Exercise 11

Objectives

In this exercise, you will study how to write Java code for exceptions.

A: Study a Code Example for the try and catch Blocks [2 pt]

Study the program *FinallyClause.java* below. Run the program and analyse the result. Write your answers to the following questions:

1. Why does the program print the two lines below?

```
(2) Exception occurred
(2) in finally clause
```

2. Why does the program print the line below (It looks like the program does not print the line because of the break statement)?

```
(3) in finally clause
```

3. Why does the program print the line below?

```
Main program ends
```

```
public class FinallyClause {
    static int count = 0;
    public static void main(String[] args) {
        while (true) {
            try {
                if (++count == 2) throw new Exception();
                if (count == 3) break;
                System.out.println("(" + count + ") No exception");
            } catch (Exception e) {
                System.out.println("(" + count + ") Exception occurred");
            } finally {
                System.out.println("(" + count + ") in finally clause");
            }
        } // end while
        System.out.println("Main program ends");
    }
}
```

Submission Files	Types
FinallyClause.txt	Text File

B: Write a Program Dealing with Exceptions [2 pt]

The following program assigns the numbers from 0 to 9 to an array object *vector*, which can store 10 integers, and then outputs them in order. However, an exception occurs due to a bug that causes excessive references to the array elements in the output process.

```
import java.io.*;

public class ListOfNumbers {
   private static final int SIZE = 10;
   private int[] vector;

   public ListOfNumbers () {
      vector = new int[SIZE];
      for (int i = 0; i < SIZE; i++)  vector[i] = i;
   }

   public void writeList() {
      for (int i = 0; i <= SIZE; i++){
         System.out.println("Value at: " + i + " = " + vector[i]);
      }
   }

   public static void main(String[] args){
       ListOfNumbers l = new ListOfNumbers();
       l.writeList();
   }
}
```

In the method writeList(), add exception handling using try, catch, and finally blocks, and modify the program so that the output shown below is obtained. Output the message "Entering block" to indicate that the process has moved to each block.

In the catch block, catch the ArrayIndexOutOfBoundsException and output the message by the method getMessage. Note that you must not modify the code (`i <= SIZE`) that is the source of the bug.

```
Entering try block.
Value at: 0 = 0
Value at: 1 = 1
Value at: 2 = 2
Value at: 3 = 3
Value at: 4 = 4
Value at: 5 = 5
Value at: 6 = 6
Value at: 7 = 7
Value at: 8 = 8
Value at: 9 = 9
Entering catch block.
Caught ArrayIndexOutOfBoundsException: 10
Entering finally block.
```

Submission Files	Types
ListOfNumbers.java	Java Class

C: Stack with Exception Handling [3 pt]

Extend the Stack data structure created in ex08 by adding exception handling. Create StackUnderflowException and StackOverflowException classes so that the following output is obtained when the following StackApplication is executed.

```
class StackApplication{
    public static void main(String[] args){
        Stack stack = new Stack(5);
        int[] a = {8, 0, 0, 5, 2, 4, 8, 6, 7, 0, 0};

        for ( int i = 0; i < a.length; i++ ){
            try{
                if ( a[i] == 0 ) {
                    int v = stack.pop();
                    System.out.println("pop " + v);
                } else {
                    stack.push(a[i]);
                    System.out.println("push " + a[i]);
                }
            } catch (StackUnderflowException e){
                System.out.println(e.getMessage());
            } catch (StackOverflowException e){
                System.out.println(e.getMessage());
            }
        }
    }
}
```

```
push 8
pop 8
Underflow. There is no element in the stack.
push 5
push 2
push 4
push 8
push 6
Overflow. 7 can not be pushed.
pop 6
pop 8
```

Note that you must not modify StackApplication class to obtain the above result.

Submission Files	Types
StackApplication.java	Java Class
Stack.java	Java Class
StackOverflowException.java	Java Class
StackUnderflowException.java	Java Class

Summary

In this exercise, we learned how to handle exceptions. We also studied how to define our own exception classes and throw and catch them. Through some application of data structure, we learned how we can separate codes for error handling and codes for the main stream.