CSC 360 Assignment # 6
Total Points: 40

Problem 1: Using a stack to check order of parenthesis (20 points)

In this program, you will read your inputs from a file *input.txt*, which will contain multiple lines with different sequences of brackets. You task is to create a class ParenthesisMatch.java, with two method s: isParenthesisMatch and main. The main method will read the input.txt file, which will contain the following example inputs.

```
(())
(())()()
()
()()
(()(
)())
(()
```

The main method will read each line from the file, and pass the string to the isParenthesisMatch method. The isParenthesis method will check if the order of the parenthesis is valid, and will return a Boolean true/false. E.g. for the above lines, the outputs will be :

```
true
true
true
true
false
false
false
```

The above problem can be easily solved using a stack. The algorithm is that, as you scan each character in the string of parenthesis, every time you see a '(', then you push it in the stack. Otherwise, if you see a ')', you check if the current top of stack is '('. If it is, then you pop it out of the stack. Otherwise, if the stack is empty, or if the current top is not '(' , you return false. Finally, when all the characters in the string is scanned, and if the stack is empty, you return true. Otherwise, if the scanning of the string is competed, but the stack is not empty, you return false. Use the pseudocode below to solve the problem as described.

```
For each character C in string of parenthesis
        If C is '(', then push C in stack
        Else if C is ')', then
            If stack is empty, then return false
            Else if top of stack is '(', then pop element from stack
            Else return false

Return true if stack is empty, or false otherwise
```

Problem 2: Using a map to count (20 points)

Review the `CountOccurrenceOfWords` program in Section 21.6. Your task is to write a class CountWithMap.java and corresponding methods that reads in integers from an input file called integers.txt and reports the integer that has the most occurrences. Use a `java.util.Map<Integer, Integer>` to keep track of how many times each integer has appeared. The input ends at the end of the file. If not one, but several numbers have the most occurrences then all of them should be reported. For each such number, display its value and its number of occurrences. Shown below is a sample Integers.txt input file and the corresponding output.

```
56
-5
10000007
-52
56
2
19
19
19
Thirteen
-2

-2
Nine hundred and ninety-seven
3
40
3
5
4
-3
3
3
2
-2
-2
0
19
```

Output:

```
Warning - unable to parse string as integer: Thirteen
Warning - unable to parse string as integer:
Warning - unable to parse string as integer:  Nine hundred and ninety-seven
Most frequently occurring values:
Value:  19  Number of occurrences:   4
Value:   3  Number of occurrences:   4
Value:  -2  Number of occurrences:   4

Process completed.
```

You will, of course, read the information from the file using a Scanner, as in Section 12.11.3. Use a `while (input.hasNextLine())` loop. Read in each line using `input.nextLine()`. Once, you have read in a line, parse it as an integer by calling the `Integer` constructor that takes a string:

```
integer = new Integer(string);
```

Your program should use try/catch blocks to catch exceptions under two circumstances: First, if the input file Integers.txt cannot be opened from the current working directory then the program should display an "Unable to open file" message and terminate. Secondly, if any of the lines in the input file consist of data that cannot be parsed by the `Integer(String s)` constructor then a warning message should be displayed (see example output on previous page) but the program should continue. The `Integer(String s)` constructor throws a `NumberFormatException` when it cannot parse the string.

It is conceivable that an `IOException` will be thrown while your program is running. You do not have to try to catch this checked exception unless you want to do so. Instead, you can simply declare that your main method `throws IOException`.

Note: The following is a really convenient way to get the maximum value in a map:

```
Integer maxCount = Collections.max(map.values());
```

What to turn in:
Submit your two programs on Blackboard. Include some comments at the beginning of each file including your name, the course number (CSC 360), and a brief description of what the program does. Name the files as **ParenthesisUsername.java** and **MapCountUsername.java**, substituting Username with your NKU username.