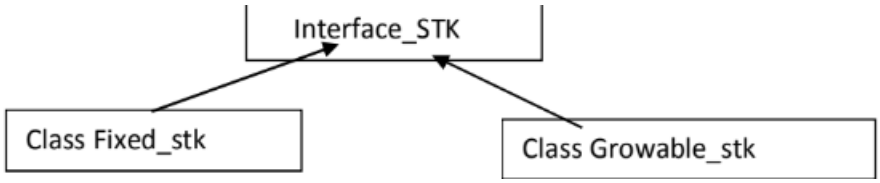
PROGRAMMING IN JAVA LAB-6

Name: Aditya Sable

PRN: 21070126009

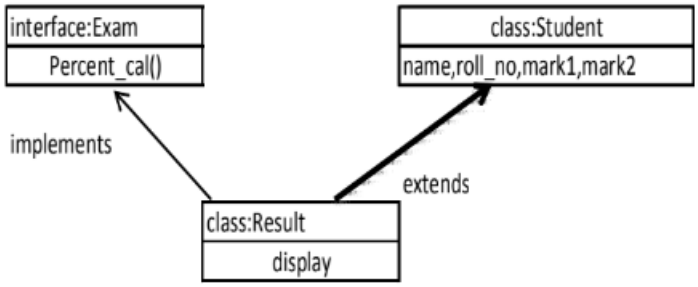
Batch: AIML A1

Part 1: An implementation of IntStack (integer stack) that uses fixed storage as well as "growable" using interface.



Create a user defined package "pkg_Stack" where the interface is stored. The other two complete classes will need to import the package 'pkg_Stack' and then use it.

Part 2: Program to implement the following Multiple Inheritance.



//

PART-1

#GROWABLE STACK

```
package fixed_grow_stack;

import fixed_grow_stack.pkg_Stack.Interface_STK;
import java.util.Vector;

public class Growable_stk implements Interface_STK{

    // creating Vector of type Integer
    Vector<Integer> grow_STK = new Vector<Integer>();

    @Override
    public void push(int a) {
        grow_STK.add(a);
    }
```

```
    @Override
    public int pop() {
        if(grow_STK.isEmpty()){
            System.out.println("Stack is empty - Cannot remove element");
            return 0;
        }
        else{
            return grow_STK.remove(grow_STK.size()-1);
        }
    }

    @Override
    public int peek() {
        return grow_STK.get(grow_STK.size()-1);
    }

    @Override
    public boolean isEmpty() {
        if(grow_STK.isEmpty()){
            return true;
        }
        else{
            return false;
        }
    }

    @Override
    public boolean isFull() {
        System.out.println("Growable stack is never full");
        return false;
    }

    @Override
    public void clear() {
        grow_STK.clear();
        System.out.println("Stack is cleared");
    }

    @Override
    public int size() {
        return(grow_STK.size());
    }

    @Override
    public void display() {
        System.out.println("Stack elements are:");
        for(int i=0; i<grow_STK.size(); i++){
            System.out.println(grow_STK.get(i));
        }
    }
}
```

#FIXED STACK

```
package fixed_grow_stack;

import fixed_grow_stack.pkg_Stack.Interface_STK;

public class Fixed_stk implements Interface_STK{
```

```
    int fix_STK[] = new int[MAX];
    int top = 0;

    @Override
    public void push(int a){
        if(top==MAX){
            System.out.println("Stack is full - Cannot insert element");
            return;
        }
        else{
            fix_STK[top] = a;
            top++;
        }
    }

    @Override
    public int pop() {
        if(top==0){
            System.out.println("Stack is empty - Cannot remove element");
            return 0;
        }
        else{
            return fix_STK[--top];
        }
    }

    @Override
    public int peek() {
        return fix_STK[top--];
    }

    @Override
    public boolean isEmpty() {
        if(top==0){
            return true;
        }
        else{
            return false;
        }
    }

    @Override
    public boolean isFull() {
        if(top==MAX){
            return true;
        }
        else{
            return false;
        }
    }

    @Override
    public void clear() {
        for(int i=0; i<top; i++){
            pop();
        }
        System.out.println("Stack is cleared");
    }

    @Override
    public int size() {
```

```
        return top;
    }

    @Override
    public void display() {
        System.out.println("Stack elements are: ");
        for(int i=0; i<top; i++){
            System.out.println(fix_STK[i]);
        }
    }
}
```

#MAIN CLASS

```
package fixed_grow_stack;
import java.util.Scanner;

public class master_main {
    public static void main(String[] args) {

        // Main menu
        Scanner sc = new Scanner(System.in);
        System.out.println("Choose sub-menu:");
        System.out.println("1. Fixed Stack");
        System.out.println("2. Growing Stack");
        System.out.println("3. Exit");
        System.out.print("\nEnter your choice: ");
        int choice = sc.nextInt();

        // Sub-menu for Fixed Stack [Fixed_stk.java]
        if(choice == 1){
            Fixed_stk stk = new Fixed_stk();
            System.out.println("\n\nSub-menu: Fixed Stack");
            System.out.println("Choose operation:");
            System.out.println("1. Push Element");
            System.out.println("2. Pop Element");
            System.out.println("3. Peek Element");
            System.out.println("4. Check if stack is empty");
            System.out.println("5. Check if stack is full");
            System.out.println("6. Clear stack");
            System.out.println("7. Display stack");
            System.out.println("8. Exit");
            System.out.print("\nEnter your choice: ");
            int choice_fix = sc.nextInt();

            if(choice_fix == 1){
                System.out.print("Enter element to push: ");
                int element_fix = sc.nextInt();
                stk.push(element_fix);
            }
            else if(choice_fix == 2){
                System.out.println("Popped element: " + stk.pop());
            }
            else if(choice_fix == 3){
                System.out.println("Peeked element: " + stk.peek());
            }
            else if(choice_fix == 4){
                System.out.println("Is stack empty? " + stk.isEmpty());
            }
            else if(choice_fix == 5){
```

```
                System.out.println("Is stack full? " + stk.isFull());
            }
            else if(choice_fix == 6){
                stk.clear();
            }
            else if(choice_fix == 7){
                stk.display();
            }
            else if(choice_fix == 8){
                System.exit(0);
            }
            else{
                System.out.println("Invalid choice");
            }

        }

        else if (choice == 2){
            Growable_stk stk = new Growable_stk();
            System.out.println("\n\nSub-menu: Growing Stack");
            System.out.println("Choose operation:");
            System.out.println("1. Push Element");
            System.out.println("2. Pop Element");
            System.out.println("3. Peek Element");
            System.out.println("4. Check if stack is empty");
            System.out.println("5. Check if stack is full");
            System.out.println("6. Clear stack");
            System.out.println("7. Display stack");
            System.out.println("8. Exit");
            System.out.print("\nEnter your choice: ");
            int choice_grow = sc.nextInt();

            if(choice_grow == 1){
                System.out.println("Enter element to push: ");
                int element_grow = sc.nextInt();
                stk.push(element_grow);
            }

            else if(choice_grow == 2){
                System.out.println("Popped element: " + stk.pop());
            }

            else if(choice_grow == 3){
                System.out.println("Peeked element: " + stk.peek());
            }

            else if(choice_grow == 4){
                System.out.println("Is stack empty? " + stk.isEmpty());
            }

            else if(choice_grow == 5){
                System.out.println("Is stack full? " + stk.isFull());
            }

            else if(choice_grow == 6){
                stk.clear();
            }

            else if(choice_grow == 7){
                stk.display();
            }
```

```
            else if(choice_grow == 8){
                System.exit(0);
            }

            else{
                System.out.println("Invalid choice");
            }


        }

        else if (choice == 3){
            System.exit(0);
        }

        else{
            System.out.println("Invalid choice");
        }
        sc.close();
    }
}
```

#INTERFACE

```
package fixed_grow_stack.pkg_Stack;

public interface Interface_STK {
    int MAX = 5; // maximum size of the stack
    public void push(int item); // push an item onto the stack
    public int pop(); // pop an item from the stack
    public int peek(); // peek at the top of the stack
    public boolean isEmpty(); // true if stack is empty
    public boolean isFull(); // true if stack is full
    public void clear(); // clear the stack
    public int size(); // return the number of items in the stack
    public void display(); // display the stack
}
```

OUTPUT:

```
Choose sub-menu:
1. Fixed Stack
2. Growing Stack
3. Exit

Enter your choice: 1


Sub-menu: Fixed Stack
Choose operation:
1. Push Element
2. Pop Element
3. Peek Element
4. Check if stack is empty
5. Check if stack is full
6. Clear stack
7. Display stack
8. Exit

Enter your choice: 7
Stack elements are:

Process finished with exit code 0
```

```
Choose sub-menu:
1. Fixed Stack
2. Growing Stack
3. Exit

Enter your choice: 2


Sub-menu: Growing Stack
Choose operation:
1. Push Element
2. Pop Element
3. Peek Element
4. Check if stack is empty
5. Check if stack is full
6. Clear stack
7. Display stack
8. Exit

Enter your choice: 1
Enter element to push:
12

Process finished with exit code 0
```

PART-2

```
public interface Exam {
    public double Percent_Cal();
}
public class Student implements Exam {
    private String name;
    private int rollNo;
    private String branch;

    private String subject;
    private int marks1;

    private int marks2;

    public Student(String name, int rollNo, String branch, String subject,
int marks1, int marks2) {
        this.name = name;
        this.rollNo = rollNo;
        this.branch = branch;

        this.subject = subject;
        this.marks1 = marks1;
        this.marks2 = marks2;
    }

    public void display() {
        System.out.println("Name: " + name);
        System.out.println("Roll No: " + rollNo);
        System.out.println("Branch: " + branch);
        System.out.println("Subject: " + subject);
        System.out.println("Marks1: " + marks1);
        System.out.println("Marks2: "+ marks2);
    }

    public double Percent_Cal() {
        return (double) (marks1+marks2) / 200 * 100;
```

```
    }
}
public class ResultPrinter {
    private Exam result;

    public ResultPrinter(Exam result) {
        this.result = result;
    }

    public void display() {
        System.out.println("Percentage: " + result.Percent_Cal());
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Student student = new Student("Anuj", 1220, "AIML", "Java", 95,
85);
        student.display();
        ResultPrinter printer = new ResultPrinter(student);
        printer.display();
    }
}
```

OUTPUT

```
Name: Anuj
Roll No: 1220
Branch: AIML
Subject: Java
Marks1: 95
Marks2: 85
Percentage: 90.0

Process finished with exit code 0
```

Github Link: https://github.com/adityasable22/pij-assignment-6.git