Assignment 2: Expression Parser

Academic integrity is taken seriously in this course, which has a zero tolerance policy for cheating. All discovered incidents of cheating will be reported and result in an E in the course.

You are allowed to:

- discuss high level solutions with other students
- ask questions in the class slack channel and during office hours
- use code presented in the lectures
- share code for Junit tests

You are NOT allowed to

- Share your code with anyone else (except your official project partner)
- Receive code from others (except your official project partner), including students in previous semesters.
- Use the Scanner code from another student, even after it has been graded.

All work submitted should be the work of the submitting student(s), created this semester for this assignment.

Specification

Your task is to write a parser for a subset of the language we will be implementing this semester. We will extend the parser to cover the rest of the language in assignment 3. The subset covers most of the expressions in our language.

The parser should return an AST if the given input is a legal expression and throw a SyntaxException if the sentence is not legal according to the given context-free grammar. Grammars for both the concrete and abstract syntax are found in the Syntax document. If an error is discovered by the Scanner, a LexicalException should be thrown, so lexical errors are handled the same way as before.

Add the following method to your CompilerComponentFactory class

```
public static IParser makeAssignment2Parser(String input)
      throws LexicalException {
        //add code to create a scanner and parser and return the parser.
}
```

Provided code

IParser.java Interface for the parser. Your parser must implement this interface.

AST.java and subclasses for the abstract syntax tree nodes in package edu.ufl.cise.plcsp23.ast. You may want to change the toString methods in these classes, but otherwise they should not be modified. This is a subpackage of edu.ufl.cise.plcsp23 and the code in the file system should be places in a directory called ast inside edu/ufl/cise/plcsp23.

SyntaxException.java The Exception to throw for errors discovered during parsing

Assignment2Test_starter.java Example JUnit tests. This class will be expanded with additional tests, so it is essential that your code runs and passes these tests. The provided tests do not provide complete coverage, so you will need to create your own to adequately test your parser.

Github

See Assignment 1. Evaluate your workflow from the previous assignment and improve as necessary. Make sure to backup frequently so you won't lose your work if your laptop breaks or some other disaster occurs.

To Turn in

- **A jar file containing the source code** (i.e. java files) of all classes necessary to test your parser with the appropriate directory structure. Do NOT include class files. Be aware that your IDE may provide a way to generate jar files that defaults to including .class files instead of .java files. Double check the contents of your jar file before submission.
- **A zip file containing your git repository WITH history**. Do NOT use the "download zip file" option on github as it does not include the history. See Assignment 1, and the FAQ for more information.

Hints

- The given grammar is not necessarily LL(1). Check this before you start and tweak the grammar if necessary.
- The background necessary for this assignment was discussed in class during several lectures. It will save you a lot of time to understand this material before jumping in to write code.