Sukkur IBA University
Department of Computer Science

DATA STRUCTURES
Undo and Redo Application

Instructor: Saif Hassan

READ IT FIRST

Prior to start solving the problems in this assignments, please give full concentration on following points.

1. WORKING – This is individual lab. If you are stuck in a problem contact your teacher, but, in mean time start doing next question (don't waste time).
2. DEADLINE – 11th March, 2022
3. SUBMISSION – This assignment needs to be submitted in a soft copy.
4. WHERE TO SUBMIT – Please visit your LMS.
5. WHAT TO SUBMIT – Submit this docx and pdf file.

KEEP IT WITH YOU!

1. Indent your code inside the classes and functions. It's a good practice!
2. It is not bad if you keep your code indented inside the loops, if and else blocks as well.
3. Comment your code, where it is necessary.
4. Read the entire question. Don't jump to the formula directly.

I, **Amjad Ali** with student ID **191-21-0001**

Section _**A**_hereby declare that I do understand the instructions above and follow them. This is

my own work.

Exercises

Task1 Description

Task (Undo/Redo Program)

You know about Undo and Redo operations almost in every software such as: MS Word, Excel and etc. You have to implement Undo/Redo operations using linked list, you can use linked list code which you did in class/lab. After completion of this task, program flow will be as follows:

Please enter your choice: (1 for insert, 2 for undo, 3 for redo, 4 display stack)

Choice: 1	**Input1**: This
Choice: 1:	**Input2**: is
Choice: 1:	**Input3**: my
Choice: 1:	**Input4**: stack
Choice: 2	**Output**: Undo successful
Choice: 2	**Output**: Undo successful
Choice: 4	**Output**: is This
Choice: 3	**Output**: Redo Successful
Choice: 4	**Output**: my is This

Choice: 3	**Output**: Redo Successful
Choice: 4	**Output**: stack my is This
Choice: 3	**Output**: Redo Unsuccessful
Choice: 2	**Output**: Undo successful
Choice: 2	**Output**: Undo successful
Choice: 2	**Output**: Undo successful
Choice: 2	**Output**: Undo successful
Choice: 2	**Output**: Stack is Empty
And so on	

Solution:

Stack Class

```
1. package com.company;
2.
3.
4.
5. import java.util.*;
6.
7. // A linked list node
8. class Node {
9.     String data; // integer data
10.     Node next; // pointer to the next node
11.
12.     Node(String data) {
13.         this.data = data;
```

```
14.         this.next = null;
15.     }
16. }
17.
18. class Stack {
19.     private Node top, tail;
20.
21.     Stack() {
22.         this.top = null;
23.         this.tail = null;
24.     }
25.
26.     // Utility function to add an element x in the
  stack
27.     public void push(String x) // insert at the
  beginning
28.     {// Write your code here
29.         Node newNode = new Node(x);
30.         if (isEmpty()) {
31.             top = tail = newNode;
32.         } else {
33.             newNode.next = top;
34.             top = newNode;
35.         }
36.     }
37.
38.
39.     // Utility function to check if the stack is empty
  or not
40.     public boolean isEmpty() {
41.         // Write your code here
42.         return top == null;
43.     }
44.
45.     // Utility function to return top element in a
  stack
46.     public String peek() {
47.         // Write your code here
48.         if (isEmpty()) {
```

```
49.             System.out.println("Stack underflow");
50.             return "-1";
51.         } else {
52.             return top.data;
53.         }
54.
55.     }
56.
57.     // Utility function to pop top element from the
  stack and check for Stack underflow
58.
59.     public String pop() // remove at the beginning
60.     {// Write your code here
61.         if (isEmpty()) {
62.             System.out.println("Stack underflow");
63.             return "-1";
64.         } else {
65.
66.             String temp = top.data;
67.             top = top.next;
68.             return temp;
69.
70.         }
71.
72.     }
73.
74.     public String toString()
75.     {
76.         ArrayList<String> list=new ArrayList<>();
77.         while(!isEmpty())
78.         {
79.             String str=pop();
80.             list.add(str);
81.         }
82.
83.         int size=list.size();
84.         String result="[ ";
85.         for(int i=0;i<size;i++)
86.         {
```

```
87.             String str = list.get(i);
88.             push(str);
89.             result+=str+" ";
90.
91.         }
92.         result+="]";
93.
94.             return result;
95.     }
96. }
```

Undo Redo implementation

```
1. package com.company;
2.
3. public class UndoRedoImplementation {
4.
5.
6.     static Stack ordinary = new Stack();
7.     static Stack UndoRedo = new Stack();
8.
9.     public void insert(String x) {
10.         ordinary.push(x);
11.     }
12.
13.     public void undo() {
14.         if (ordinary.isEmpty()) {
15.             System.out.print("Can't call the function
  because Stack is Empty");
16.         } else {
17.             UndoRedo.push(ordinary.pop());
18.             System.out.println("Undo Successful");
19.         }
20.     }
21.
22.     public void redo() {
23.         if (UndoRedo.isEmpty()) {
```

```
24.             System.out.print("Can't call the function
  because Stack is Empty");
25.         } else {
26.             ordinary.push(UndoRedo.pop());
27.             System.out.println("Redo Successful");
28.         }
29.     }
30.
31.     public void displayStack() {
32.         System.out.println(ordinary);
33.     }
34. }
```

DemoClass

```
1. import com.company.UndoRedoImplementation;
2.
3. import java.util.Scanner;
4. import java.util.Stack;
5.
6. public class DemoRedo {
7.
8.     public static void main(String[] args) {
9.         Scanner sc = new Scanner(System.in);
10.         UndoRedoImplementation UndoRedo = new
  UndoRedoImplementation();
11.         int a = 0;
12.         int counter = 1;
13.         System.out.println("Please Enter Your Choice:
  (1 for insert(), 2 for undo() , 3 for Redo(), 4 for
  displayStack() and 5 for Exit()");
14.         while (a != 5) {
15.             System.out.print("Choice: ");
16.             a = sc.nextInt();
17.             if (a == 1) {
18.                 System.out.print("Input" + counter +
  ":  ");
```

```
19.                 String str = sc.next();
20.                 UndoRedo.insert(str);
21.             } else if (a == 2) {
22.                 UndoRedo.undo();
23.             } else if (a == 3) {
24.                 UndoRedo.redo();
25.             } else if (a == 4) {
26.                 UndoRedo.displayStack();
27.             } else if (a != 5) {
28.                 System.out.println("Invalid input");
29.             }
30.
31.         }
32.
33.
34.     }
35.
36. }
```

Output:-

```
Choice: 1
Input1:  this
Choice: 1
Input1:  is
Choice: 1
Input1:  my
Choice: 1
Input1:  stack
Choice: 2
Undo Successful
Choice: 2
Undo Successful
Choice: 4
[ is, this, ]
Choice: 3
Redo Successful
Choice: 4
[ my, this, is, ]
Choice: 3
Redo Successful
Choice:
```