# NIKOLAOS ILIOPOULOS

# 1115201800332

## Second Exercise Compilers

### SymbolTable Structure

```
class SymbolTable {
    Map<String, ST_Class> classes; // holds all the symbol tables
    // key = class name , value = the class
}

class ST_Class {
    String name; // the name of the class
    ST_Class parent; // points to the parent else null
    ST_Class child; // points to the child else null

    Map<String, String> atributes;  // the attributes of the class
    // key = attribute name , value = attribute type
    Map<String, ST_Method> methods; // the methods of the class
    // key = method name , value = the method
}

class ST_Method {
    String name; // the name of the method
    String type; // the return type of the method

    Map<String, String> arguments; // the arguments of the class
    // key = argument name , value = argument type
    Map<String, String> bodyVariables; // the bodyVariables of the class
    // key = variable name , value = variable type
}
```

When a type checking error occurs i throw a custom error that it is just printing a message and then i continue to the next file.

```
class TypeCheckError extends Exception {
    public TypeCheckError(String message) {
        super(message);
    }
}
```

I have a SymbolTable inside the MyVisitor class so i can access the SymbolTables from all the visits.

### Second Argument for scope purposes

In visits i used the second argument to pass in the scope we are in.

> For example in varDeclaration():
>
> - If argu is "main->foo" it means that this variable declaration belongs to the **main class in the foo method** as a body variable.
> - But if the argu is "main" it means that this variable declaration is **an attibute of the main class**.

### Visits return

Bassically we care only about the types of all the things and not the thing it self.

> If we have:

```
int a;
a = 2; // int = int;
```

We care that in the assignment the right and the left parts are the same type. So i return for the 2->int.

If we have a messageSend as the left part i return a string **"return " + mytype**.

```
boolean a;
Test foo;
a = foo.getBool(); // boolean = boolean;

class Test{
    public boolean getBool(){
        return true;
    }
}
```

So the left part is **"return boolean"**. I dont care about the messageSend but it will return me a **boolean**. That is the only thing that i care.