Computer Science
Engineering School

Software
Engineering

Lab 05
AST Construction with ANTLR

Francisco Ortín Soler

University of Oviedo

Objective

- Extend your language recognizer (parser) to create an AST representing the input program

Syntax Analyzer

- Merge in one project
  - your AST implementation (lab02)
  - your language recognizer (lab04)
- Objective
  - Add embedded actions to your grammar to create the AST
  - Visualize the AST using Instrospector

Introspector

1. Add the `introspector.jar` file to your project
2. Include it as a library
3. Modify your main method with the following code to show your AST with Introspector

```
Program ast = parser.program().ast;

IntrospectorModel model=new IntrospectorModel("Program", ast);
new IntrospectorView("Introspector", model);
```

4. **Implement all the `toString` methods** in your AST nodes (all must be simple implementations)

Autonomous work

1. Add embedded actions to your grammar to create the AST
   - **Step by step process**:
     1. Add one embedded action to one production
     2. Generate the code with ANTLR
     3. Check whether CmmParser.java has any errors
        a) If so, locate the origin of the error in Cmm.g4
        b) Think for a solution
        c) Modify the embedded action in Cmm.g4
        d) Go back to step 2
     4. Repeat the process for a new production (step 1)
2. Visualize the AST using Instrospector
3. Test the AST created is the expected one, using the input.txt file from lab04, among others

Questions

- Given the following grammar, extend it to create the AST

```
grammar Cmm;

@header {
}

expression:
      ID
    | INT_CONSTANT
    | expression ('+'|'-') expression
    | ...
    ;
```

Question

- What is the AST you must create for the following type `int[20][10]`?