**Assingment-6**

**Code-**

public interface IntStack {  
 void push(int data);  
 int pop();  
 boolean isEmpty();  
}

public class FixedStack implements IntStack {  
 private int stack[];  
 private int top;  
  
 public FixedStack(int len) {  
 stack = new int[len];  
 top = -1;  
 }  
  
 @Override  
 public void push(int data) {  
 if (top == stack.length - 1) {  
 System.*out*.println(" Stack Overflow");  
 } else {  
 stack[++top] = data;  
 System.*out*.println(" Data Added!");  
 }  
 }  
  
 @Override  
 public int pop() {  
 if (top < 0) {  
 System.*out*.println(" Stack Underflow");  
 return 0;  
 } else {  
 return stack[top--];  
 }  
 }  
  
 @Override  
 public boolean isEmpty() {  
 return top < 0;  
 }  
}

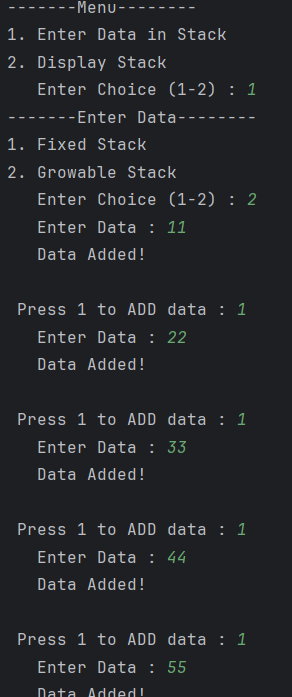
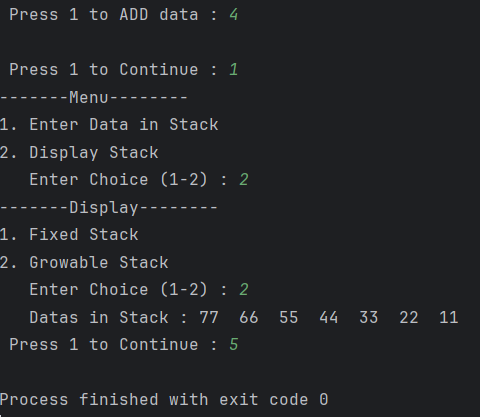
public class GrowableStack implements IntStack {  
 private int stack[];  
 private int top;  
  
 public GrowableStack(int len) {  
 stack = new int[len];  
 top = -1;  
 }  
  
 @Override  
 public void push(int data) {  
 if (top == stack.length - 1) {  
 int temp[] = new int[stack.length \* 2];  
 for (int i = 0; i < stack.length; i++) {  
 temp[i] = stack[i];  
 }  
 stack = temp;  
 }  
 stack[++top] = data;  
 System.*out*.println(" Data Added!");  
 }  
  
 @Override  
 public int pop() {  
 if (top < 0) {  
 System.*out*.println(" Stack Underflow");  
 return 0;  
 } else {  
 return stack[top--];  
 }  
 }  
  
 @Override  
 public boolean isEmpty() {  
 return top < 0;  
 }  
}

import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.*in*);  
  
 int st = 1, ch, hh, size, data;  
  
 System.*out*.println("------STACK------\n");  
  
 System.*out*.print(" Enter Initial Length of Stack : ");  
 size = scan.nextInt();  
 IntStack fixed = new FixedStack(size);  
 IntStack growable = new GrowableStack(size);  
 System.*out*.println("\n");  
  
 while (st == 1) {  
 System.*out*.println("-------Menu--------");  
 System.*out*.println("1. Enter Data in Stack");  
 System.*out*.println("2. Display Stack");  
 System.*out*.print(" Enter Choice (1-2) : ");  
 ch = scan.nextInt();  
  
 switch (ch) {  
 case 1:  
 System.*out*.println("-------Enter Data--------");  
 System.*out*.println("1. Fixed Stack");  
 System.*out*.println("2. Growable Stack");  
 System.*out*.print(" Enter Choice (1-2) : ");  
 hh = scan.nextInt();  
 while (true) {  
 switch (hh) {  
 case 1:  
 System.*out*.print(" Enter Data : ");  
 data = scan.nextInt();  
 fixed.push(data);  
 break;  
 case 2:  
 System.*out*.print(" Enter Data : ");  
 data = scan.nextInt();  
 growable.push(data);  
 break;  
 default:  
 System.*out*.println(" Invalid Choice");  
 }  
 System.*out*.print("\n Press 1 to ADD data : ");  
 if (scan.nextInt() != 1) break;  
 }  
 break;  
 case 2:  
 System.*out*.println("-------Display--------");  
 System.*out*.println("1. Fixed Stack");  
 System.*out*.println("2. Growable Stack");  
 System.*out*.print(" Enter Choice (1-2) : ");  
 hh = scan.nextInt();  
 switch (hh) {  
 case 1:  
 System.*out*.print(" Datas in Stack : ");  
 if (fixed.isEmpty()) {  
 System.*out*.print(" Stack Underflow ");  
 } else {  
 while (!fixed.isEmpty()) {  
 System.*out*.print(fixed.pop() + " ");  
 }  
 }  
 break;  
 case 2:  
 System.*out*.print(" Datas in Stack : ");  
 if (growable.isEmpty()) {  
 System.*out*.print(" Stack Underflow ");  
 } else {  
 while (!growable.isEmpty()) {  
 System.*out*.print(growable.pop() + " ");  
 }  
 }  
 break;  
 default:  
 System.*out*.println(" Invalid Choice");  
 }  
 break;  
 default:  
 System.*out*.println(" Invalid Choice");  
 }  
 System.*out*.print("\n Press 1 to Continue : ");  
 st = scan.nextInt();  
 }  
 }  
}

**Output-**

****

**Output-**

**  
**

**GITHUB LINK :<https://github.com/RiyaGupta122/Implementation-of-Fixed-and-Growable-Stack/tree/main>**