NIKOLAOS ILIOPOULOS

1115201800332

Third Exercise Compilers

VTable Structure

```
class VTable {
    String name; // the name of the class
    Map<String, Integer> methods; // holds method -> offset of the method
    Map<String, String> belongsTo; // holds methods -> class of the method
    Map<String, Integer> variables; // holds variable -> offset of the variable
}
```

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.

MyVisitor atributes

I hold those variables in order to have information in the visits for the labels counters.

```
Integer register;
Integer exp_res_;
Integer if_else_;
Integer if_then_;
Integer if_end_;
Integer loop;
Integer oob_ok_;
Integer oob_err_;
Integer nsz_ok_;
Integer nsz_err_;
```

I hold those variables in order to have information about the previous messageSend return type in the other visits and the previous allocated object.

```
String LastClassAllocated;
String LastMessageSendType;
```

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**.