Assignment 7

Instructions

Max score is 10.
Deadline is 24: 00 March 18,Sunday.
Extra credits are added only if total score is less than 10.

You should write JUnit Test for question 3,4 and 5.

1. When a program fails due to an uncaught exception, the system automatically prints out the exception's stack trace. If the failure is not easily reproducible, it may be difficult or impossible to get any more information. Therefore, it is critically important that the exception's toString() method return, as much information as possible concerning the cause of the failure. In other words, the detail message of an exception should capture the failure for subsequent analysis. To capture the failure, the detail message of an exception should contain the values of all parameters and fields that "contributed to the exception." (Score 2).

- - Create your own **MyIndexOutOfBoundException** Class. It should contain these parameters

    - **lowerBound** - the lowest legal index value.
    - **upperBound** - the highest legal index value.
    - **index** - the current index value.
- Test your code in main method, by creating an indexOutOfBoundException. Output error message should be like this:

```
 "Error Message: Index: 10, but Lower bound: 0, Upper bound: 9"
```

2. Modify the following parse() method so that it will compile: (Score 1)

```
  public static void parse(File file) {
      RandomAccessFile input = null;
      String line = null;

      try {
          input = new RandomAccessFile(file, "r");
          while ((line = input.readLine()) != null) {
              System.out.println(line);
          }
          return;
      } finally {
            if (input != null) {
              input.close();
            }
        }
  }
```

3. Implement a class called Tool. It should have an int field called strength and a char field called type. You may make them either private or protected. The Tool class should also contain the function void setStrength(int), which sets the strength for the Tool.

Create 3 more classes called Rock, Paper, and Scissors, which inherit from Tool. Each of these classes will need a constructor which will take in an int that is used to initialise the strength field. The constructor should also initialise the type field using ‘r’ for Rock, ‘p’ for Paper, and ’s' for Scissors.

These classes will also need a public function bool fight(Tool) that compares their strengths in the following way:

Rock's strength is doubled (temporarily) when fighting scissors, but halved (temporarily) when fighting paper. In the same way, paper has the advantage against rock, and scissors against paper. The strength field shouldn't change in the function, which returns true if the original class wins in strength and false otherwise. You may also include any extra auxiliary functions and/or fields in any of these classes. Run the program without changing the main function, and verify that the results are correct. (Score 3)

```
class Tool{
// add your code here


}

/* Implement class Scissors */

/* Implement class Paper */

/* Implement class Rock */


class RockPaperScissorsGame{

        public static void main(String args[]){

            Scissors s = new Scissors(5);
            Paper p = new Paper(7);
            Rock r = new Rock(15);

            System.out.println(s.fight(p) + " , "+
p.fight(s) );
            System.out.println(p.fight(r) + " , "+
r.fight(p) );
            System.out.println(r.fight(s) + " , "+
s.fight(r) );

}

}
```

4.Given a set of time intervals in any order, merge all overlapping intervals into one and output the result which should have only mutually exclusive intervals. Let the intervals be represented as pairs of integers for simplicity.
For example, let the given set of intervals be {{1,3}, {2,4}, {5,7}, {6,8} }. The intervals {1,3} and {2,4} overlap with each other, so they should be merged and become {1, 4}. Similarly {5, 7} and {6, 8} should be merged and become {5, 8}

Write a function which produces the set of merged intervals for the given set of intervals.(score 2)

```
class Solution{

    public List<Interval> merge(List<Interval> intervals) {

// add your code here

}

}
```

5.There are two sorted arrays nums1 and nums2 of size m and n respectively. Find the median of the two sorted arrays. (Score 2)
Example 1:
  nums1 = [1, 3]
  nums2 = [2]
  The median is 2.0

Example 2:
  nums1 = [1, 2]
  nums2 = [3, 4]
  The median is (2 + 3)/2 = 2.5

Here is the prototype you can work with.

```
Class solution{

    public double findMedianSortedArrays(int[] nums1, int[]
nums2) {

    // add your code here

  }
}
```

Extra credit

Write a Java function to remove vowels in a string. (Score 2)

i. The function should take a string as input.

ii. Should return the input string after omitting the vowels. Here is the prototype you can work with

```
        public String removeVowelsFromString(String
input){

        // add your code here
        }
```