LAB-9

NOTE: Share screenshots of results as well.

Q.1

a) Use the following unsynchronized counter class, which you can include as a nested class in your program:

```
static class Counter {
   int count;
   void inc() {
      count = count+1;
   }
   int getCount() {
      return count;
   }
}
```

Write a thread class that will repeatedly call the inc() method in an object of type Counter. The object should be a shared global variable. Create several threads, start them all, and wait for all the threads to terminate. Print the final value of the counter, and see whether it is correct.
Let the user enter the number of threads and the number of times that each thread will increment the counter. You might need a fairly large number of increments to see an error. And of course there can never be any error if you use just one thread. Your program can use join() to wait for a thread to terminate.

b) Now, synchronise these threads such that the correct value of counter variable is printed, no matter how large the number of threads and the number of increments is given.

Q.2

Write a program which processes **push()** and **pop()** operations in a stack of size 20 concurrently. Push random numbers between 0 and 100 into the stack. There should be a random interval of less than 1 seconds between two push operations. Similarly, between two pop operations there should be a random interval of less than 1.5 seconds. If the stack is empty while performing pop(), display the message: “Nothing to pop”. If the stack is full while performing push(), the number will fail to be pushed. So, display the message “Cannot push. Limit reached”. In this case, once the space is created due to the concurrent pop operation, push the same number which failed to be pushed earlier because the stack was full.

Q. 3

The following Java program should allow the user to compute sums of integers given in input. The program repeatedly asks the user if he/she wants to calculate a sum; if the answer is yes, then the program takes in input a sequence of integers terminated by a 0, then prints its sum, and asks the same question again. If the answer is no, then the program terminates.

```
import java.io.*;

public class Sums {

  public static void sum(BufferedReader in){
    // takes a sequence of integers in input, and outputs their sum

      int s, nextInt;
      s = 0;

      System.out.println("Please input the sequence of integers to sum, terminated by a 0");
    nextInt = Integer.parseInt(in.readLine());
        //Read the next datum in input. An integer is expected

      while (nextInt!=0) {
        s = s + nextInt;
        nextInt = Integer.parseInt(in.readLine());
      }

    System.out.println("The sum is " + s);
  }
```

```
   public static void main(String[] arg) {

         BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
                  //"in" will receive data from the standard input stream
         String c;

         System.out.println("Do you wish to calculate a sum? (y/n)");

         c = in.readLine();
       //Read next datum in input. A string "y" or "n" is expected

         while (!c.equals("y") && !c.equals("n")) {
            System.out.println("Please answer y or n");
            c = in.readLine();
         }

         while (c.equals("y")) {
            sum(in);
            System.out.println("Do you wish to calculate another sum? (y/n)");
            c = in.readLine();

            while (!c.equals("y") && !c.equals("n")) {
                  System.out.println("Please answer y or n");
                  c = in.readLine();
            }
         }

         System.out.println("Goodbye");
   }
}
```

The program as it is written, however, will not even pass compilation, because of uncaught (checked) exceptions.

Please modify the program and add the suitable try-catch statement(s) so to cope with the following exceptions:

1. NumberFormatException: an exception of this kind means that the datum returned by in.readLine() does not represent an integer. If this occurs, the user should be asked to reenter the datum and continue.
2. IOException: an exception of this kind means that the readLine() operation could not be completed normally, for instance because of a coding error. Also in this case, the user should be asked to reenter the datum and continue.

Example of session:

```
java Sums

Do you wish to calculate a sum? (y/n)
y
Please input the sequence of integers to sum, terminated by a 0
20
5
a
Invalid number. Please reenter.
2
0
The sum is 27
Do you wish to calculate another sum? (y/n)
s
Please input y or n
y
Please input the sequence of integers to sum, terminated by a 0
870
30
0
The sum is 900
Do you wish to calculate another sum? (y/n)
y
Please input the sequence of integers to sum, terminated by a 0
0
The sum is 0
Do you wish to calculate another sum? (y/n)
n
Goodbye
```

Q.4

You are required to compute the power of a number by implementing a calculator. Create a class MyCalculator which consists of a single method long power(int n, int p). This method takes two integers, n and p, as parameters and finds n raised to p . If either n or p is negative, then the method must throw an exception which says "n or p should not be negative". Also, if both n and p are zero, then the method must throw an exception which says "n and p should not be zero"

For example, -4 and -5 would result in **java.lang.Exception: n or p should not be negative**.

Complete the function power in class MyCalculator and return the appropriate result after the power operation

or an appropriate exception as detailed above.