Q1. Stack Operations using Interface: Create an interface Stack with a variable size and abstract methods push(), pop(), display( ), overflow(), and underflow(). Implement a subclass IntegerStack that implements the Stack interface. Create a test class to check the working of all methods in the IntegerStack class.

```
import java.util.*;
class Student{
        int roll;
        int getroll(){
                return roll;
        }
        void setroll(int roll){
                this.roll = roll;
        }
}
class Test extends Student{
        int sub1;
        int sub2;
        int getmarks1(){
                return sub1;
        }
        int getmarks2(){
                return sub2;
        }
        void setmarks(int sub1, int sub2){
                this.sub1 = sub1;
                this.sub2 = sub2;
        }
}
interface Sports{
        int sMarks = 10;
        void set();
}
class Result extends Test implements Sports{
        int totalmarks;
                public void set(){};
        public void display(){
                totalmarks = getmarks1() +getmarks2() + sMarks;
                System.out.println("Roll no : "+getroll());
                System.out.println("Marks is subject 1 : "+getmarks1());
                System.out.println("Marks is subject 2 : "+getmarks2());
                System.out.println("Sports marks : "+sMarks);
                System.out.println("Total marks : "+totalmarks);
        }
}
public class student{
        public static void main(String[] args){
        Result r = new Result();
        r.setroll(111);
        r.setmarks(90,90);
        r.set();
        r.display();
        }
}
```

```
1 error
vishal@vishal-LOQ-15IRX9:~/vishal/java/Ass4$ javac Studentdemo.java
vishal@vishal-LOQ-15IRX9:~/vishal/java/Ass4$ java Studentdemo
Roll no : 111
Marks is subject 1 : 90
Marks is subject 2 : 90
Sports marks : 10
Total marks : 190
vishal@vishal-LOQ-15IRX9:~/vishal/java/Ass4$
```

Q2. Shape Interface with Rectangle and Triangle: Implement the following: a. Create an interface Shape with an abstract method area(). b. Create two classes, Rectangle and Triangle, that implement the Shape interface. c. Calculate and display the area of both Rectangle and Triangle.

```
import java.util.*;
interface Shape{
   public double area();
}
class rect implements Shape{
double dim1;
double dim2;
public rect(double height, double width) {
      dim1 = height;
      dim2 = width;
   }
   @Override
   public double area(){
      return dim1*dim2;
   }
}
class tri implements Shape{
double dim1,dim2;

 public tri(double height ,double base){
      dim1=height;
      dim2=base;
 }
 @Override
 public double area(){
    return 0.5*dim1*dim2;
 }
}
class Interfacearea{
   public static void main(String[] args){
      Scanner scan = new Scanner (System.in);

     System.out.println("Enter dimensions for Rectangle:");
      System.out.print("Length: ");
      double length = scan.nextDouble();
      System.out.print("Width: ");
      double width = scan.nextDouble();
      rect rectangle = new rect(length, width);

      System.out.println("\nEnter dimensions for Triangle:");
      System.out.print("Base: ");
      double base = scan.nextDouble();
      System.out.print("Height: ");
      double height = scan.nextDouble();
      tri triangle = new tri(height, base);

      System.out.println("\nArea of Rectangle: " + rectangle.area());
      System.out.println("Area of Triangle: " + triangle.area());

      scan.close();
   }
```

```
vishal@vishal-LOQ-15IRX9:~/vishal/java/Ass4$ javac Interfacearea.java
vishal@vishal-LOQ-15IRX9:~/vishal/java/Ass4$ java Interfacearea
Enter dimensions for Rectangle:
Length: 10
Width:
10

Enter dimensions for Triangle:
Base: 10
Height: 20

Area of Rectangle: 100.0
Area of Triangle: 100.0
vishal@vishal-LOQ-15IRX9:~/vishal/java/Ass4$
```

3. Student Exam Results Using Inheritance and Interface in: Implement the following hierarchy: a. Create a class Student with a variable rollNo and methods getRollNo() and setRollNo(). b. Create a class Test that inherits Student and has variables sub1 and sub2 with methods getMarks() and setMarks(). c. Create an interface Sports with a variable sMarks and a method set(). d. Create a class Result that inherits Test and implements the Sports interface. It should display the marks. e. Demonstrate the functionality of these classes in a test application

```
import java.util.*;

interface Stack{

public void push();

        public void pop();

        public void display();

        public boolean isEmpty();

        public boolean isFull();

}
 class IntegerStack implements Stack{
  Scanner scan = new Scanner(System.in);

    public int size;
    public int top ;
    public int[] stack ;


    public IntegerStack(int size) {
       this.size = size;
       this.top = -1;
       this.stack = new int[size];
    }
    @Override
  public void push() {
       if (top == size - 1) {
          System.out.println("Stack is full.");
       } else {
          System.out.print("Enter the element to be pushed in stack: ");
          int element = scan.nextInt();
          top++;
          stack[top] = element;
          System.out.println(element + " is pushed into the stack.");
       }
    }
    @Override
    public void pop() {
       if (top == -1) {
          System.out.println("Stack is empty. Cannot pop.");
       } else {
          int ele = stack[top];
          System.out.println(ele + " is removed from the stack.");
          top--;
       }
    }
    @Override
    public void display() {
       if (top == -1) {
          System.out.println("Stack is empty. No elements to display.");
       } else {
          System.out.println("Stack elements (top to bottom):");
          for (int i = top; i >= 0; i--) {
             System.out.println(stack[i] + " ");
```

```
            }
            System.out.println();
        }
    }
    @Override
     public boolean isFull() {
        return top == size - 1;
    }
    @Override
     public boolean isEmpty() {
        return top == -1;
    }

}

 class Interface {
public static void main(String[] args){
 Scanner scan = new Scanner(System.in);

 System.out.print(" enter the size of the stack: ");
  int stacksize =scan.nextInt();

  IntegerStack ob1 = new IntegerStack(stacksize);
  int choice;

        do {
            System.out.println("\nStack Operations:");
            System.out.println("1. Push");
            System.out.println("2. Pop");
            System.out.println("3. Display");
            System.out.println("4. Check if Stack is Empty");
            System.out.println("5. Check if Stack is Full");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            choice = scan.nextInt();

            switch (choice) {
                case 1:
                    ob1.push();
                    break;
                case 2:
                    ob1.pop();
                    break;
                case 3:
                    ob1.display();
                    break;
                case 4:
                    System.out.println("Is stack empty? " + ob1.isEmpty());
                    break;
                case 5:
                    System.out.println("Is stack full? " + ob1.isFull());
                    break;
                case 6:
                    System.out.println("Exiting the program.");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 6);

        scan.close();

}}
```

```
Enter the element to be pushed in stack: 10
10 is pushed into the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Check if Stack is Empty
5. Check if Stack is Full
6. Exit
Enter your choice: 3
Stack elements (top to bottom):
10
30
20


Stack Operations:
1. Push
2. Pop
3. Display
4. Check if Stack is Empty
5. Check if Stack is Full
6. Exit
Enter your choice: 4
Is stack empty? false

Stack Operations:
1. Push
2. Pop
3. Display
4. Check if Stack is Empty
5. Check if Stack is Full
6. Exit
Enter your choice: 5
Is stack full? false
```