

Name: Dev Kamlesh Bhanushali

PRN: 22070126032

Java Assignment 6

Part - 1

Stack Interface

```
1 package Java.Assignments.Assignment6.Part1;
2
3 public class MainClass {
4     public static void main(String[] args) {
5         // Fixed f_stack
6         System.out.println("Fixed Stack Output -----");
7         FixedStack f_stack = new FixedStack(12);
8         f_stack.push(0);
9         f_stack.push(2);
10        f_stack.push(4);
11        f_stack.push(6);
12        f_stack.push(8);
13        f_stack.display();
14        f_stack.pop();
15        f_stack.push(10);
16        f_stack.pop();
17        f_stack.pop();
18        f_stack.pop();
19        f_stack.display();
20        f_stack.pop();
21        f_stack.pop();
22        f_stack.display();
23        System.out.println(f_stack.isOverflow());
24        System.out.println(f_stack.isUnderflow());
25
26        try {
27            f_stack.pop();
28        } catch (RuntimeException e) {
29            System.out.println(e.getMessage());
30        }
31
32        System.out.println("\n");
33
34        // Dynamic f_stack
35        System.out.println("Dynamic Stack Output -----");
36        DynamicStack d_stack = new DynamicStack();
37        d_stack.push(0);
38        d_stack.push(2);
39        d_stack.push(4);
40        d_stack.push(6);
41        d_stack.push(8);
42        d_stack.display();
43        d_stack.pop();
44        d_stack.push(10);
45        d_stack.pop();
46        d_stack.pop();
47        d_stack.pop();
48        d_stack.display();
49        d_stack.pop();
50        d_stack.pop();
51        d_stack.display();
52        System.out.println(d_stack.isOverflow());
53        System.out.println(d_stack.isUnderflow());
54
55        try {
56            d_stack.pop();
57        } catch (RuntimeException e) {
58            System.out.println(e.getMessage());
59        }
60    }
61 }
62
```



```
1  package Java.Assignments.Assignment6.Part1;
2
3  public interface StackInterface{
4      public abstract boolean isOverflow();
5      public abstract boolean isUnderflow();
6      public abstract void pop();
7      public abstract void push(int x);
8      public abstract void display();
9  }
```

```

1  package Java.Assignments.Assignment6.Part1;
2
3  public class FixedStack implements StackInterface{
4
5      private int sizeState = 0;
6      // sizeState = 0 means normal stack state
7      // sizeState = -1 means stack underflow
8      // sizeState = 1 means stack overflow
9      private int insertPos = 0;
10     private int[] arr;
11
12     FixedStack(int size){
13         if(size < 1){
14             throw new IllegalArgumentException("Fixed Stack must have size greater than 0");
15         }
16
17         // initialize array and values
18         this.arr = new int[size];
19         for (int i = 0; i < this.arr.length; i++) {
20             this.arr[i] = Integer.MIN_VALUE;
21         }
22     }
23
24     @Override
25     public boolean isUnderflow(){
26         if(sizeState == -1) return true;
27         return false;
28     }
29
30     @Override
31     public boolean isOverflow(){
32         if(sizeState == 1) return true;
33         return false;
34     }
35
36     @Override
37     public void pop() {
38         if(sizeState > -1){
39             this.arr[insertPos - 1] = Integer.MIN_VALUE;
40             this.insertPos--;
41
42             if(this.insertPos == 0){
43                 this.sizeState = -1;
44             }
45         }
46         else{
47             throw new RuntimeException("Stack Underflow");
48         }
49     }
50
51     @Override
52     public void push(int value) {
53         if(sizeState < 1){
54             this.arr[insertPos] = value;
55             this.insertPos++;
56
57             if(this.insertPos >= this.arr.length){
58                 this.sizeState = 1;
59             }
60         }
61         else{
62             throw new RuntimeException("Stack Overflow");
63         }
64     }
65
66     @Override
67     public void display() {
68         for (int index = 0; index < this.arr.length; index++) {
69             if(this.arr[index] != Integer.MIN_VALUE)
70                 System.out.println(String.format("Element %d: %d", index, this.arr[index]));
71         }
72         System.out.println("\n");
73     }
74
75 }
76

```

```
1 package Java.Assignments.Assignment6.Part1;
2
3 import java.util.ArrayList;
4
5 public class DynamicStack implements StackInterface{
6
7     private ArrayList<Integer> arr = new ArrayList<Integer>();
8
9     @Override
10    public boolean isOverflow() {
11        return false;
12    }
13
14    @Override
15    public boolean isUnderflow() {
16        if(!(this.arr.size() > 0)){
17            return true;
18        }
19        return false;
20    }
21
22    @Override
23    public void pop() {
24        if(this.arr.size() > 0){
25            this.arr.remove(this.arr.size() - 1);
26        }
27        else{
28            throw new RuntimeException("Stack Underflow");
29        }
30    }
31
32    @Override
33    public void push(int value) {
34        this.arr.add(value);
35    }
36
37    @Override
38    public void display() {
39        for (int i = 0; i < this.arr.size(); i++) {
40            System.out.println(String.format("Element %d: %d", i, this.arr.get(i)));
41        }
42        System.out.println("\n");
43    }
44
45
46 }
47
```

Output



```
1  Fixed Stack Output -----
2  Element 0: 0
3  Element 1: 2
4  Element 2: 4
5  Element 3: 6
6  Element 4: 8
7
8
9  Element 0: 0
10 Element 1: 2
11
12
13
14
15 false
16 true
17 Stack Underflow
18
19
20 Dynamic Stack Output -----
21 Element 0: 0
22 Element 1: 2
23 Element 2: 4
24 Element 3: 6
25 Element 4: 8
26
27
28 Element 0: 0
29 Element 1: 2
30
31
32
33
34 false
35 true
36 Stack Underflow
```

Github: <https://github.com/devilb2103/Sem-4/tree/main/Java/Assignments/Assignment6>