") < (0)) ||

_itemNum == (4))
endGame();
}
}
}
endGame();
}
//intdecinit
int _moment;
//start funtion
public void

_dragon_cave() {
currentLocation = "_dragon_cave";
while (!((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))){
{
if(_moment == (0)) {
{
System.out.println(""+

("You are in Prof. Aho's dragon cave!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("Aho is ready to fight you.. You must

fight him to get his dragon book!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("... or maybe not?..."));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (_moment =

(1));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else {
//Empty stmt
}
Map<String,String> keysToActionName11 = new HashMap<String, String>();
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String, String> actionNameToOutput11 = new HashMap<String, String>();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println

("CHOOSE AN ACTION:");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum

== (4))
endGame();
```

```
keysToActionName11.put("f", "_fight");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput11.put("_fight",

"fight");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type f for fight");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName11.put("r", "_runaway");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput11.put("_runaway", "run away");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type r for run away");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName11.put("a", "_ask");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
actionNameToOutput11.put("_ask", " ask nicely");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println

("Type a for  ask nicely");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
Scanner in11 = new Scanner(System.in);
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String input11 = in11.nextLine();
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
while(!

keysToActionName11.containsKey(input11)) {
System.out.println("Invalid input, try again");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
input11

= in11.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
```

```
}
System.out.println("You typed " + input11);
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String action11 =

keysToActionName11.get(input11);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
if(action11.equals("_fight")) {
{
int num = r.nextInt(100);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
if(num >= 0 && num <

70) {
{
System.out.println(""+ ("You say: "));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("behold Aho, my shiny laser

pointer"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt("_prof_aho", Type.CHARACTER, "_life", (entityHasInt("_prof_aho",

Type.CHARACTER, "_life") - entityHasInt("_laser", Type.ITEM, "_damage"))));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+

("You caused Prof. Aho a damage of:"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
System.out.println(""+ (entityHasInt("_laser", Type.ITEM,

"_damage")));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
if((entityHasInt("_prof_aho", Type.CHARACTER, "_life") <= (1))) {
{
System.out.println(""+

("You have defeated Aho... well done!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
_prof_aho.removeItem ("_dragon_book", currentLocation,

locations);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else {
//Empty stmt
}
```

```
}
}
if(num >= 70 && num < 90) {
{
System.out.println(""+ ("As he hits you

Prof. Aho lets you know:"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println(""+ (entityHasString("_prof_aho", Type.CHARACTER,

"_slogan")));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt("_prof_edwards", Type.CHARACTER, "_life", (entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") - entityHasInt("_dragon_book", Type.ITEM,
"_damage"))));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("Prof Aho hurt you this much:"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ (entityHasInt

("_dragon_book", Type.ITEM, "_damage")));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
}
}
if(num >= 90 && num < 100) {
{
System.out.println(""+ ("wuah wuah

wuah"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("Aho surrenders."));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("YOu have won

the dragon book"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
_prof_aho.removeItem ("_dragon_book", currentLocation, locations);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
if(isTrue

(entityExistsItem("_dragon_cave", Type.LOCATION, "_dragon_book"))) {
{
System.out.println(""+ ("In

victory you say your famous slogan:"));
```

```
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
System.out.println(""+ (entityHasString("_prof_edwards",

Type.CHARACTER, "_slogan")));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println(""+ ("Would you like to pick up the dragon book?"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String,String> keysToActionName3 = new HashMap<String, String>();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String, String>

actionNameToOutput3 = new HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("CHOOSE AN ACTION:");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName3.put("g", "_grabbook");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
actionNameToOutput3.put("_grabbook", "grab");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println

("Type g for grab");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
keysToActionName3.put("n", "_notgrab");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput3.put("_notgrab", "not

grab");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
System.out.println("Type n for not grab");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
Scanner in3 = new Scanner(System.in);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
```

```
String input3 =

in3.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
while(!keysToActionName3.containsKey(input3)) {
System.out.println("Invalid input, try

again");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
input3 = in3.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0))

|| _itemNum == (4))
endGame();
}
System.out.println("You typed " + input3);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String action3 =

keysToActionName3.get(input3);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
if(action3.equals("_grabbook")) {
{
System.out.println(""+ ("You have the

dragon book!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
_prof_edwards.addItem ("_dragon_book", currentLocation, locations);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
if(_itemNum == (3)) {
{
System.out.println(""+ ("You did it!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You have collected al four items"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("With them you build the most premiums compiler eveeeeer!!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else

{
//Empty stmt
}
dummy = (_itemNum = (_itemNum + (1)));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
```

```
System.out.println(""+ ("Lets go back to columbia"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
_columbia();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
if(action3.equals("_notgrab")) {
{
System.out.println(""+ ("You dont have the dragon

book!!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
_prof_aho.addItem ("_dragon_book", currentLocation, locations);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
_dragon_cave();
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
else

{
//Empty stmt
}
}
_dragon_cave();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
}
if(action11.equals("_runaway")) {
{
System.out.println(""+ ("You run away to

columbia.. aaaawww"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum

== (4))
endGame();
dummy = (_moment = (0));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
}
_columbia();
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
}
if(action11.equals("_ask")) {
{
if(isTrue

(entityExistsItem("_prof_aho", Type.CHARACTER, "_dragon_book"))) {
{
_prof_aho.removeItem
```

```
("_dragon_book", currentLocation, locations);
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
_prof_edwards.addItem ("_dragon_book", currentLocation,

locations);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("He actually just gave you the dragon book!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+

("... after 3000 hours of telling you about Awk..."));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
if(_itemNum == (3)) {
{
System.out.println(""+ ("You did it!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You have collected all four items"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("With them you build the most premiums compiler eveeeeer!!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else

{
//Empty stmt
}
dummy = (_itemNum = (_itemNum + (1)));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("Anyway, time to go back to

columbia and choose the next location"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
}
}
else {
System.out.println(""+ ("The professor does not have the

dragon book!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
}
}
_columbia();
```

```
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
}
}
}
endGame();
}
//intdec
int _attack;
//start funtion
public void _ucb() {
currentLocation = "_ucb";
while (!((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))){
{
if((entityHasInt("_bill_gates", Type.CHARACTER, "_life") <= (0)) || (entityHasInt

("_plt_mob", Type.CHARACTER, "_life") <= (0))) {
{
if(isTrue(entityExistsItem("_ucb", Type.LOCATION,

"_not_windows_computer"))) {
{
System.out.println(""+ ("You have won!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+

("You have received the not-windows computer!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
_prof_edwards.addItem ("_not_windows_computer",

currentLocation, locations);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
if(_itemNum == (3)) {
{
System.out.println(""+ ("You did it!!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You have collected all four items"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("With them you

build the most premiums compiler eveeeeer!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else {
//Empty stmt
}
dummy = (_itemNum = (_itemNum +

(1)));
```

```
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
System.out.println(""+ ("Time to return to Columbia"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String,String> keysToActionName15 =

new HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
Map<String, String> actionNameToOutput15 = new HashMap<String, String>();
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("CHOOSE AN ACTION:");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
keysToActionName15.put("c", "_col");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput15.put("_col", "Columbia");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type c for Columbia");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Scanner

in15 = new Scanner(System.in);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
String input15 = in15.nextLine();
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
while(!keysToActionName15.containsKey

(input15)) {
System.out.println("Invalid input, try again");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
input15 = in15.nextLine();
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
System.out.println("You typed " + input15);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
```

```
String action15 = keysToActionName15.get(input15);
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
if

(action15.equals("_col")) {
//Empty stmt
_columbia();
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
else {
{
System.out.println(""+ ("You have what you

needed from here"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
System.out.println(""+ ("Time to return to Columbia"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String,String>

keysToActionName2 = new HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
Map<String, String> actionNameToOutput2 = new

HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println("CHOOSE AN ACTION:");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName2.put("c", "_col");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput2.put("_col", "Columbia");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type c for Columbia");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Scanner

in2 = new Scanner(System.in);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
```

```
String input2 = in2.nextLine();
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
while(!keysToActionName2.containsKey

(input2)) {
System.out.println("Invalid input, try again");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
input2 = in2.nextLine();
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
System.out.println("You typed " + input2);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
String action2 = keysToActionName2.get(input2);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
if(action2.equals

("_col")) {
//Empty stmt
_columbia();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0))

|| _itemNum == (4))
endGame();
}
}
}
}
}
else {
//Empty stmt
}
if(_enemy == (0)) {
{
int num = r.nextInt(100);
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
if(num

>= 0 && num < 50) {
dummy = (_enemy = (1));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
}
if(num >= 50 && num < 100) {
dummy = (_enemy = (2));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
```

```
else

{
if(_enemy == (1)) {
{
System.out.println(""+ ("Get ready to fight Bill Gates!!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
int num = r.nextInt

(100);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
if(num >= 0 && num < 40) {
{
dummy = (_attack = (1));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("Bill throws

Microsoft Office at your head!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0))

|| _itemNum == (4))
endGame();
}
}
if(num >= 40 && num < 70) {
{
dummy = (_attack = (2));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+

("Bill throws Windows 7 at your feet!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
}
}
if(num >= 70 && num < 100) {
{
dummy = (_attack = (3));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("Bill is trying to tell you the new features of c#!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
Map<String,String>

keysToActionName20 = new HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
Map<String, String> actionNameToOutput20 = new
```

```
HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println("CHOOSE AN ACTION:");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName20.put("d", "_duck");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput20.put("_duck", "duck");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
System.out.println("Type d for duck");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName20.put("j", "_jump");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
actionNameToOutput20.put("_jump", "jump");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println

("Type j for jump");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
keysToActionName20.put("c", "_cover");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput20.put("_cover", "cover your ears");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type c for cover your ears");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
Scanner in20 = new Scanner(System.in);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String input20 =

in20.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
while(!keysToActionName20.containsKey(input20)) {
System.out.println("Invalid input, try
```

```
again");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
input20 = in20.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
}
System.out.println("You typed " + input20);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String action20 =

keysToActionName20.get(input20);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
if(action20.equals("_duck")) {
{
if(_attack == (1)) {
{
System.out.println(""+

("You escaped the horror!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println(""+ ("Gates is hyperventilating!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt

("_bill_gates", Type.CHARACTER, "_life", (entityHasInt("_bill_gates", Type.CHARACTER,
"_life") -

(1))));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
}
}
else {
if(_attack == (2)) {
{
System.out.println(""+ ("You got hit by windows.. you will never

function correctly again!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt

("_prof_edwards", Type.CHARACTER, "_life", (entityHasInt("_prof_edwards", Type.CHARACTER,
"_life") -

(5))));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
```

```
endGame

();
}
}
else {
{
System.out.println(""+ ("You heard it all... You are utterly confused!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt("_prof_edwards",

Type.CHARACTER, "_life", (entityHasInt("_prof_edwards", Type.CHARACTER, "_life") -
(5))));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
}
_ucb();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
}
if(action20.equals("_jump")) {
{
if(_attack == (2)) {
{
System.out.println(""+ ("You escaped that

which makes many stumble and crack their heards!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("Gates is sweating so much!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy

= (entitySetInt("_bill_gates", Type.CHARACTER, "_life", (entityHasInt("_bill_gates",
Type.CHARACTER,

"_life") - (1))));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
}
}
else {
if(_attack == (1)) {
{
System.out.println(""+ ("You got hit by office square in

the face... just like every user!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
```

```
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy =

(entitySetInt("_prof_edwards", Type.CHARACTER, "_life", (entityHasInt("_prof_edwards",
Type.CHARACTER,

"_life") - (5))));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
}
}
else {
{
System.out.println(""+ ("You heard it all... You are utterly confused!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt("_prof_edwards",

Type.CHARACTER, "_life", (entityHasInt("_prof_edwards", Type.CHARACTER, "_life") -
(5))));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
}
_ucb();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
}
if(action20.equals("_cover")) {
{
if(_attack == (3)) {
{
System.out.println(""+ ("Good! You would have

never been able to make a good compiler again!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("Gates looks dizzy!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy =

(entitySetInt("_bill_gates", Type.CHARACTER, "_life", (entityHasInt("_bill_gates",
Type.CHARACTER,

"_life") - (1))));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
```

```
}
}
else {
if(_attack == (1)) {
{
System.out.println(""+ ("You got hit by office square in

the face... just like every user!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy =

(entitySetInt("_prof_edwards", Type.CHARACTER, "_life", (entityHasInt("_prof_edwards",
Type.CHARACTER,

"_life") - (5))));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
}
}
else {
{
System.out.println(""+ ("You got hit by windows.. you will never function

correctly again!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt("_prof_edwards",

Type.CHARACTER, "_life", (entityHasInt("_prof_edwards", Type.CHARACTER, "_life") -
(5))));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
}
_ucb();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
}
}
}
else {
{
System.out.println(""+ ("Get ready to fight a huge PLT mob!!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
int num = r.nextInt
```

```
(100);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
if(num >= 0 && num < 40) {
{
dummy = (_attack = (1));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("Someone throws

a C Reference Manual at you!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
}
}
if(num >= 40 && num < 70) {
{
dummy = (_attack = (2));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+

("someone is trying to throw the Ocaml camel at your feet!!"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
if(num >= 70 && num < 100) {
{
dummy =

(_attack = (3));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
System.out.println(""+ ("They are all screaming about a new language that is scanned

backwards!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
Map<String,String> keysToActionName20 = new HashMap<String, String>();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String, String>

actionNameToOutput20 = new HashMap<String, String>();
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("CHOOSE AN ACTION:");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName20.put("d", "_duck");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
```

```
endGame();
actionNameToOutput20.put("_duck", "duck");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println

("Type d for duck");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
keysToActionName20.put("j", "_jump");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput20.put("_jump", "jump");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type j for jump");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
keysToActionName20.put("c", "_cover");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput20.put("_cover", "cover your ears");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type c for cover

your ears");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Scanner in20 = new Scanner(System.in);
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
String input20 = in20.nextLine();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
while(!

keysToActionName20.containsKey(input20)) {
System.out.println("Invalid input, try again");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
input20

= in20.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
}
System.out.println("You typed " + input20);
if ((entityHasInt("_prof_edwards",
```

```
Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String action20 =

keysToActionName20.get(input20);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
if(action20.equals("_duck")) {
{
if(_attack == (1)) {
{
System.out.println(""+

("You escaped the horror!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println(""+ ("They are getting tired!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt

("_plt_mob", Type.CHARACTER, "_life", (entityHasInt("_plt_mob", Type.CHARACTER, "_life")
- (1))));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else

{
if(_attack == (2)) {
{
System.out.println(""+ ("You got hit by the camel.. you appreciate O'Caml a

little less!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt("_prof_edwards",

Type.CHARACTER, "_life", (entityHasInt("_prof_edwards", Type.CHARACTER, "_life") -
entityHasInt

("_plt_mob", Type.CHARACTER, "_damage"))));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
}
}
else {
{
System.out.println(""+ ("You heard it all... You are

utterly confused!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum

== (4))
```

```
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt

("_prof_edwards", Type.CHARACTER, "_life", (entityHasInt("_prof_edwards", Type.CHARACTER,
"_life") -

entityHasInt("_plt_mob", Type.CHARACTER, "_damage"))));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
}
_ucb();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
if(action20.equals

("_jump")) {
{
if(_attack == (2)) {
{
System.out.println(""+ ("You escaped that which makes many stumble

and crack their heards!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println(""+ ("The mob is getting bored!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt

("_plt_mob", Type.CHARACTER, "_life", (entityHasInt("_plt_mob", Type.CHARACTER, "_life")
- (1))));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else

{
if(_attack == (1)) {
{
System.out.println(""+ ("You got hit by the manual square in the face... your

brain is now a big null pointer!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
```

```
dummy =

(entitySetInt("_prof_edwards", Type.CHARACTER, "_life", (entityHasInt("_prof_edwards",
Type.CHARACTER,

"_life") - entityHasInt("_plt_mob", Type.CHARACTER, "_damage"))));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else {
{
System.out.println(""+ ("You

heard it all... You are utterly confused!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy =

(entitySetInt("_prof_edwards", Type.CHARACTER, "_life", (entityHasInt("_prof_edwards",
Type.CHARACTER,

"_life") - entityHasInt("_plt_mob", Type.CHARACTER, "_damage"))));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
}
_ucb();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
if(action20.equals

("_cover")) {
{
if(_attack == (3)) {
{
System.out.println(""+ ("Good! You would have never been able to

make a good compiler again!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println(""+ ("Every head in that mob hurts!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt

("_bill_gates", Type.CHARACTER, "_life", (entityHasInt("_bill_gates", Type.CHARACTER,
"_life") -

(1))));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
```

```
endGame

();
}
}
else {
if(_attack == (1)) {
{
System.out.println(""+ ("You got hit by the manual square in the

face... your brain is now a big null pointer!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You loose 5 life points!!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy =

(entitySetInt("_prof_edwards", Type.CHARACTER, "_life", (entityHasInt("_prof_edwards",
Type.CHARACTER,

"_life") - entityHasInt("_plt_mob", Type.CHARACTER, "_damage"))));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else {
{
System.out.println(""+ ("You

got hit by the camel.. you appreciate O'Caml a little less!"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You loose 5

life points!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
dummy = (entitySetInt("_prof_edwards", Type.CHARACTER, "_life", (entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") - entityHasInt("_plt_mob", Type.CHARACTER,
"_damage"))));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
}
_ucb();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
}
}
}
}
}
}
endGame();
```

```
}
//start funtion
public void _mcdonalds() {
currentLocation = "_mcdonalds";
while (!

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))){
{
if(isTrue

(entityExistsItem("_mcdonalds", Type.LOCATION, "_fish_sandwich"))) {
{
System.out.println(""+ ("Ronald

gives you a riddle!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum

== (4))
endGame();
System.out.println(""+ ("What Burger was featured in the greatest advertising

campaign ever?"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
Map<String,String> keysToActionName2 = new HashMap<String, String>();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String, String>

actionNameToOutput2 = new HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("CHOOSE AN ACTION:");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName2.put("f", "_fish");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
actionNameToOutput2.put("_fish", " fish sandwitch");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type f for  fish sandwitch");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName2.put("b", "_burger");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput2.put("_burger", " Big Mac");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type b for  Big Mac");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
```

```
endGame();
keysToActionName2.put("c", "_caesar");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
actionNameToOutput2.put("_caesar", " Caesar Salad");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type c for  Caesar Salad");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
Scanner in2 = new Scanner(System.in);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String input2 =

in2.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
while(!keysToActionName2.containsKey(input2)) {
System.out.println("Invalid input, try

again");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
input2 = in2.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0))

|| _itemNum == (4))
endGame();
}
System.out.println("You typed " + input2);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String action2 =

keysToActionName2.get(input2);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
if(action2.equals("_fish")) {
{
System.out.println(""+ ("Yes, you do believe

in magic!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("YOu have earned a  fish Burger"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+

("Lets go back to Columbia to keep on with the adventure!!"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
_prof_edwards.addItem ("_fish_sandwich",

currentLocation, locations);
```

```
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
if(_itemNum == (3)) {
{
System.out.println(""+ ("You did it!!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You have collected all four items"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("With them you

build the most premiums compiler eveeeeer!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else {
//Empty stmt
}
dummy = (_itemNum = (_itemNum +

(1)));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
}
_columbia();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
}
if(action2.equals("_burger")) {
System.out.println(""+ ("no... come on... you were

there!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
_mcdonalds();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum

== (4))
endGame();
}
if(action2.equals("_caesar")) {
System.out.println(""+ ("Thats not even a

burger!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
_mcdonalds();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum

== (4))
endGame();
}
}
}
else {
{
System.out.println(""+ ("you have collected what you needed here"));
```

```
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String,String> keysToActionName1 = new HashMap<String, String>();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String, String>

actionNameToOutput1 = new HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("CHOOSE AN ACTION:");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName1.put("c", "_col");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0))

|| _itemNum == (4))
endGame();
actionNameToOutput1.put("_col", "Columbia");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println

("Type c for Columbia");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
Scanner in1 = new Scanner(System.in);
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String input1 = in1.nextLine();
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
while(!

keysToActionName1.containsKey(input1)) {
System.out.println("Invalid input, try again");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
input1 =

in1.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
System.out.println("You typed " + input1);
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String action1 = keysToActionName1.get

(input1);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
```

```
if(action1.equals("_col")) {
//Empty stmt
_columbia();
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
}
}
endGame();
}
//intdecinit
int

_narrated;
//start funtion
public void _taiwan() {
currentLocation = "_taiwan";
while (!((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))){
{
if((entityHasInt("_godzilla",

Type.CHARACTER, "_life") <= (0))) {
{
System.out.println(""+ ("...oh no wait, you beat Godzilla!! holy

shhh... wow!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
System.out.println(""+ ("You have received the PDP-8!!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
if(!isTrue

(entityExistsItem("_prof_edwards", Type.CHARACTER, "_PDP_8"))) {
{
_prof_edwards.addItem ("_PDP_8",

currentLocation, locations);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
if(_itemNum == (3)) {
{
System.out.println(""+ ("You did it!!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You have collected all four items"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("With them you

build the most premiums compiler eveeeeer!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
}
```

```
}
else {
//Empty stmt
}
dummy = (_itemNum = (_itemNum +

(1)));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame

();
}
}
else {
//Empty stmt
}
}
}
else {
//Empty stmt
}
if(isTrue(entityExistsItem("_taiwan", Type.LOCATION,

"_PDP_8"))) {
{
if(_narrated == (0)) {
{
System.out.println(""+ ("You have traveled to Taiwan to get the

biggest PDP-8 in the world"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println(""+ ("You need it for its humongous stack!!"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("The problem is, it belongs to Godzilla!!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (_narrated =

(1));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
else {
//Empty stmt
}
System.out.println(""+ ("Godzilla is about to step on you!!"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String,String>

keysToActionName12 = new HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
Map<String, String> actionNameToOutput12 = new

HashMap<String, String>();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
```

```
endGame();
System.out.println("CHOOSE AN ACTION:");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName12.put("s", "_shoot");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput12.put("_shoot", "shooting your laser pointer at his foot");
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println

("Type s for shooting your laser pointer at his foot");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
keysToActionName12.put("d", "_die");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput12.put("_die", "dying squished between its toes");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type d for dying

squished between its toes");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
keysToActionName12.put("t", "_tickle");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput12.put("_tickle",

"tickling the monster's foot");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
System.out.println("Type t for tickling the monster's foot");
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Scanner

in12 = new Scanner(System.in);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
String input12 = in12.nextLine();
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
while(!keysToActionName12.containsKey

(input12)) {
```

```
System.out.println("Invalid input, try again");
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
input12 = in12.nextLine();
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
System.out.println("You typed " + input12);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
String action12 = keysToActionName12.get(input12);
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
if

(action12.equals("_shoot")) {
{
System.out.println(""+ ("You hit the monster in the foot"));
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy =

(entitySetInt("_godzilla", Type.CHARACTER, "_life", (entityHasInt("_godzilla",
Type.CHARACTER,

"_life") - entityHasInt("_laser", Type.ITEM, "_damage"))));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ (" You did 10

points damage!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum ==

(4))
endGame();
System.out.println(""+ ("Keep fighting!!"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
_taiwan();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
if(action12.equals

("_die")) {
{
System.out.println(""+ ("You have died between its green toes"));
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt

("_prof_edwards", Type.CHARACTER, "_life", (0)));
```

```
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
}
_columbia();
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
if(action12.equals("_tickle")) {
{
System.out.println(""+ ("the monster fell back laughing"));
if ((entityHasInt("_prof_edwards",

Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println(""+ ("You have done

50 damage!!"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
dummy = (entitySetInt("_godzilla", Type.CHARACTER, "_life", (entityHasInt("_godzilla",

Type.CHARACTER, "_life") - (50))));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0))

|| _itemNum == (4))
endGame();
}
_taiwan();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <=

(0)) || _itemNum == (4))
endGame();
}
}
}
else {
{
System.out.println(""+ ("You have the calculator. Time to

return to columbia"));
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum

== (4))
endGame();
Map<String,String> keysToActionName1 = new HashMap<String, String>();
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
Map<String, String> actionNameToOutput1 = new HashMap<String, String>();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println

("CHOOSE AN ACTION:");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum

== (4))
endGame();
keysToActionName1.put("c", "_col");
if ((entityHasInt("_prof_edwards", Type.CHARACTER,

"_life") <= (0)) || _itemNum == (4))
endGame();
actionNameToOutput1.put("_col", "Columbia");
```

```
if

((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
System.out.println("Type c for Columbia");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life")

<= (0)) || _itemNum == (4))
endGame();
Scanner in1 = new Scanner(System.in);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String input1 =

in1.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
while(!keysToActionName1.containsKey(input1)) {
System.out.println("Invalid input, try

again");
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
input1 = in1.nextLine();
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0))

|| _itemNum == (4))
endGame();
}
System.out.println("You typed " + input1);
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
String action1 =

keysToActionName1.get(input1);
if ((entityHasInt("_prof_edwards", Type.CHARACTER, "_life") <= (0)) ||

_itemNum == (4))
endGame();
if(action1.equals("_col")) {
//Empty stmt
_columbia();
if ((entityHasInt

("_prof_edwards", Type.CHARACTER, "_life") <= (0)) || _itemNum == (4))
endGame();
}
}
}
}
}
endGame();
}
}

abstract class Entity {

   Map<String, Integer> intAttrs = new HashMap<String, Integer>();
   Map<String, String> strAttrs = new HashMap<String, String>();

   public void addIntAttr(String name, int value) {
      intAttrs.put(name, value);
   }

   public void addStrAttr(String name, String value) {
      strAttrs.put(name, value);
   }
}
```

```
class Location extends Entity {
   Set<String> characters = new HashSet<String>();
   Set<String> items = new HashSet<String>();

   public void addItem(String name, Map<String, Item> itemses) {
      if(itemses.containsKey(name))
                  items.add(name);
      else
            System.out.println("Error: The item you attempted to add no longer exists");
   }

   public void addItem(String name){
      items.add(name);
      }

   public void removeItem(String name) {
      items.remove(name);
   }

   public void showCharacter(String name, Map<String, Character> characterses) {
      if(characterses.containsKey(name))
      characters.add(name);
      else
            System.out.println("Error: The character you attempted to use no longer
exists");
   }
   public void showCharacter(String name){

      characters.add(name);
      }

   public void hideCharacter(String name) {
      characters.remove(name);
   }
}

class Character extends Entity {
   Set<String> items = new HashSet<String>();

   public void addItem(String name, String locationNow, Map<String, Location> locations){

      if(locations.get(locationNow).items.contains(name)){
            locations.get(locationNow).removeItem(name);
            items.add(name);
      }
      else
            System.out.println("Error: The item you attempted to grab is not in this
location");
}

public void removeItem(String name, String locationNow, Map<String, Location> locations)
{
            if (items.contains(name)){
            items.remove(name);
            locations.get(locationNow).addItem(name);
   }
      else
            System.out.println("Error: The character does not have the item you attempted
to

drop");
}

   public void addItem(String name) {
      items.add(name);
   }

   public void removeItem(String name) {
      items.remove(name);
   }
}
```

```
class Item extends Entity {
}
```

Test Files

Each .next test file is followed by the output that would be found in its reference file. The last three test files have no reference output because they are meant to be run manually.

```
#CleanTests.pl by Danny
#!/usr/bin/perl -w

@testFileDirs = <Tests/*>;

foreach $testFileDir(@testFileDirs) {
    system("rm ". $testFileDir . "/Next" . '\$' . "Type.class");

    @testFileDir = <$testFileDir/*>;
    foreach $testFile(@testFileDir) {
        if(!($testFile =~ /(\.next)$/i) && !($testFile =~ /reference/i)) {
            system("rm " . $testFile);
        }
    }
}
```

```
#RunTests.pl by Danny
#!/usr/bin/perl -w

use File::Compare;
use FileHandle;

system("perl CleanTests.pl");

my @failedTestFiles = ();

@testFileDirs = <Tests/*>;

foreach $testFileDir(@testFileDirs) {
    @testFiles = <$testFileDir/*>;
    foreach $testFile(@testFiles) {
        print $testFile . "\n";

        if($testFile =~ /\.next$/i) {
            #do the compilation
            if(system("./next < " . $testFile . " > " . $testFileDir .
"/Next.java") == 0) {
                #do java compilation
                if(system("javac " . $testFileDir . "/Next.java") == 0) {
                    system("java -classpath " . $testFileDir . " Next > " .
$testFileDir . "/output");
                }
            }

            #compare output with reference
            #if comparison returns 0 they are the same, otherwise they are
not
            #write to log file of ones that are not
            if(File::Compare::compare_text($testFileDir . "/output",
$testFileDir . "/reference")) {
                push(@failedTestFiles, $testFileDir);
            }
        }
    }
}


#write to log file
$fh = FileHandle -> new();
if($fh -> open("> Tests/FailedTests.txt")) {
    foreach $file(@failedTestFiles) {
        $fh -> print($file . "\n");
    }
}
```

```
/*AndOrOperators.next by Danny*/
int stop = 0;
string myString = "go go go";

item myItem1 {(int size = 50)}
item myItem2 {(int anotherSize = 100)}
item myItem3 {(int what = 1000)}

character myCharacter1 {(int life = 1), (myItem1)}
character myCharacter2 {(int id = 123), (myItem2)}
character myCharacter3 {(int health = 4), (myItem3)}

location place {(int count = 10), (myItem1, myItem2), (myCharacter1,
myCharacter2, myCharacter3)}

start place end (stop == 11) {
    if (11 == 11 and place.count == 10 and myItem1.size != 51) then {
        output "True";
    }
    else {
        output "Error";
    }

    if(11 != 11 or place.count == 11 or myItem1.size != 51) then {
        output "True";
    }
    else {
        output "Error";
    }

    if((11 != 11 or place.count == 11 or myItem1.size != 51) and (21 < 1))
then {
        output "Error";
    }
    else {
        output "True";
    }

    stop = 11;
}
```

```
True
True
True
```

```
/*ArithmeticOperators.next by Danny*/
int stop = 0;
string myString = "go go go";

item myItem1 {(int size = 50)}
item myItem2 {(int anotherSize = 100)}
item myItem3 {(int what = 1000)}

character myCharacter1 {(int life = 1), (myItem1)}
character myCharacter2 {(int id = 123), (myItem2)}
character myCharacter3 {(int health = 4), (myItem3)}

location place {(int count = 10), (myItem1, myItem2), (myCharacter1,
myCharacter2, myCharacter3)}

start place end (stop == 11) {
    stop = stop - 11;
    stop = stop + 1;
    stop = stop * 3 + 2;
    stop = stop / 4;
    stop = (stop * 9 / 3 + 1 - 4);
    output stop;

    stop = 11;
}
```

```
/*CharacterDeclarations.next by Ernesto */

item the_greatest_sword_ever {()}
item rubberDuckie {(string squeak = "squeak")}
item columbiaBinder {(int size = 50)}


character noone{(),()}
character me {(),(rubberDuckie)}
character you {(int life = 50),(rubberDuckie)}
character her {(int life),(rubberDuckie)}
character him {(int life, string name="him"),(rubberDuckie, columbiaBinder,
the_greatest_sword_ever)}
character someone{(string name),()}

int count = 1;


location here {(int sizex = 10000, int sizey=9283), (columbiaBinder), ()}

start here end (count == 0){

      output you.life;
      output her.life;
      output him.life;
      output him.name;
      output someone.name;

      count = count - 1;



}
```

50
0
0
him

```
/*ComparisonOperators.next by Danny*/
int stop = 0;
string myString = "go go go";

item myItem1 {(int size = 50)}
item myItem2 {(int anotherSize = 100)}
item myItem3 {(int what = 1000)}

character myCharacter1 {(int life = 1), (myItem1)}
character myCharacter2 {(int id = 123), (myItem2)}
character myCharacter3 {(int health = 4), (myItem3)}

location place {(int count = 10), (myItem1, myItem2), (myCharacter1,
myCharacter2, myCharacter3)}

start place end (stop == 11) {
    if (1 < 2) then {
        output "1 is less than 2";
    }
    else {
        output "comparison problem";
    }

    if (1 > 2) then {
        output "1 is greater than 2";
    }
    else {
        output "1 is less than 2";
    }

    if (1 <= 1) then {
        output "1 is less or equal to 1";
    }
    else {
        output "1 is greater than 1";
    }

    if (1 <= 2) then {
        output "1 is less than or equal to 2";
    }
    else {
        output "2 is less than 1";
    }

    if (1 >= 2) then {
        output "1 is greater than or equal to 2";
    }
    else {
        output "1 is less than 2";
    }

    if (1 >= 1) then {
        output "1 is greater than or equal to 1";
    }
    else {
        output "1 is less than 2";
    }

    if (1 == 1) then {
        output "1 is equal to 1";
    }
    else {
        output "1 is not equal to 1";
    }
```

```
    if (1 != 1) then {
        output "1 not equal to 1";
    }
    else {
        output "1 is equal to 1";
    }

    if ("hi" == "hi") then {
       output "hi is equal to hi";
    }
    else {
        output "hi is not equal to hi";
    }

    if ("hi" != "hi") then {
       output "hi not equal to hi";
    }
    else {
        output "hi is equal to hi";
    }

    stop = 11;
}
```

```
1 is less than 2
1 is less than 2
1 is less or equal to 1
1 is less than or equal to 2
1 is less than 2
1 is greater than or equal to 1
1 is equal to 1
1 is equal to 1
hi is equal to hi
hi is equal to hi
```

```
/*CompoundStatements.next by Morgan*/
int stop = 0;
string myString = "go go go";

item myItem {(int size = 50)}

character myCharacter {(int life = 1), (myItem)}

location place {(int count = 10), (myItem), (myCharacter)}

start place end (place.count == 1) {
    {
      myString = "stop stop stop";
      myCharacter.life = 10;
      stop = myCharacter.life - 9;

    }

    if (myString == "stop stop stop") then {
       output "correct";
    }
    else {
        output "incorrect";
    }

    {
      if (myCharacter.life > 1) then {
            output "correct";
      }
      else {
            output "incorrect";
        }


        place.count = stop;
    }


}
```

correct
correct

```
/*DotOperator.next by Danny*/
int stop = 0;
string myString = "go go go";

item myItem1 {(int size = 50)}
item myItem2 {(int anotherSize = 100)}
item myItem3 {(int what = 1000)}

character myCharacter1 {(int life = 1), (myItem1)}
character myCharacter2 {(int id = 123), (myItem2)}
character myCharacter3 {(int health = 4), (myItem3)}

location place {(int count = 10), (myItem1, myItem2), (myCharacter1,
myCharacter2, myCharacter3)}

start place end (stop == 1) {
      if(exists place.myItem1) then {
          output myItem2.anotherSize;
      }
      else {
          output myItem1.size;
      }

    output myItem3.what;

    stop = 1;
}
```

100
1000

```
/*ExistsOperator.next by Danny*/
int stop = 0;
string myString = "go go go";

item myItem1 {(int size = 50)}
item myItem2 {(int anotherSize = 100)}
item myItem3 {(int what = 1000)}

character myCharacter1 {(int life = 1), (myItem1)}
character myCharacter2 {(int id = 123), (myItem2)}
character myCharacter3 {(int health = 4), (myItem3)}

location place {(int count = 10), (myItem1, myItem2), (myCharacter1,
myCharacter2, myCharacter3)}

start place end (stop == 1) {
      if(exists place.myItem1 == 1) then {
          output "Yes!";
      }
      else {
          output "Something went wrong";
      }

      if(exists place.myCharacter2) then {
          output "Yes!!";
      }
      else {
           output "No :(";
        }

      if (exists myCharacter1.myItem2) then {
          output "Yes";
        }
      else {
           output "Nope";
      }

      if (exists myCharacter3.myItem3) then {
          output "yay";
        }
      else {
           output "oops";
      }

    stop = 1;
}
```

Yes!
Yes!!
Nope
yay

```
/*GrabDrop.next by Ernesto*/

item nothingItem {()}
item the_greatest_sword_ever {(int damage = 100000000)}
item rubberDuckie {(string squeak = "squeak")}
item columbiaBinder {(int size = 50)}
int count = 8;

character xiaowei_the_greatest_man_ever {(int life = 1, int level=99999,
string haha="hahahahaha"), (the_greatest_sword_ever, rubberDuckie)}

location where_is_this_place {(int sizex = 10000, int sizey=9283),
(columbiaBinder, rubberDuckie), (xiaowei_the_greatest_man_ever)}

start where_is_this_place end (count == -1) {


      count = count - 1;
      output count;
      if (count == 6) then drop xiaowei_the_greatest_man_ever.nothingItem;
      else if (count == 5) then grab
xiaowei_the_greatest_man_ever.nothingItem;
      else if (count == 4) then grab
xiaowei_the_greatest_man_ever.columbiaBinder;
      else if (count == 3) then drop
xiaowei_the_greatest_man_ever.the_greatest_sword_ever;
      else if (count == 2) then drop
xiaowei_the_greatest_man_ever.rubberDuckie;
      else if (count == 1) then grab
xiaowei_the_greatest_man_ever.rubberDuckie;
      else ;

      if  (exists xiaowei_the_greatest_man_ever.columbiaBinder) then
            output "binder exists in Xiao";
      else output "Binder does not exists in Xiao";

      if  (exists where_is_this_place.columbiaBinder) then
            output "binder exists in place";
      else output "Binder does not exists in place";

      if  (exists xiaowei_the_greatest_man_ever.rubberDuckie) then
            output "duckie exists in Xiao";
      else output "duckie does not exists in Xiao";

      if  (exists where_is_this_place.rubberDuckie) then
            output "duckie exists in place";
      else output "duckie does not exists in place";

      if  (exists xiaowei_the_greatest_man_ever.the_greatest_sword_ever) then
            output "the_greatest_sword_ever exists in Xiao";
      else output "the_greatest_sword_ever does not exists in Xiao";

      if  (exists where_is_this_place.the_greatest_sword_ever) then
            output "the_greatest_sword_ever exists in place";
      else output "the_greatest_sword_ever does not exists in place";


}
```

```
7
Binder does not exists in Xiao
binder exists in place
duckie exists in Xiao
duckie exists in place
the_greatest_sword_ever exists in Xiao
the_greatest_sword_ever does not exists in place
6
Error: The character does not have the item you attempted to drop
Binder does not exists in Xiao
binder exists in place
duckie exists in Xiao
duckie exists in place
the_greatest_sword_ever exists in Xiao
the_greatest_sword_ever does not exists in place
5
Error: The item you attempted to grab is not in this location
Binder does not exists in Xiao
binder exists in place
duckie exists in Xiao
duckie exists in place
the_greatest_sword_ever exists in Xiao
the_greatest_sword_ever does not exists in place
4
binder exists in Xiao
Binder does not exists in place
duckie exists in Xiao
duckie exists in place
the_greatest_sword_ever exists in Xiao
the_greatest_sword_ever does not exists in place
3
binder exists in Xiao
Binder does not exists in place
duckie exists in Xiao
duckie exists in place
the_greatest_sword_ever does not exists in Xiao
the_greatest_sword_ever exists in place
2
binder exists in Xiao
Binder does not exists in place
duckie does not exists in Xiao
duckie exists in place
the_greatest_sword_ever does not exists in Xiao
the_greatest_sword_ever exists in place
1
binder exists in Xiao
Binder does not exists in place
duckie exists in Xiao
duckie does not exists in place
the_greatest_sword_ever does not exists in Xiao
the_greatest_sword_ever exists in place
0
binder exists in Xiao
Binder does not exists in place
duckie exists in Xiao
duckie does not exists in place
the_greatest_sword_ever does not exists in Xiao
the_greatest_sword_ever exists in place
```

```
/* IntegerDeclarations.next by Ernesto */
location here {(int sizex = 10000, int sizey=9283), (), ()}
int x;
int y = x;
int z = y + 2;
int m = 8 + z;
int n = m + z;
int i = here.sizex;

int count = 1;



start here end (count == 0){


      output x;
      output y;
      output z;
      output m;
      output n;
      output i;
      count = count - 1;
}
```

0
0
2
10
12
10000

```
/*ItemDeclarations.next by Ernesto */
item the_greatest_sword_ever {()}
item rubberDuckie {(string squeak = "squeak")}
item shirt5{(string size = "medium", int number = 20, string color = "red",
int arm = 10)}
item columbiaBinder {(int size = 50)}
item ColumbiaBinder {(int size = 30)}
int count = 1;


location here {(int sizex = 10000, int sizey=9283), (columbiaBinder), ()}

start here end (count == 0){
      output rubberDuckie.squeak;
      output shirt5.size;
      output shirt5.number;
      output shirt5.color;
      output shirt5.arm;
      output columbiaBinder.size;
      output ColumbiaBinder.size;
      count = count - 1;



}
```

squeak
medium
20
red
10
50
30

```
/*Kill.next by Ernesto, Checks if kill removes from all lists*/

item the_greatest_sword_ever {(int damage = 100000000)}
item rubberDuckie {(string squeak = "squeak")}
item columbiaBinder {(int size = 50)}
int count = 4;

character xiaowei_the_greatest_man_ever {(int life = 1, int level=99999,
string haha="hahahahaha"),

(the_greatest_sword_ever, rubberDuckie, columbiaBinder)}

location where_is_this_place {(int sizex = 10000, int sizey=9283),
(columbiaBinder, rubberDuckie),

(xiaowei_the_greatest_man_ever)}

start where_is_this_place end (count == -1) {


      count = count - 1;
      output count;
      if (count == 2) then kill columbiaBinder;
      else if (count == 1) then kill xiaowei_the_greatest_man_ever;
      else ;

      if  (exists xiaowei_the_greatest_man_ever.columbiaBinder) then
            output "binder exists in Xiao";
      else output "Binder no longer exists in Xiao";

      if  (exists where_is_this_place.columbiaBinder) then
            output "binder exists in place";
      else output "Binder no longer exists in place";

      if  (exists xiaowei_the_greatest_man_ever.rubberDuckie) then
            output "duckie exists in Xiao";
      else output "duckie no longer exists in Xiao";

      if  (exists where_is_this_place.rubberDuckie) then
            output "duckie exists in place";
      else output "duckie no longer exists in place";

      if  (exists where_is_this_place.xiaowei_the_greatest_man_ever) then
            output "Xiao is alive";
      else output "Xiao is dead!!";


}
```

```
3
binder exists in Xiao
binder exists in place
duckie exists in Xiao
duckie exists in place
Xiao is alive
2
Binder no longer exists in Xiao
Binder no longer exists in place
duckie exists in Xiao
duckie exists in place
Xiao is alive
1
Binder no longer exists in Xiao
Binder no longer exists in place
duckie no longer exists in Xiao
duckie exists in place
Xiao is dead!!
0
Binder no longer exists in Xiao
Binder no longer exists in place
duckie no longer exists in Xiao
duckie exists in place
Xiao is dead!!
```

```
/*LocationDeclarations.next by Ernesto, Testing naming conventions. Forms of
declaration of Locations

*/

item the_greatest_sword_ever {()}
item rubberDuckie {(string squeak = "squeak")}
item columbiaBinder {(int size = 50)}


character noone{(),()}
character me {(),(rubberDuckie)}
character you {(int life = 50),(rubberDuckie)}
character her {(int life),(rubberDuckie)}
character him {(int life, string name="him"),(rubberDuckie, columbiaBinder,
the_greatest_sword_ever)}
character someone{(string name),()}

int count = 1;


location limbo {(),(),()}
location empty {(string ew6 = "not much"),( rubberDuckie,
columbiaBinder),(me,you,her)}
location cool{(), (the_greatest_sword_ever), (him, someone,her)}
location COOL{(int look = 100, int size = 5, string name =
"COOL"),(),(her,him,me,someone,you)}

location here {(int sizex = 10000, int sizey=9283), (columbiaBinder), ()}

start here end (count == 0){

      output empty.ew6;
      output COOL.look;
      output COOL.size;
      output COOL.name;
      output here.sizex;
      output here.sizey;

      count = count - 1;



}
start limbo end (0);
start empty end (0);
start cool end (1);
start COOL end (1);
```

not much
100
5
COOL
10000
9283

```
/*OutputStmt.Next by Pri: Xiao Sec:Ernesto */

item i1 {()}
item i2 {(string squeak = "squeak")}
item i3 {(int size = 50)}


character c1 {(int life = 10),()}
character c2 {(int life = 10),()}

int count = 1;

location l1 {(int sizex = 10000, int sizey=9283), (i1), (c1)}
location l2 {(),(),()}


start l1 end (count == -1 or c1.life == 0){

      output "output test";
      output count;
      output 777777;
      output count+count;
      output count-count;
      output count*count;
      output count/1;
      output count=count;
      output count or count;
      output count and count;
      output not count;
      output l1.sizex;
      output l1.sizex = 100;
      output exists l2.i1;
      output -count;
      output count < count+1;
      output count > count+1;
      output count >= count+1;
      output count <= count+1;
      output count == count;
      output count != count;
      output count = count - 1;

}

start l2 end (count ==0);
```

```
output test
1
777777
2
0
1
1
1
1
1
0
10000
100
0
-1
1
0
0
1
1
0
0
output test
0
777777
0
0
0
0
0
0
0
1
100
100
0
0
1
0
0
1
1
0
-1
```

```
/*ShowAfterKill.next by Ernesto, Need to check if hide and show are being
used on character or item*/

item the_greatest_sword_ever {(int damage = 100000000)}
item rubberDuckie {(string squeak = "squeak")}
item columbiaBinder {(int size = 50)}
int count = 4;
string hi = "Hello World";
character xiaowei_the_greatest_man_ever {(int life = 1, int level=99999,
string haha="hahahahaha"),

(the_greatest_sword_ever, rubberDuckie)}

location where_is_this_place {(int sizex = 10000, int sizey=9283),
(columbiaBinder),

(xiaowei_the_greatest_man_ever)}

start where_is_this_place end (count == -1) {

      output hi;
      count = count - 1;
      if (exists where_is_this_place.columbiaBinder) then {
            output "Give it to Ernesto";
            hide where_is_this_place.columbiaBinder;}

      else {output "Take morgan's and put it back";
            show where_is_this_place.columbiaBinder;}

      if (exists where_is_this_place.xiaowei_the_greatest_man_ever) then {
            output "Xiao has left the building";
            hide where_is_this_place.xiaowei_the_greatest_man_ever;}

      else {output "He is back!!";
            show where_is_this_place.xiaowei_the_greatest_man_ever;}
      output  count;

      if (count < 3) then{
            kill columbiaBinder;
            kill xiaowei_the_greatest_man_ever;
      }
      else ;

}
```

```
Hello World
Give it to Ernesto
Xiao has left the building
3
Hello World
Take morgan's and put it back
He is back!!
2
Hello World
Take morgan's and put it back
Error: The item you attempted to add no longer exists
He is back!!
Error: The character you attempted to use no longer exists
1
Hello World
Take morgan's and put it back
Error: The item you attempted to add no longer exists
He is back!!
Error: The character you attempted to use no longer exists
0
Hello World
```

```
/*StringDeclarations.next by Ernesto,  Forms of declaration of strings */
location here {(string name = "here", int sizey=9283), (), ()}
string x;
string y = x;
string z = "HI there";
string m = "You, ";
string n = z;

string i = here.name;

int count = 1;



start here end (count == 0){


      output x;
      output y;
      output z;
      output m;
      output n;
      output i;
      count = count - 1;
}
```

HI there
You,
HI there
here

```
/*types.next by Danny*/
int stop = 0;
string myString = "go go go";

item myItem1 {(int size = 50)}
item myItem2 {(int anotherSize = 100)}
item myItem3 {(int what = 1000)}

character myCharacter1 {(int life = 1), (myItem1)}
character myCharacter2 {(int id = 123), (myItem2)}
character myCharacter3 {(int health = 4), (myItem3)}

location place {(int count = 10), (myItem1, myItem2), (myCharacter1,
myCharacter2, myCharacter3)}

start place end (stop == 1) {
    output stop;
    output myString;

    output myItem1.size;

    output myCharacter1.life;

    output place.count;

    stop = 1;
}
```

0
go go go
50
1
10

```
/*UnaryOperators.next by Danny*/
int stop = 0;
string myString = "go go go";

item myItem1 {(int size = 50)}
item myItem2 {(int anotherSize = 100)}
item myItem3 {(int what = 1000)}

character myCharacter1 {(int life = 1), (myItem1)}
character myCharacter2 {(int id = 123), (myItem2)}
character myCharacter3 {(int health = 4), (myItem3)}

location place {(int count = 10), (myItem1, myItem2), (myCharacter1,
myCharacter2, myCharacter3)}

start place end (stop == 11) {
    stop = 1;
    stop = -stop;
    output stop;
    if(not myCharacter1.life) then {
        output "Error";
    }
    else {
        output "True";
    }

    stop = 11;
}
```

```
-1
True
```

```
/* ChooseStmt.next by Xiao */

item i1 {()}
item i2 {(string squeak = "squeak")}
item i3 {(int size = 50)}


character c1 {(int life = 10),()}
character c2 {(int life = 10),()}

int count = 10;

location l1 {(int sizex = 10000, int sizey=9283), (), ()}
location l2 {(),(),()}


start l1 end (count == 0 or c1.life == 0){

      choose (a,"a","a") (b,"b","b") {
            when a
            {
                  output "c1.life";
                  output c1.life = c1.life -1;
            } next l1
            when b {
                  output "count";
                  output count = count -1;
            }
            next l1
      }
}

start l2 end (count == 0);
```

```
/* ProbStmt.next by Xiao */

item i1 {()}
item i2 {(string squeak = "squeak")}
item i3 {(int size = 50)}


character c1 {(int life = 10),()}
character c2 {(int life = 10),()}

int count = 10;

location l1 {(int sizex = 10000, int sizey=9283), (), ()}
location l2 {(),(),()}


start l1 end (count == 0 or c1.life == 0){

      [?
            prob 30 {
                  output count = count-1;
            }
            prob 20 {
                  output count = count+1;
            }
            prob 30 {
                  output c1.life = c1.life-1;
            }
            prob 20 {
                  output c1.life = c1.life+1;
            }
      ?]
}
start l2 end (0);
```

```
/* StartStmt.next by Xiao */

item i1 {()}
item i2 {(string squeak = "squeak")}
item i3 {(int size = 50)}


character c1 {(int life = 10),()}
character c2 {(int life = 10),()}

int count = 10;

location l1 {(int sizex = 10000, int sizey=9283), (), ()}
location l2 {(),(),()}
location l3 {(),(),()}


start l1 end (count == 0 or c1.life == 0){
      output "in l1";
      choose (a,"1","1") (b,"2","2") (c,"3","3"){
            when a
            {
                  output "go to l1";
            } next l1
            when b
            {
                  output "go to l2";
            } next l2
            when c
            {
                  output "go to l3";
            } next l3
      }
}

start l2 end (count = 0 or c1.life = 0){
      output "in l2";
      output count;
      choose (a,"1","1") (b,"2","2") (c,"3","3"){
            when a
            {
                  output "go to l1";
            } next l1
            when b
            {
                  output "go to l2";
            } next l2
            when c
            {
                  output "go to l3";
            } next l3
      }

}

start l3 end (count == 0 or c1.life == 0) {
      output "in l3";
      choose (a,"1","1") (b,"2","2") (c,"3","3"){
            when a
            {
                  output "go to l1";
            } next l1
            when b
            {
```

```
                output "go to l2";
            } next l2
            when c
            {
                output "go to l3";
            } next l3
        }
}
```