

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	2 nd Year / 4 th Semester
Name of the Laboratory	Programming Paradigms Laboratory
Laboratory Code	19CSL217A

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 `ArrayIndexOutOfBoundsException`.
- 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

```
1
2  package stack_adt;
3  import java.util.*;
4  interface Stack
5  {
6      public abstract void push(int b);
7      public abstract void display();
8      public abstract void pop();
9
10 }
11 class Array implements Stack
12 {
13     int arr[] =new int[10];
14     int top =0;
15     public void push(int b)
16     {
17         top++;
18         if(top>=10)
19         {
20             System.out.println("Stack is full");
21         }
22         else
23         {
24             System.out.println("Enter the element to be pushed:"+b);
25             arr[top] = b;
26         }
27     }
28     public void pop()
29     {
30         if(top == -1)
31         {
32             System.out.println("Stack is empty");
33         }
34         else
35         {
36             int c = arr[top];
37             top--;
38             System.out.println("The deleted elemnt is:"+c);
39         }
40     }
41 }
```

```
43     public void display(){
44         System.out.println("The stack elements are:");
45         for(int i =top;i>0;i--)
46         {
47             System.out.println(arr[i]);
48         }
49     }
50     public class Stack_ADT {
51
52     public static void main(String[] args) {
53         Array obl = new Array();
54         obl.push(27);
55         obl.push(72);
56         obl.push(98);
57         obl.push(45);
58         obl.push(65);
59         obl.push(34);
60         obl.push(92);
61         obl.push(38);
62         obl.pop();
63         obl.display();
64         obl.pop();
65         obl.display();
66     }
67 }
68
69
70
```

Figure 8.1

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

```
1
2 package finding_exception;
3 import java.util.*;
4
5 public class Finding_exception {
6
7     public static void main(String[] args) {
8         try
9         {
10             for(int i=0;i<5;i++)
11             {
12                 System.out.println(args[i]);
13             }
14             int[] arr = new int[5];
15             int total =0;
16             for(int i=0;i<5;i++)
17             {
18                 total = total+Integer.parseInt(args[i]);
19             }
20             System.out.println("Sum of the elements is:"+total);
21             int average =0;
22             for(int i=0;i<5;i++)
23             {
24                 average=total/5;
25             }
26             System.out.println("The Average is:"+average);
27         }
28         catch(Exception n)
29         {
30             System.out.println("Exception occurred"+n);
31         }
32         finally{
33             System.out.println("Successful");
34         }
35     }
36 }
37
38
```

Figure 8.2

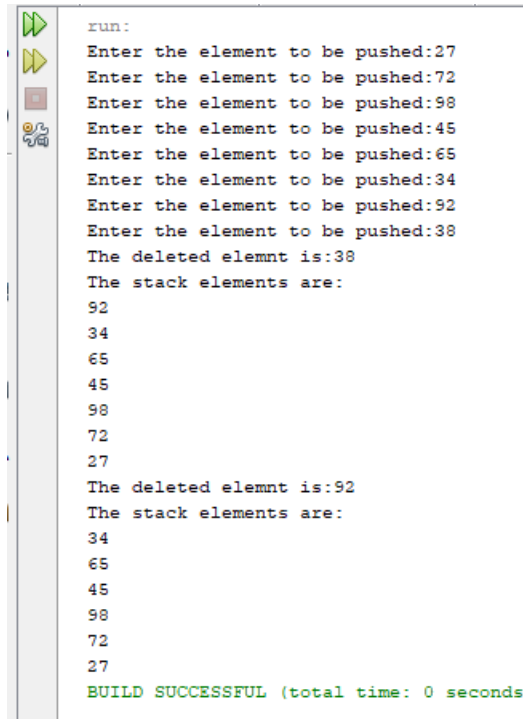
Figure 8.2 represent program using exception handling to input five array elements through command line and find the sum and average by throwing `ArrayIndexOutOfBoundsException`.

```
1
2 package demoproj;
3 import java.util.*;
4
5 public class Demoproj {
6
7
8     public static void main(String[] args) {
9         int total = 0;
10        Scanner s1 = new Scanner(System.in);
11        System.out.println("Enter the name");
12        String a = s1.nextLine();
13        String b = s1.nextLine();
14
15        try
16        {
17            total = Integer.parseInt(a)+Integer.parseInt(b);
18            System.out.println(total);
19        }
20        catch(Exception n)
21        {
22            System.out.println("Exception ocuured is:"+n);
23        }
24        finally
25        {
26            System.out.println("succes");
27        }
28    }
29 }
30
31
32
```

Figure 8.3

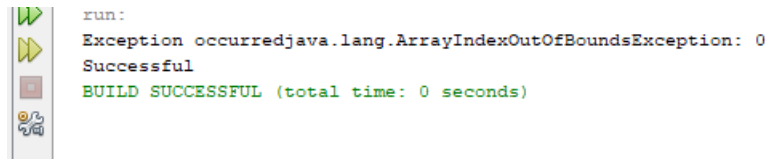
Figure 8.3 represent sum of two input string using exception handling

3. Presentation of Results



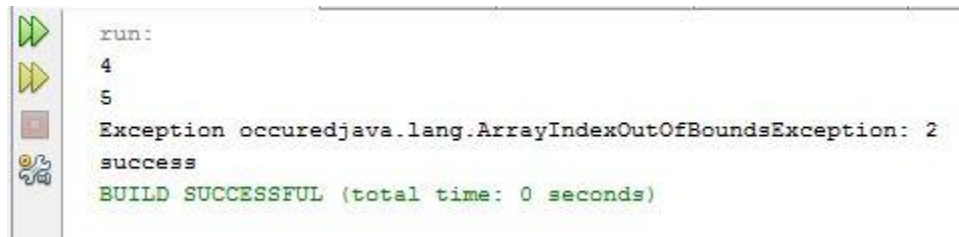
```
run:
Enter the element to be pushed:27
Enter the element to be pushed:72
Enter the element to be pushed:98
Enter the element to be pushed:45
Enter the element to be pushed:65
Enter the element to be pushed:34
Enter the element to be pushed:92
Enter the element to be pushed:38
The deleted elemnt is:38
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.



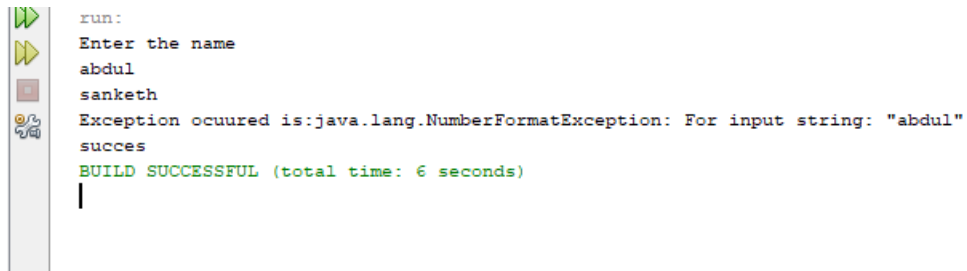
```
run:
Exception occurredjava.lang.ArrayIndexOutOfBoundsException: 0
Successful
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 8.5 represent output exception handling.



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

Figure 8.6 represent output exception handling.



```
run:
Enter the name
abdul
sanketh
Exception occurred 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.