CS 352 – Spring 2011
Project 3b

Handed out: **Mar 29th, 2011**
Due: **Apr 24th, 2011 @ 11:59pm**

Abstract:
In this project, you will create a visitor that performs type-checking. You will be traversing the Abstract Syntax Tree (which was generated by the parser in Project 3a), and checking for various type compatibilities. The grammar for this project has been modified.

Setting up required tools:
Your setup for this project will be similar to that in project 3a.

Setting up your code skeleton:
The code skeleton consists of **project3b.zip** file.
(You are required to use this skeleton code as a starting point.)
You are provided with class files for the parser (as implemented in 3a), syntaxtree, semant and visitor packages.

Changes in Grammar:
The Grammar no longer has a main class, class inheritance (extends), arrays, not operator and certain binary operators, the new BNF is attached to this file.

What you need to do:
You only need to modify semant/Semant.java.
You are expected to catch type errors like:

- Using a variable that hasn't been defined.
- Using undeclared member function in a class.
- Redefining a class.

etc…
Visualize what kinds of type errors are possible, create test cases to test those and check with solution key. Scope should be enforced like regular Java.

The output of your code:
Your compiler must produce the same (standard and error) output as the solution compiler, including generating the same number and formatting of error messages.
Use diff to compare the outputs, use the following commands:
```
javac */*.java
java Parser/MiniJavaParser test.java >& out1 (your program)
p3b test.java >& out2 (solution program)
diff -w out1 out2
```
The last command should not produce any output

Turnin:
To submit: turnin –c cs352 –p p3b semant/Semant.java
To verify: turnin –v –c cs352

Any doubts concerning the project requirements should be raised by sending email to lib@purdue.edu. The TA may then discuss things with the instructor if necessary.

BNF for MiniJava

NON-TERMINALS

```
             Goal ::= ( ClassDeclaration )+ <EOF>
 ClassDeclaration ::= class Identifier "{" ( VarDeclaration | MethodDeclaration )* "}"
   VarDeclaration ::= Type Identifier ("," Identifier)*  ";"
MethodDeclaration ::= public Type Identifier "(" ( Type Identifier ( "," Type Identifier )* )? ")" "{" (
                      VarDeclaration | Statement )* return Expression ";" "}"
             Type ::= int
                    | boolean
                    | Identifier
        Statement ::= "{" ( Statement )* "}"
                    | if "(" Expression ")" Statement ( else Statement )?
                    | while "(" Expression ")" Statement
                    | System.out.println "(" Expression ")" ";"
                    | Identifier "=" Expression ";"
       Expression ::= Expression ("&&"  |  "<"  |  "+"  |  "-"  ) Expression
                    | Expression "." Identifier "(" ( Expression ( "," Expression )* )? ")"
                    | <INTEGER_LITERAL>
                    | true
                    | false
                    | Identifier
                    | this
                    | new Identifier "(" ")"
                    | "(" Expression ")"
       Identifier ::= <IDENTIFIER>
```