***Compilation course***

Asaf Bruner, 036762961, **asafbrun**

Aviv Goll, 036858777, avivgoll

***PA3 Docuementation***

1. **Code structure / Class hierarchy**
2. **Major classes description:**

PA2:

* Compiler class - Main program
* Lexer class – auto generated lexical analyzer.
* Parser class - auto generated syntax analyzer.
* LibraryParser - auto generated syntax analyzer for the Library only.
* LexicalError class – An exception class for the lexer's failed analysis.
* SyntaxError class – An exception class for the parser’s failed analysis.
* Sym class – auto generated Constants collection for the lexical analysis.
* Token class – The atom building block of the IC program.
* ValuedToken class – A Token with a meaningful value.

PA3:

Under IC.SemanticChecks package:

* BreakAndContCheck class – a visitor that checks that the break and continue statements appear only inside loops.
* SingleMainMethod – a visitor that checks that the program has only one main method in a correct structure.
* TypeCheck class – a visitor that checks all the type rules as defined in the IC specification.

Under IC.Types package:

* The classes: ArrayType, IntType, StringType, BoolType, ClassType, MethodType, NullType, VoidType inherits from the abstract class Type and represent a type for the TypeTable.
* TypeTable class – holds all the programs primitive and user defined types as singletons.

Under IC.SymbolTable package:

* Kind – an enum which holds the kind of the symbol table entry.
* Symbol class – an entry in the symbol table.
* SymbolTable class – each scope has a matching symbol table which holds all the relevant symbols and their types.
* SymbolTableConstructor class – a visitor which builds a symbol table for each scope.

1. **Testing strategy**

Our testing strategy tried to simulate all the normal and extreme cases. In addition we made sure to have test cases creating all possible errors to see if they are shown.

We also made sure that the sample tests will produce the same output as the supplied output – **excluding the order in which the Symbol Tables are printed and the type numbering.**

We did not include in the submission tests for the parts implemented in previous exercises.

1. **Semantic analysis implementation strategy:**

After we have built the AST for the program we use the SymbolTableConstructor visitor which traverse the AST and builds symbol table for each scope in the program, and a type table for the program. During this process we handle scope rule violations.

Now we run the SingleMainMethod and the BreakAndContCheck visitors which ensures that there is only one properly structured main method and that the 'break' and 'continue' statements are inly inside loops.

Finally we run the TypeCheck visitor which handles type rule violations.

1. **Bonus:** we have implemented that a method with a non-void return type returns a value on every control path.