Syntax Analyzer

In this project, you will build a small compiler for MiniJava language. In the second phase of this project, you are required to build a program that parse a stream of tokens and output a parse tree.

You should follow these steps to finish this phase:

1- Read tokens from the file.
2- Remove ambiguity in MiniJava grammar. **(P2 part1)**
3- Prepare the types of each node that can be in the tree. **(P2 part2)**
   each node should include:
   - Its own suitable member variables.
   - Constructors to initialize these member variables.
   - A function that pretty prints the class and its content.
4- Test your nodes by constructing a tree by hand. (Like lab 4)
5- Create a Parser class that implements MiniJava grammar using Recursive decent parser. **(P2 part3)**
   This class should has:
   - A stack/Queue -or any suitable containers as you see fits- that holds tokens constructed by Lexical analyzer.
   - A node root, which is the root of the constructed parse tree.
   - A constructor.
   - A parse function that calls the start rule in the grammar.
   - Each grammar rule almost has a function. This function should return its suitable node data type from parse tree in P2 part2.
6- Test your parsing functions by printing the content of each node in the tree.

Notes:

- **The input** of this program is a stream of tokens read from a file.
- **The output** is a parse tree tested by a pretty print function.
- If your **phase 1** is not implemented correct,
  - Only fix class Token as you may need it as a data type while you go throw this phase.
  - Write your own test cases by writing directly in the file by hand with the same format used in phase 1.

- **Grammar** + **Code sample** are in a separate document named **MiniJava.pdf**.
- **Submission should include:**
  - Edited Token class
  - Parsing code
  - Grammar after removing its ambiguity in a word/ pdf document.
- Your submission will be on **Acadox** as zip or rar file.
- **Deadline:** 21/4/2018 at 11:55 PM
- **Name of the submitted file should follow**
  P2_Lab#_ID1_ID2_ID3_ID4.rar
  Lab1 if you are from CS_IS_3 or CS_IS_4
  Lab2 => CS_IS_1 or CS_IS_2
  Lab3 => CS_IT or CS_DS
  For example:
  P2_lab2_20140001_20140002_20140003.rar