Introduction to Language Theory and Compilation Exercises

Session 7: Semantic Analysis

Reminders

The stages of compiling are: Preprocessing, Lexical analysis, Parsing (syntactical analysis), **Semantic analysis**, Code generation, Code optimization and Linker.

The role of of the `Semantic analysis` is to check if lexical units make sense in their context.

- Resolve names: association between the labels (lexical unit) and the identifiers (from the table of symbols).
- Construct the table of symbols: add all required information (identifier, type, ...) into the table of symbols.
- Check type of expressions: conversion between integer/real, strings, ...
- Check if the constants are well defined (non-null).

Exercises

Ex. 1. Assuming that the Java grammar contains rules of Figure 1, explain why these methods are (not) semantically correct.

```
1  public class Exercise1{
2      public static void exerciseA(String[] args){
3          if(!args) throw new IllegalArgumentException();
4          System.out.println("All right!");
5      }
6      public static void exerciseB(String[] args){
7          if(new Boolean(true)) throw new IllegalArgumentException();
8          System.out.println("All right!");
9      }
10 }
```

<IF>	→	if <EXPRESSION> <END_IF>
<EXPRESSION>	→	<BOOLEAN_EXPRESSION>
<EXPRESSION>	→	<NUMERIC_EXPRESSION>
<EXPRESSION>	→	<CHAR_EXPRESSION>
<EXPRESSION>	→	<OBJECT_EXPRESSION>

Figure 1: Some speculative rules about conditions of the Java grammar

Ex. 2. Assuming that the Java grammar contains rules of Figure 2, write the expression tree decorated with the type of each node. Do not forgot to respect the arithmetical priority of operators.

```
1  public class Exercise2{
2      public static void main(String[] args){
3      System.out.println(3+4+"."+(1+2*3.0));
4      }
5  }
```

<EXP>	→	<EXP> + <EXP>
<EXP>	→	<EXP> * <EXP>
<EXP>	→	<EXP> - <EXP>
<EXP>	→	<EXP> / <EXP>
<EXP>	→	*(int)* <EXP>
<EXP>	→	(<EXP>)
<EXP>	→	ID
<EXP>	→	Bool
<EXP>	→	Real
<EXP>	→	Int
<EXP>	→	String

Figure 2: Some speculative rules about expressions of the Java grammar

Ex. 3. Assuming that the Java grammar contains rules of Figure 2, identify the scope of all variables

- Give the table of symbols (ToS)
- Give the parse tree of each numerical expression
- Annotated the parse trees with changes of the table of symbols

Report any semantic error.

```java
public class Exercise3{
   public static final double PI = 3.141592653589793;
   public static double diameter;
   public static void main(String[] args){
      double perimeter = 50;
      diameter = perimeter / PI;
      final int diameter = Exercise3.diameter;  //15.915494309189533
      System.out.println(((int)(diameter*100))/100.0);
   }
}
```