Professor Stephen Edwards
COMS W4115, Programming Languages & Translators
Fall 2016

Java +-

-or-

Java but worse but also better

Zeynep Ejder - Language Guru
Ashley Daguanno - Manager
Anna Wen - Tester
Amal Abid - Systems Architect
Tin Nilar Hlaing - Systems Architect

Java+- Introduction

Java+- is a general purpose, object oriented-programming language that looks and feels a lot like our most favorite programming language, Java and compiles down to LLVM. .

MOTIVATION

Our goal in writing was is essentially a limited version of Java was not to reinvent the wheel but to understand what's going on under the hood of Java.

ADDED FUNCTIONALITY

We aimed to incorporate tuples into our language in order to merge the functionality of tuples with the familiarity of a language that most programmers are already well oriented with.

FEATURES

What was brought over from Java?

- Types, Operators, Classes & Objects, Loops, Control Statements, Scoping

What was not?

- Inheritance, Garbage Collection

What was added?

- Tuples!

Software Technologies Used

Github

OCaml

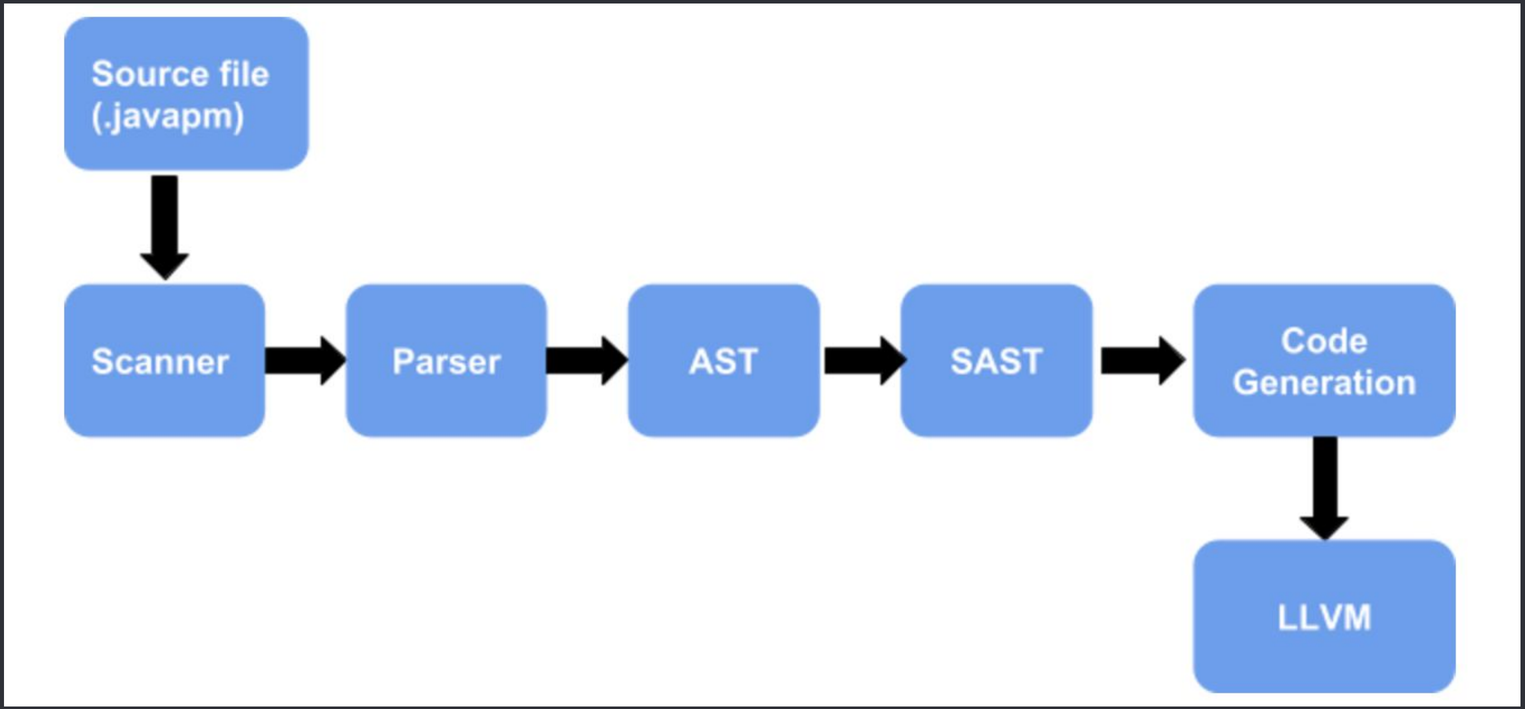
LLVM

Bash

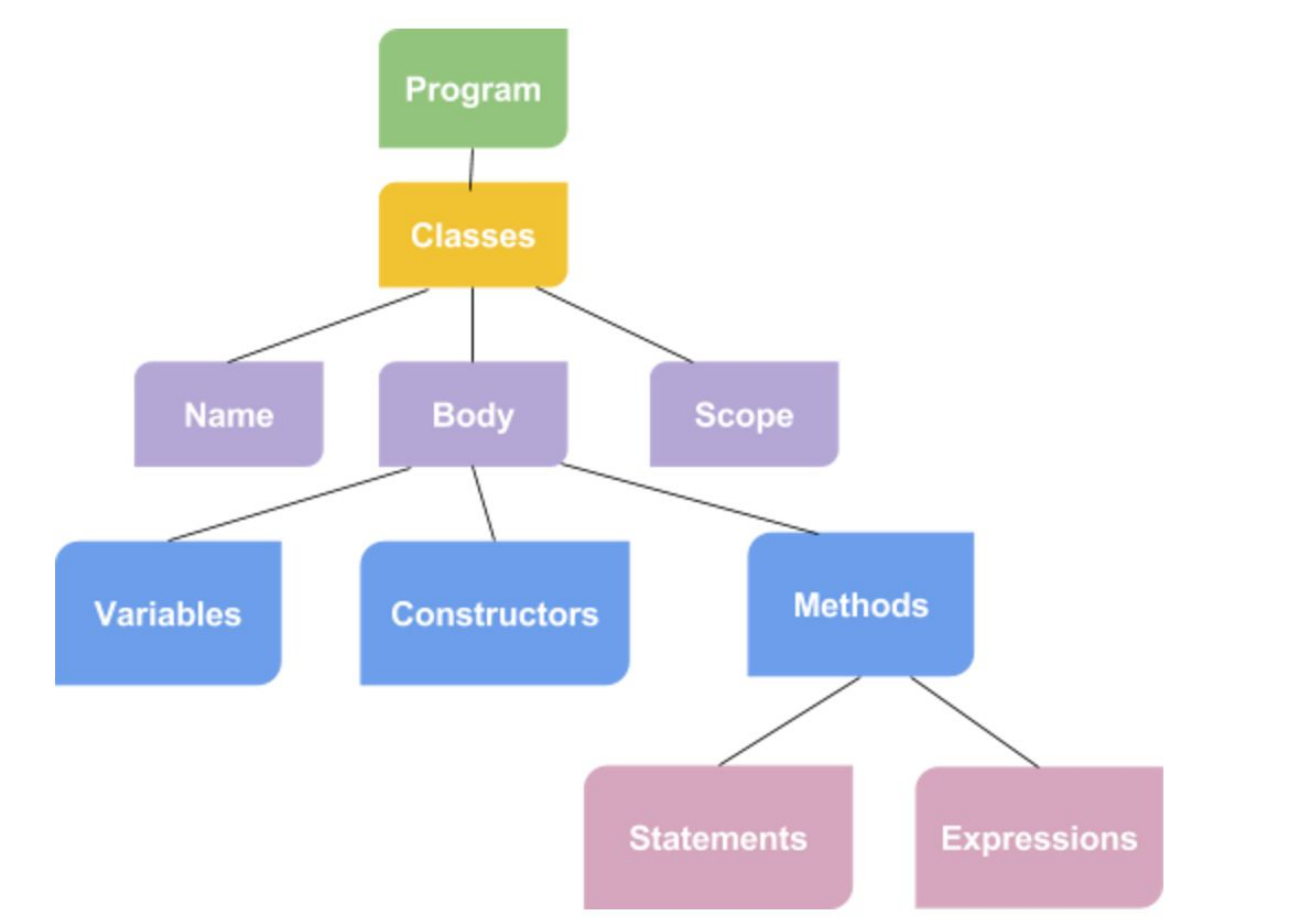
Ubuntu

OSX

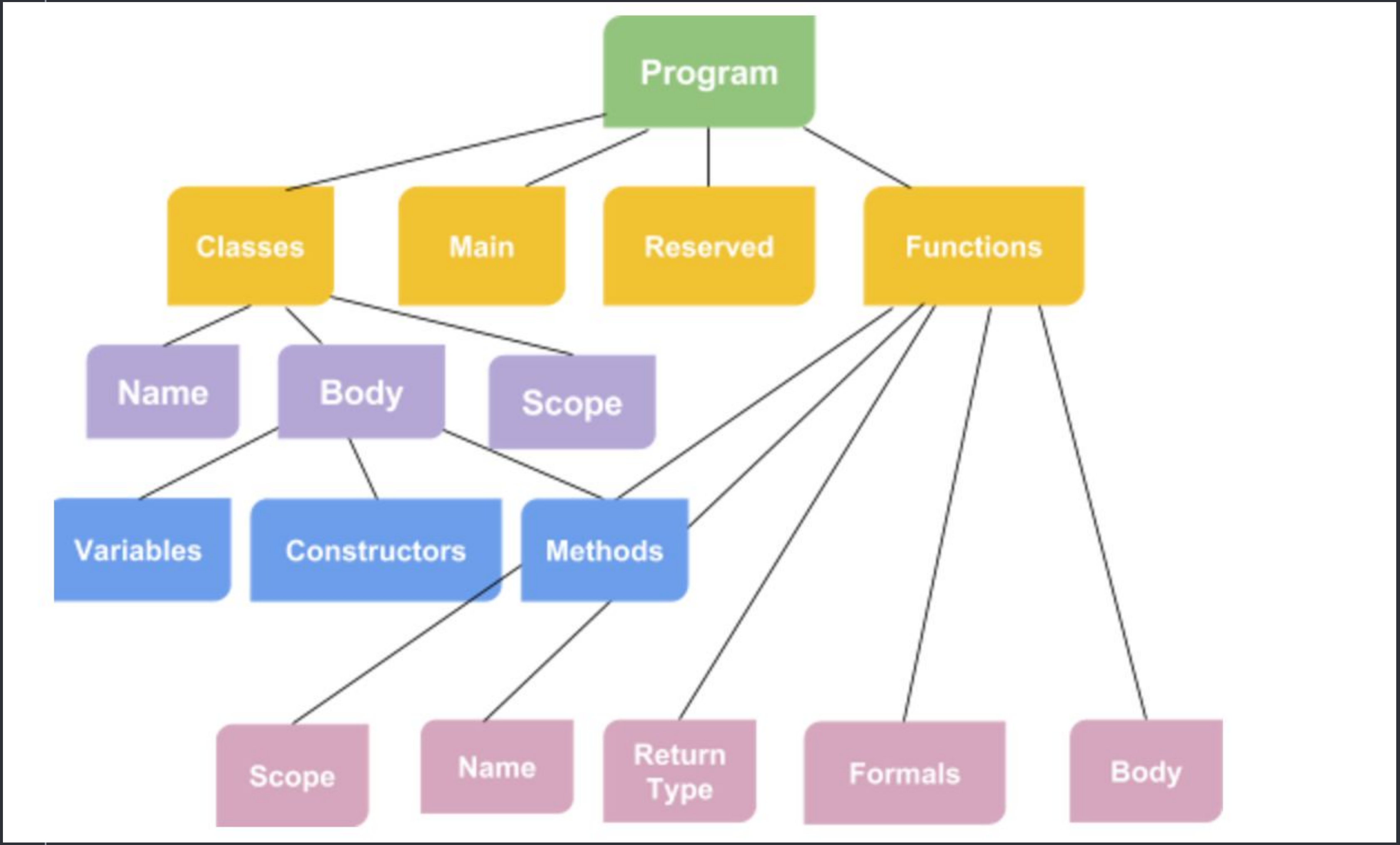
Architecture



AST


Program
Classes
Name
Body
Scope
Variables
Constructors
Methods
Statements
Expressions

SAST


Program
Classes
Main
Reserved
Functions
Name
Body
Scope
Variables
Constructors
Methods
Scope
Name
Return Type
Formals
Body

Testing

Testall.sh

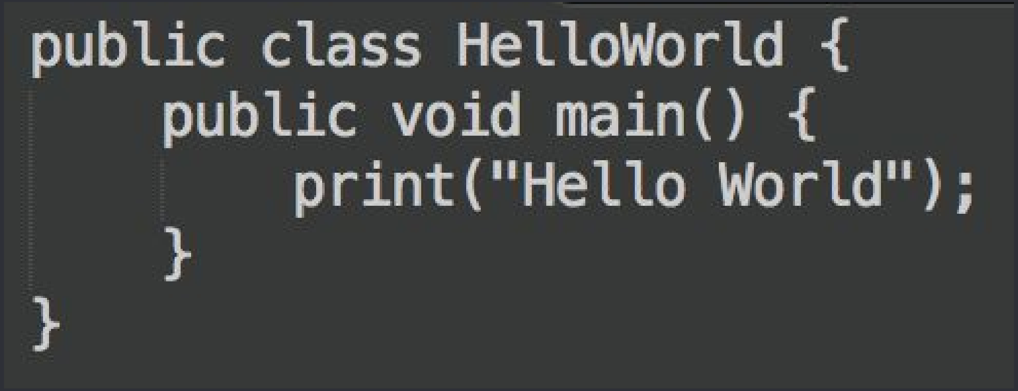
- Micro C’s test script
- Able to run all tests with one command: ./testall.sh

How?

- Complies and runs test.javapm and if result matches the corresponding .err or .out file, it passes.

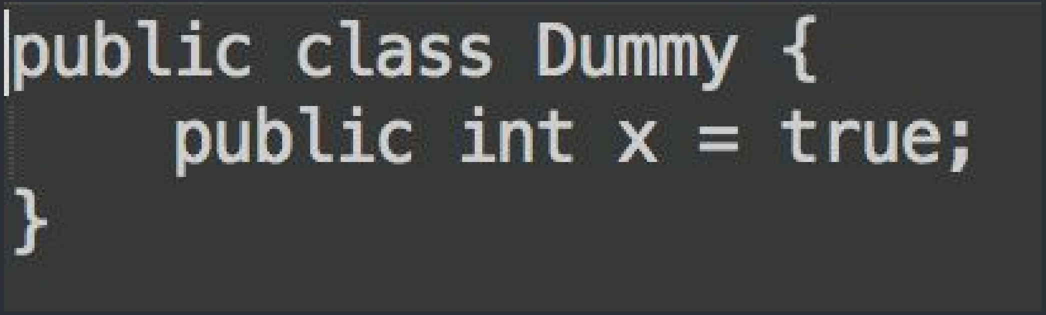
Testing

Pass Tests



```
public class HelloWorld {
    public void main() {
        print("Hello World");
    }
}
```

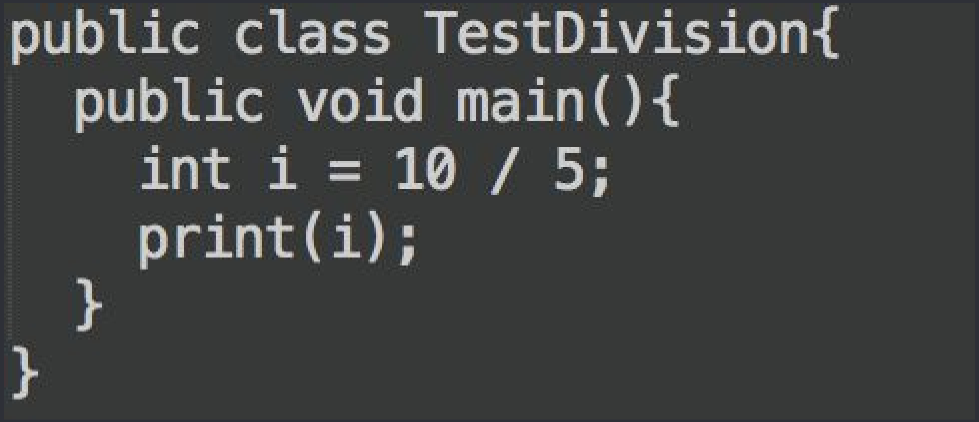
Fail Tests



```
public class Dummy {
    public int x = true;
}
```

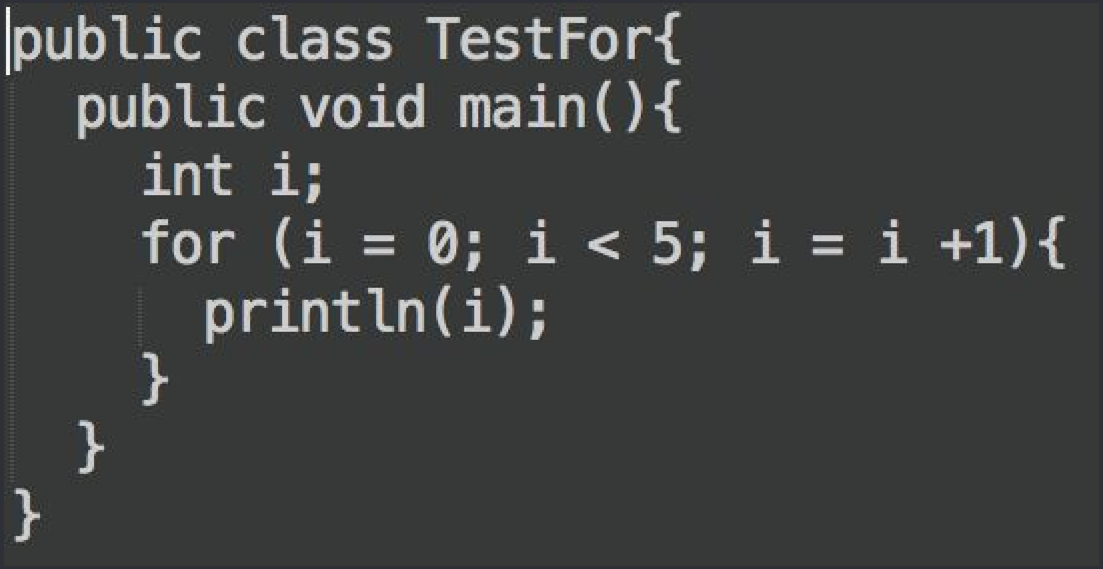
Testing

UnitTests



```
public class TestDivision{
  public void main(){
    int i = 10 / 5;
    print(i);
  }
}
```

Integration Tests



```
public class TestFor{
  public void main(){
    int i;
    for (i = 0; i < 5; i = i +1){
      println(i);
    }
  }
}
```

Tuple Creation

```
Tuple<int> myTuple = new Tuple<int>(5);
```

Tuple Access

```
myTuple<<0>>
```

Tuple Example

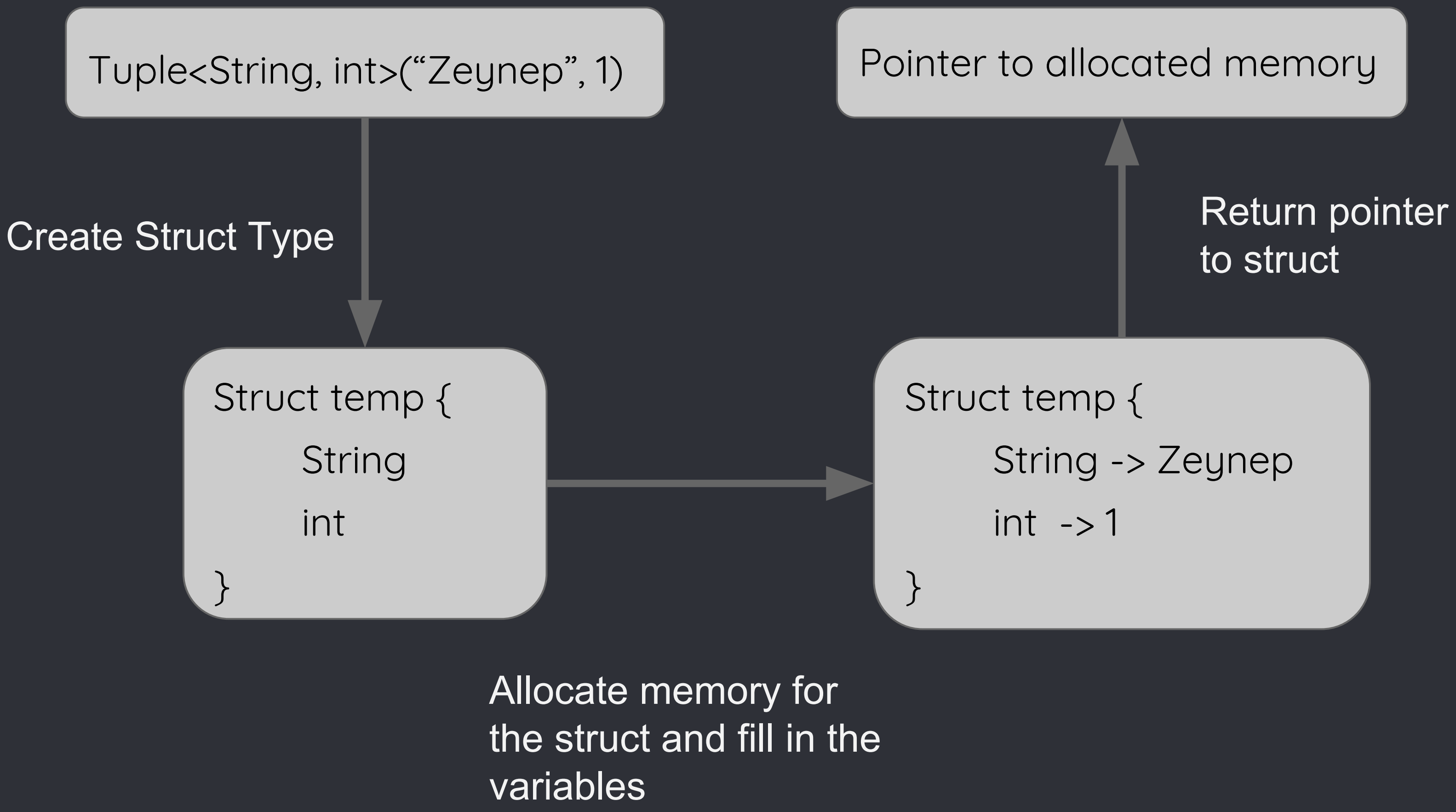
```
public class TestTupleAccess {
        public Tuple<String, Int>  getInfo() {
                int age = 5;
                Tuple<String, int> myTuple = new Tuple<String, int>(“Zeynep”, age);

                myTuple<<1>> = 22;

                return myTuple;
        }
        public  void main() {
                Tuple<String, int> info = getInfo();
                println(info<<0>>);
                println(info<<1>>);
        }
}
```

Tuples behind the scenes

Creation



Tuple<String, int>(“Zeynep”, 1)

Create Struct Type

```
Struct temp {
        String
        int
}
```

Allocate memory for the struct and fill in the variables

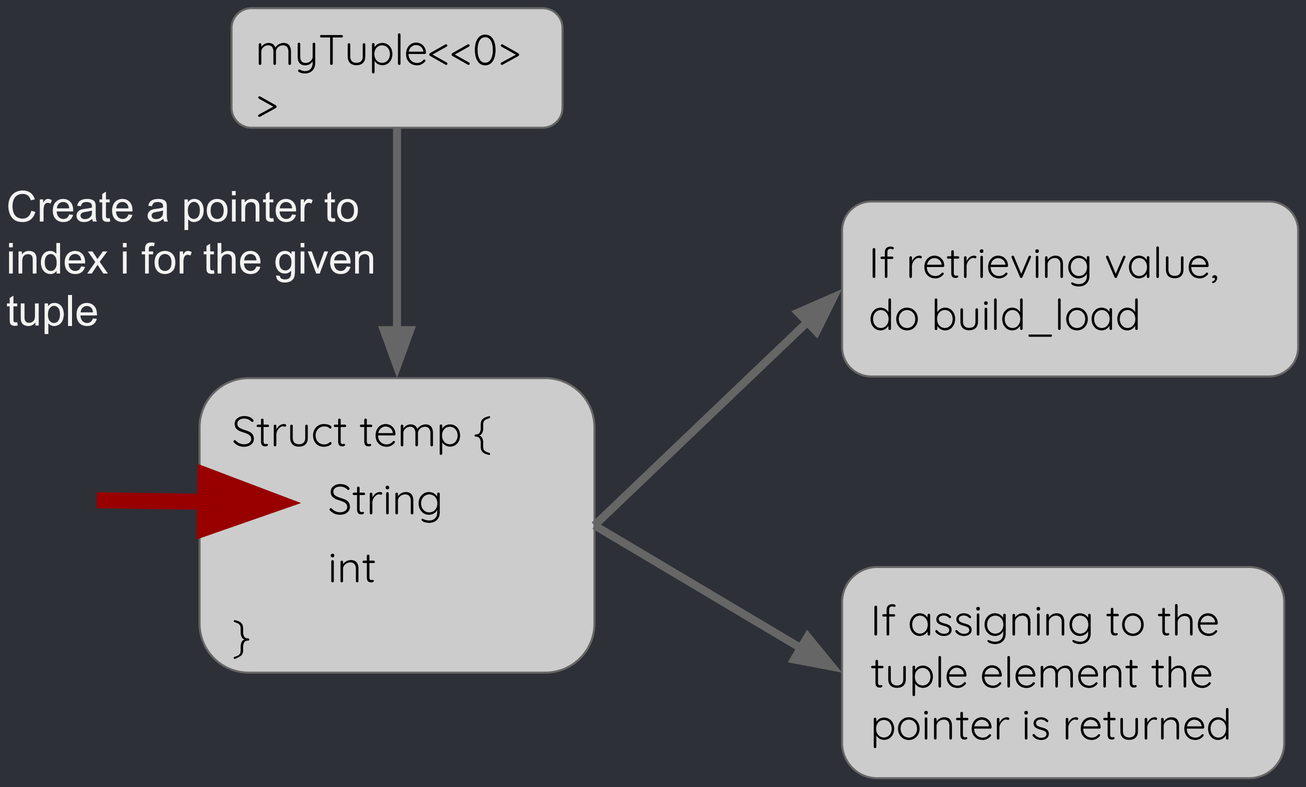
```
Struct temp {
        String -> Zeynep
        int  -> 1
}
```

Return pointer to struct

Pointer to allocated memory

Tuples behind the scenes

Access



myTuple<<0>
>

Create a pointer to index i for the given tuple

Struct temp {
String
int
}

If retrieving value, do build_load

If assigning to the tuple element the pointer is returned

Thanks!

And now a demo!