## Minimal Submitted Files

You are required, but not limited, to turn in the following source files:

Assignment11.java (You do not need to modify this file.)
Matching.java -- to be created.

## Requirements to get full credits in Documentation

- The assignment number, your name, StudentID, Lecture number, and a description of each class/file need to be included at the top of each file/class (in this assignment, you might have only one file/class, but in future there will be more than one file.)
- A description of each method is also needed.
- Some additional comments inside of methods (especially for a "main" method) to explain code that are hard to follow should be written. You can look at Java programs in the text book to see how comments are added to programs.

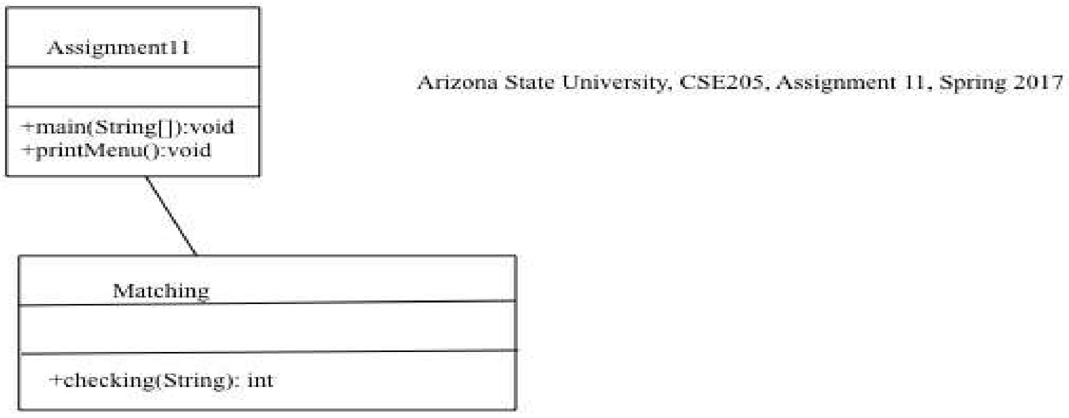
## New Skills to be Applied

In addition to what has been covered in previous assignments, the use of the following items, discussed in class, will probably be needed:

Stacks

## Program Description

**Class Diagram:**



Assignment #11 will be the construction of a program that takes an input string, and check if parentheses and braces in each string are matching or not.

The method checking() in the Matching class needs to check if parentheses and braces in the parameter string is matching or not using an object of the Stack class in java.util package.

## Instruction:

## Assignment11 class

This class displays a menu for the parentheses or braces checking. If a user enters "P", then it asks to enter a string. This class is given by the instructor.

### Matching class

You need to complete the following method.

**public static int checking(String inputString)**

You need to write the checking method that takes an input string, and read every character from its left to right, then returns 0, 1, 2, 3, or 4 based on the following description.
It should create an empty Stack object at the beginning, and if a character is a left parenthesis '(' or a left brace '{', then push it (or its corresponding string such as "(" or "{") onto the stack.
If a character is a right parenthesis ')' and the top of the stack is '(', then pop it and continue to process the next character of the input string. But if the top of the stack is not '(' or the stack is empty, then that means that there was no matching left parenthesis, thus the method should return 1.
If a character is a right brace '}' and the top of the stack is '{', then pop it and continue to process the next character of the input string. But if the top of the stack is not '{' or the stack is empty, then that means that there was no matching left parenthesis, thus the method should return 3.
After reading all characters of the input string, if the stack is empty, then it means that everything is matching, the method should return 0.
After reading all characters of the input string, if the stack is not empty, and the top of the stack is a left parenthesis '(', then there was no matching right parenthesis, thus the method should return 2.
After reading all characters of the input string, if the stack is not empty, and the top of the stack is a left brace '{', then there was no matching right brace, thus the method should return 4.

The error messages are based on the first parenthesis or brace from left of the input string that does not have any matching.

***Requirements:***

You need to implement this method using an object of the Stack class) in java.util package.

### Input

The following files are the test cases that will be used as input for your program (Right-click and use "Save As"):

Test Case #1
Test Case #2
Test Case #3
Test Case #4

### Output

The following files are the expected outputs of the corresponding input files from the previous section (Right-click and use "Save As"):

Test Case #1
Test Case #2
Test Case #3
Test Case #4

### Error Handling

Your program is expected to be robust enough to pass all test cases. You may use the Scanner class, but are not limited to it, to handle your user input.