***Compilation course***

Asaf Bruner, 036762961, **asafbrun**

Aviv Goll, 036858777, avivgoll

***PA4 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.

PA4:

Under IC.Types package:

* DVCreator – traverses the symbol tables and builds the appropriate dispatch vector.
* StringMapper – a visitor that builds the string headlines for LIR.
* Translator – the main visitor method which translates to LIR.

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. **LIR implementation strategy:**

We have implemented the LIR translation in three phases:

* 1. Extracting and printing the string literals to LIR string literals.
  2. Building the dispatch vector and field offset data structure and printing it.
  3. Finally doing the visitor which traverses our AST and using the symbol tables to translate to LIR.

1. **Bonus:** we have implemented the Reusing registers bonus assignment.
2. **Feedback:** It has been a pleasure – cheers.