Computer Science
Engineering School

Software
Engineering

Lab 09
Type-Checking Phase

Francisco Ortín Soler

University of Oviedo

Objective

- Design (first) and implement (second) the **type-checking phase** of your compiler

Purpose

1. The **type-checking phase** of your compiler must be defined with an **Attribute Grammar**
2. Then, the AG must be implemented as the `TypeCheckingVisitor` class in the `semantic` package
   - Already created in lab 07

- Purpose
  1. Detect any error in the input program, not detected by the previous phases
  2. **Annotate** all the `Expression` nodes in the AST with their **types**

Question

- Identify the errors (if any) in the following program (input1-wrong.txt)

```
01: int integer;
02: char character;
03: double real;
04:
05: void main() {
06:     read integer;
07:     character=8.5;
08:     3=integer;
09:     read integer+2;
10:     integer = character + 'a';
11: }
```

Question

- Identify the errors (if any) in the following program (input2-wrong.txt)

```
01: struct  {
02:       int day;
03:       int month;
04:       double day;
05: } date;
06:
07: void main() {
08:       read date.year;
09: }
```

Question

- Identify the errors (if any) in the following program (input3-wrong.txt)

```
01: int i;
02: double f;
03:
04: struct  {
05:     int a;
06:} a;
07:
08: void main() {
09:     write i && f;
10:     write a >= 3;
11:     while (f)
12:         if (f)
13:             write !f;
14: }
```

Question

- Identify the errors (if any) in the following program (input4-wrong.txt)

```
01: int i;
02: int f(int a) {
03:    double a;
04:    return 34.5;
05: }
06: void g(int b, double b) {
07:    return 3;
08: }
09: void g() {
10: }
11: void main() {
12:     f();
13:     f(3.2);
14:     g(3.2, 3);
15:     i=g(1, 2.3);
16: }
```

Question

- Identify the errors (if any) in the following program (input5-wrong.txt)

```
01: int[10] v;
02:
03: void main() {
04:     int i;
05:     i[0]=0;
06:     v[3]=3.4;
07:     v[1][2]='a';
08: }
```

Activity 1: Design of Type System

- Define an AG that, using your type system, infers the type of any expression
- Let's write some semantic rules of the AG
- How do we infer the type for arithmetic operations?

Arithmetic: $\text{expression}_1 \rightarrow \text{expression}_2\ \text{expression}_3$

- Write the AG rule to infer and type-check arithmetic expressions

R: ?

Activity 1: Design of Type System

1. How do we infer the type for indexing operations

   Indexing: $\text{expression}_1 \rightarrow \text{expression}_2 \;\; \text{expression}_3$

   R: ?

2. How do we infer the type for function invocations

   P: ?

   R: ?

3. Do we need to perform any type checking in while statements?

   WhileStmt: $\text{statement}_1 \rightarrow \text{expression statement}_2$*

   R: ?

Activity 1: Design of Type System

- Finish all rules in the AG to type-check the whole language
  - Read the supported-operations.pdf file
  - Ask the lecturer if you have any doubts
  - Write the AG as comments in TypeCheckingVisitor.java
- Show the AG to the lecturer (**mandatory**) before starting Activity 2

Activity 2: Implementation

- Once the AG is validated by the lecturer, implement it by extending your `TypeCheckingVisitor` (lab07)

- Check that your compiler shows the appropriate error messages for the all the input-wrong.txt files provided
- Use Introspector to check that your compiler infers types correctly for valid input files (input.txt)