CSCI U530 Assignment 02: Part 02

# The Task

You will modify the recursive-descent parser from Part 01 to build an Abstract Syntax Tree (AST). To create the AST, you will use a series of classes to represent each grammar rule. Object-Oriented programming will really play to your favor in this project. When a grammar rule has multiple choices, you can create a base class (sometimes just an abstract class) and have each option in the grammar as a separate class that inherits from your base class. That way when you are parsing, you can create an object of the appropriate class as you parse through the syntax. **We will build the class hierarchy together in class**.

In order to build the AST, you will create all the classes we have identified in the class hierarchy, as well as modify Parser.java to create the appropriate nodes for the syntax tree. (Hint: each recursive descent method in the parser can be modified to return an object of the appropriate type to help build the tree). Once the Syntax tree is built, you will traverse the tree and use the knowledge you gain from each tree node to translate the statements into matching Java statements. Each class must have a convertToJava method which will return a String of the appropriate Java statement for that class. In addition to printing the corresponding Java statements, you must also keep proper formatting and indentation. You will do this through the use of a tabLevel property every class must have. In order to ensure that all of the classes are forced to have a convertToJava method and a tabLevel property, all of the class will inherit from an abstract class called ASTNode.

# Sample Output

Below you will find the expected output for each of the three input files provided on Blackboard. Your output must match this exactly. Any programs which do not provide the expected output will receive an automatic deduction of 50%. **Programs which do not compile will receive a zero.**

|  |  |
| --- | --- |
| in1.simplelang | Expected Output |
| declare x as integer = 2 + 5  loop  print x  let x = x + 1  until x > 10 | int x = 2 + 5;  do  {  System.out.println(x);  x = x + 1;  }  while (x > 10); |

|  |  |
| --- | --- |
| in2.simplelang | Expected Output |
| declare mpg as integer  declare miles as integer = 400  declare gallons as integer = 20  let mpg = miles / gallons  print mpg | int mpg;  int miles = 400;  int gallons = 20;  mpg = miles / gallons;  System.out.println(mpg); |
| in3.simplelang | Expected Output |
| declare i as integer = 1  declare num as integer = 100  loop  if num % i == 0 then  print i  endif  let i = i + 1  until i >= num | int i = 1;  int num = 100;  do  {  if (num % i == 0)  {  System.out.println(i);  }  i = i + 1;  }  while (i >= num); |

# Submission

When finished, you will submit a zipped folder of your entire Eclipse project to Blackboard. Make sure you named the project **YourLastNameAssign02**. Programs named incorrectly, turned in incorrectly, and programs that do not compile **will receive a zero**. Programs that do not provide the appropriate output will receive a 50% deduction in grade. **This is an individual assignment**. Any discussions on the assignment should be very high level and not discuss particular implementations.