Computer Science
Engineering School

Software
Engineering

Lab 08
Identification Phase

Francisco Ortín Soler

University of Oviedo

Objective

- Implement the **identification phase** of your compiler
  - `IdentificationVisitor` class

    in the `semantic` package

Recall

- **Identification phase** is the first traversal in semantic analysis
- Its purpose is to link all the `Variable`s (including function names in function invocations) to their `Definition`s

Question

- Identify the errors (if any) in the following program (input-wrong.txt)

```
 int integer;
 char character;
 double real, integer;

 void p(int a) {
      char a;
 }
void main() {
      double character;
      read integer, i;
      f();
}
```

Questions

- Given the following program

```
01:  int a;
02:  double f(double b) {
03:       return a+b;
04:  }
05:  void main() {
06:       write a, f(3.8);
07:  }
```

- For each variable occurrence in the program (`Variable` node in the AST):
    1. Identify its location in the program
    2. Indicate the `Definition` node it must be bound to

3. What is the name of the new field to implement such link?
4. To which AST nodes should we add that field?
5. What is the type of that new field?

Questions

1. What is the name of the data structure to be used?
2. What are the messages (public methods) to be provided by that data structure?
3. Trace, for the following code, the messages to be passed to that data structure while traversing the AST

```
int a;
double f(double b) {
        return a+b;
}
void main() {
        write a, f(3.8);
}
```

Activity 1: Implement Symbol Table

- Finish the implementation of `SymbolTable.java`

```
public class SymbolTable {
      private int scope=0;
      private List<Map<String,Definition>> table;
      public SymbolTable()  { /* ... */ }
      public void set() { /* ... */ }
      public void reset() { /* ... */ }
      public boolean insert(Definition definition) {
             /* ... */ }
      public Definition find(String id) { /* ... */ }
      public Definition findInCurrentScope(String id) {
             /* ... */ }
}
```

Activity 1: Implement Symbol Table

- Test the implementation of `SymbolTable.java` by running `SymbolTableTest.java`
  - Enable asserts! (i.e., pass `–ea` to the Java virtual machine when running `SymbolTableTest`)
- First, make it fail on purpose to make sure asserts are being checked

Activity 2: Implement Symbol Table

- Implement the `IdentificationVisitor` to link all `Variable` nodes to their `Definitions`
  - Use your `SymbolTable` class as a private field
- When done, check
  1. That your compiler shows the expected errors for `input-wrong.txt`
  2. Using Introspector, that all the variables in `input.txt` are correctly bound to their definitions
     - Global, local variables and parameters
     - Variables and functions
     - Check that their **`scope`** field has the correct value