Programming Paradigms Laboratory

B.Tech.

Name : **KAUSHAL VASHISTH**

Roll Number : **18ETCS002147**

Department : **Computer Science and Engineering**

Faculty of Engineering & Technology

Ramaiah University of Applied Sciences

Faculty	Engineering & Technology
Programme	B. Tech. in Computer Science and Engineering
Year/Semester	2nd Year / 4th Semester
Name of the Laboratory	Programming Paradigms Laboratory
Laboratory Code	19CSL217A

Name: KAUSHAL VASHISTH Registration No.: 18ETCS002147

Laboratory 8

Title of the Laboratory Exercise: Interface and Exception handling

1. Questions

    a. Develop a java interface for Stack ADT and implement it using class Array.

    b. Develop a Java program using exception handling to input five array elements through command line and find the sum and average by throwing ArrayIndexOutOfBoundException.

    c. Develop a Java program using exception handling to throw NumberFormatException using Integer.parseInt to find sum of two input strings by typecasting them into integers.

2. Calculations/Computations/Algorithms

```java
package stack_adt;
import java.util.*;
interface Stack
{
    public abstract void push(int b);
    public abstract void display();
    public abstract void pop();

}
class Array implements Stack
{
    int arr[] =new  int[10];
    int top =0;
    public  void push(int b)
    {
        top++;
        if(top>=10)
        {
            System.out.println("Stack is full");
        }
        else
        {
            System.out.println("Enter the element to be pushed:"+b);
            arr[top] = b;
        }
    }
    public  void pop()
    {
        if(top == -1)
        {
            System.out.println("Stack is empty");
        }
        else
        {
            int c = arr[top];
            top--;
            System.out.println("The deleted elemnt is:"+c);
        }

    }
```

```java
    public void display(){
        System.out.println("The stack elements are:");
        for(int i =top;i>0;i--)
        {
            System.out.println(arr[i]);
        }
    }
}
public class Stack_ADT {

    public static void main(String[] args) {
        Array obl = new Array();
        obl.push(27);
        obl.push(72);
        obl.push(98);
        obl.push(45);
        obl.push(65);
        obl.push(34);
        obl.push(92);
        obl.push(38);
        obl.pop();
        obl.display();
        obl.pop();
        obl.display();

    }

}
```

Figure 8.1

Figure 8.1 represent interface for Stack ADT and implement it using class Array.

```java
package finding_exception;
import java.util.*;

public class Finding_exception {

    public static void main(String[] args) {
        try
        {
            for(int i=0;i<5;i++)
            {
                System.out.println(args[i]);
            }
            int[]arr = new int[5];
            int total =0;
              for(int i=0;i<5;i++)
            {
                total = total+Integer.parseInt(args[i]);
            }
              System.out.println("Sum of the elements is:"+total);
            int average =0;
            for(int i=0;i<5;i++)
            {
                average=total/5;
            }
            System.out.println("The Average is:"+average);
        }
        catch(Exception n)
        {
            System.out.println("Exception occurred"+n);
        }
        finally{
            System.out.println("Successful");
        }
    }

}
```

Figure 8.2

Figure 8.2 represent program using exception handling to input five array elements through command line and find the sum and average by throwing ArrayIndexOutOfBoundException.

```java

package demoproj;
import java.util.*;

public class Demoproj {


    public static void main(String[] args) {
        int total = 0;
        Scanner sl = new Scanner(System.in);
        System.out.println("Enter the name");
        String a = sl.nextLine();
        String b = sl.nextLine();

        try
        {
            total = Integer.parseInt(a)+Integer.parseInt(b);
            System.out.println(total);
        }
        catch(Exception n)
        {
            System.out.println("Exception ocuured is:"+n);
        }
        finally
        {
            System.out.println("succes");
        }

    }

}

```

Figure 8.3

Figure 8.3 represent sum of two input string using exception handling

3. Presentation of Results

```
run:
Enter the element to be pushed:27
Enter the element to be pushed:72
Enter the element to be pushed:98
Enter the element to be pushed:45
Enter the element to be pushed:65
Enter the element to be pushed:34
Enter the element to be pushed:92
Enter the element to be pushed:38
The deleted elemnt is:38
The stack elements are:
92
34
65
45
98
72
27
The deleted elemnt is:92
The stack elements are:
34
65
45
98
72
27
BUILD SUCCESSFUL (total time: 0 seconds
```

Figure 8.4 represent output for stack using interface.

```
run:
Exception occurredjava.lang.ArrayIndexOutOfBoundsException: 0
Successful
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 8.5 represent output exception handling.

```
run:
4
5
Exception occuredjava.lang.ArrayIndexOutOfBoundsException: 2
success
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 8.6 represent output exception handling.

```
run:
Enter the name
abdul
sanketh
Exception ocuured is:java.lang.NumberFormatException: For input string: "abdul"
succes
BUILD SUCCESSFUL (total time: 6 seconds)
```

Figure 8.7 represent output for sum of two input string using exception handling.

4. Conclusions

Another way to achieve abstraction in Java, is with interfaces.

An interface is a completely "abstract class" that is used to group related methods with empty bodies:

There are mainly three reasons to use interface.

1. It is used to achieve abstraction.

2. By interface, we can support the functionality of multiple inheritance.

3. It can be used to achieve loose coupling.

All Java exception classes inherit directly or indirectly from class Exception, forming an

Inheritance hierarchy.

5. Limitations of Experiments and Results

Java interfaces are slower and more limited than other ones. - Interface should be used multiple number of times else there is hardly any use of having them.