Experiment No. 4

Title: Program based on Interface

Batch:SY-IT(B3) **Roll No.: 16010423076** **Experiment No.:4**

Aim: Create a class student having method get_rollnumber() and put_rollnumber() to accept and display roll number respectively. Derive a sub-class test having methods get_marks() and put_marks() to accept and display marks in three subjects respectively. Create an interface having member bonus_marks=5 and put_bonus() method declaration. Derive a class result from test and sports. Class results should have method display which should calculate the sum of marks including bonus_marks and display all the details of student. Also write method put_bonus() to display the bonus marks awarded.

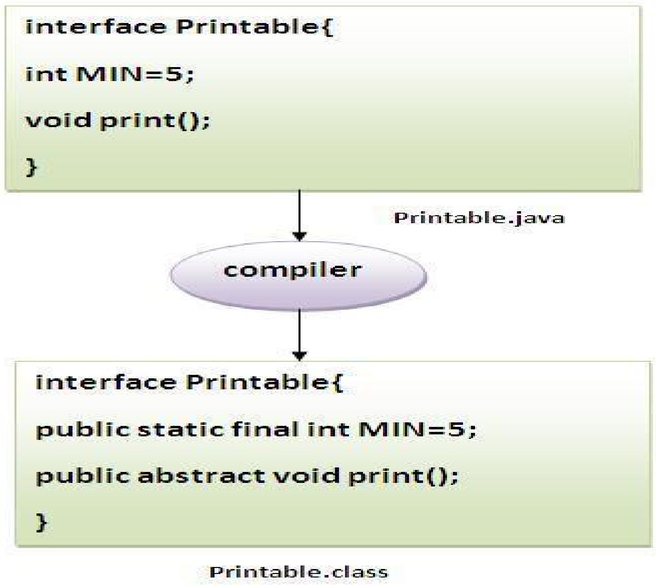
Resources needed: Java

Theory:

An interface in java is a blueprint of a class. It has static constants and abstract methods only. The interface in java is a mechanism to achieve fully abstraction. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java. Java Interface also represents IS-A relationship. It cannot be instantiated just like abstract class. Java Interface also represents IS-A relationship. It cannot be instantiated just like abstract class.

Why use Java interface?
There are mainly three reasons to use interface. They are given below.
i) It is used to achieve fully abstraction.
ii) By interface, we can support the functionality of multiple inheritance.
iii) It can be used to achieve loose coupling.

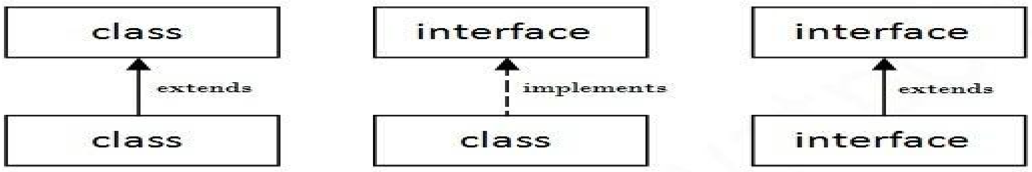
NOTE: The java compiler adds public and abstract keywords before the interface method and public, static and final keywords before data members. In other words, Interface fields are public, static and final bydefault, and methods are public and abstract.



```
interface Printable{
int MIN=5;
void print();
}
```

```
interface Printable{
public static final int MIN=5;
public abstract void print();
}
```

As shown in the figure given below, a class extends another class, an interface extends another interface but a **class implements an interface**.



Simple example of Java interface

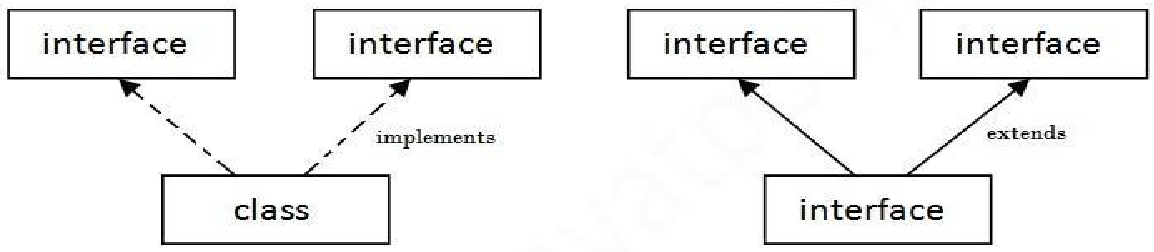
In this example, Printable interface have only one method, its implementation is provided in the A class.

```
interface printable{
void print();
}

class A6 implements printable{
 public void print(){
 System.out.println("Hello"); }

 public static void main(String args[]){
  A6 obj = new A6();
  obj.print();
  } }
```

Output:Hello

Multiple inheritance in Java by interface

If a class implements multiple interfaces, or an interface extends multiple interfaces i.e. known as multiple inheritance.



Multiple Inheritance in Java

```
interface Printable{
void print();  }

interface Showable{
void show();  }

class A7 implements Printable,Showable{
public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}

public static void main(String args[]){
A7 obj = new A7();
obj.print();
obj.show();  }  }
```

```
Output: Hello
        Welcome
```

Interface inheritance
A class implements interface but one interface extends another interface.

```
interface Printable{
void print();  }

interface Showable extends Printable{
void show();  }

class Testinterface2 implements Showable{
public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}
public static void main(String args[]){
Testinterface2 obj = new Testinterface2();
obj.print();
obj.show();  }  }
```

```
Output: Hello
```

Welcome

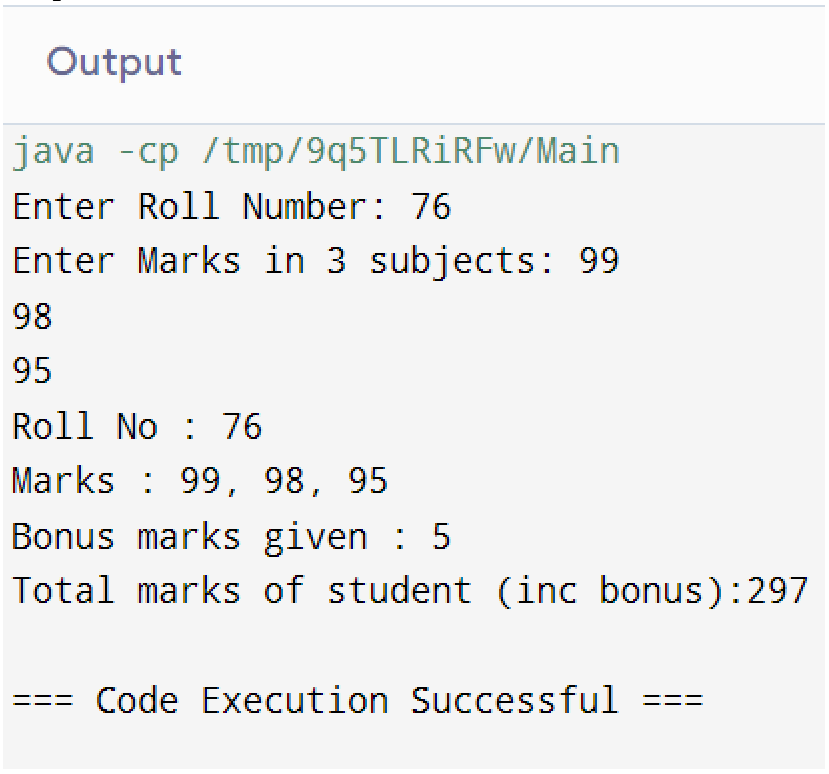
Results: (Program with output)

Code :

```
import java.util.Scanner;

interface Sports{
int bonusmarks = 5;
void put_bonus();
}

class Student {
int rollno;

void get_rollnumber(int roll){
        rollno = roll;
        }

void put_rollnumber(){
        System.out.println("Roll No : "+ rollno);
        }
}

class Test extends Student{
int marks1, marks2, marks3;

void get_marks(int m1,int m2,int m3){
        marks1 = m1;
        marks2 = m2;
        marks3 = m3;
        }

void put_marks(){
        System.out.println("Marks : "+marks1+", "+marks2+", "+marks3);
        }

}

class Result extends Test implements Sports{
int totalmarks;

void display(){
totalmarks = marks1+marks2+marks3+bonusmarks;
put_rollnumber();
put_marks();
put_bonus();
System.out.println("Total marks of student (inc bonus):" +totalmarks);
}

public void put_bonus(){
System.out.println("Bonus marks given : "+bonusmarks);
}

}
```

```
public class Main{
public static void main(String argz[]){
Scanner sc = new Scanner(System.in);
Result student1 = new Result();
System.out.print("Enter Roll Number: ");
int roll = sc.nextInt();
student1.get_rollnumber(roll);
System.out.print("Enter Marks in 3 subjects: ");
int m1 = sc.nextInt();
int m2 = sc.nextInt();
int m3 = sc.nextInt();
student1.get_marks(m1, m2, m3);
student1.display();
}
}
```

Output :



Output

```
java -cp /tmp/9q5TLRiRFw/Main
Enter Roll Number: 76
Enter Marks in 3 subjects: 99
98
95
Roll No : 76
Marks : 99, 98, 95
Bonus marks given : 5
Total marks of student (inc bonus):297

=== Code Execution Successful ===
```

Questions:

1. What is the purpose of ‘extends’ and ‘implements’ keywords?

The extends keyword in Java is used when one class inherits properties and behaviors (methods) from another class.
This means the new class (child class) gets all the features of the existing class (parent class) and can add or modify some of them.

The implements keyword is used when a class wants to use an interface.
An interface is like a contract that says the class must include certain methods.
When a class implements an interface, it must provide concrete implementations for all the methods declared in the interface.

Outcomes:

CO2: Apply String manipulation functions , inheritance and polymorphism using Java

Conclusion: (Conclusion to be based on the outcomes achieved)

From experiment number 4, I mainly learned about the implementation of Interface in Java by implementing a program related to student marks.
Other than interface, I also learned the difference between class and interface, extending interfaces, difference between extends and implements for inheriting properties.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:
Books:

1. Herbert Schildt; JAVA The Complete Reference; Seventh Edition, Tata McGraw- Hill Publishing Company Limited 2007.
2. Java 7 Programming - Black Book : Kogent Learning Solutions Inc.
3. Sachin Malhotra, Saurabh Chaudhary “Programming in Java”, Oxford University Press, 2010
4. Jaime Nino, Frederick A. Hosch, ‘An introduction to Programming and Object Oriented Design using Java’, Wiley Student Edition.