

PART-IV Inheritance, Interface, Package

No	Aim of the Practical
17.	<p><u>AIM :</u> Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call 1 - method of parent class by object of parent</p> <p><u>PROGRAM CODE :</u></p> <pre> public class Prac_17 { void print1() { System.out.println("This is parent class"); } public static void main(String[] args) { Prac_17 obj1 = new Prac_17(); Child obj2 = new Child(); obj1.print1(); obj2.print1(); System.out.print("\nName : OM BARVALIYA \nID : 23DCS009"); } } class Child extends Prac_17 { void print1() { System.out.println("This is child class"); } } </pre> <p><u>OUTPUT:</u></p>  <p style="text-align: center;"><u>OUTPUT: PRACTICAL-17</u></p> <p><u>CONCLUSION:</u></p> <p>This Java program demonstrates method overriding in inheritance. The Prac_17 class has a method print1 which is overridden by the Child class. When the print1 method is called on an instance of Child, it executes the overridden method in the Child class.</p>

	<p>The output shows the different messages from the parent and child classes, illustrating polymorphism.</p>
18.	<p><u>AIM :</u> Create a class named 'Member' having the following members: Data members</p> <ul style="list-style-type: none"> 1 – Name 2 - Age 3 - Phone number 4 - Address 5 – Salary <p>It also has a method named 'printSalary' which prints the salary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.</p> <p><u>PROGRAM CODE :</u></p> <pre> class Member { String name; int age; String phoneNumber; String address; double salary; void printSalary() { System.out.println("Salary: " + salary); } } class Employee extends Member { String specialization; } class Manager extends Member { String department; } public class Prac_18 { public static void main(String[] args) { Employee employee = new Employee(); employee.name = "Name 1"; } } </pre>

```

employee.age = 30;
employee.phoneNumber = "123456789";
employee.address = "India";
employee.salary = 50000;
employee.specialization = "Software Engineering";

Manager manager = new Manager();
manager.name = "name 2 ";
manager.age = 40;
manager.phoneNumber = "987654321";
manager.address = "456 Elm St";
manager.salary = 80000;
manager.department = "IT";

System.out.println("Employee Details:");
System.out.println("Name: " + employee.name);
System.out.println("Age: " + employee.age);
System.out.println("Phone Number: " + employee.phoneNumber);
System.out.println("Address: " + employee.address);
employee.printSalary();
System.out.println("Specialization: " + employee.specialization);

System.out.println("\nManager Details:");
System.out.println("Name: " + manager.name);
System.out.println("Age: " + manager.age);
System.out.println("Phone Number: " + manager.phoneNumber);
System.out.println("Address: " + manager.address);
manager.printSalary();
System.out.println("Department: " + manager.department);

System.out.print("\nName : OM BARVALIYA \nID : 23DCS009");
}
}

```

OUTPUT:

```

PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\Part4\" ;
Employee Details:
Name: Name 1
Age: 30
Phone Number: 123456789
Address: India
Salary: 50000.0
Specialization: Software Engineering

Manager Details:
Name: name 2
Age: 40
Phone Number: 987654321
Address: 456 Elm St
Salary: 80000.0
Department: IT

Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\Part4>

```

OUTPUT: PRACTICAL-18

	<p><u>CONCLUSION:</u></p> <p>This Java program demonstrates inheritance by defining a base class Member and two derived classes Employee and Manager. The Member class includes common attributes and a method to print the salary. The Employee and Manager classes inherit these attributes and add their own specific attributes. The main method creates an instance of Employee and sets its attributes, showcasing the use of inheritance.</p>
19.	<p><u>AIM :</u> Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used to initialize length and breadth of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructor having a parameter for its side (suppose s) calling the constructor of its parent class as 'super(s,s)'. Print the area and perimeter of a rectangle and a square. Also use array of objects.</p> <p><u>PROGRAM CODE :</u></p> <pre> public class Prac_19 { public static void main(String[] args) { Rectangle rect = new Rectangle(10, 5); Square sq = new Square(7); Rectangle[] shapes = new Rectangle[2]; shapes[0] = rect; shapes[1] = sq; for (Rectangle shape : shapes) { shape.printArea(); shape.printPerimeter(); System.out.println(); } System.out.print("\nName : OM BARVALIYA \nID : 23DCS009"); } } class Rectangle { protected int length; protected int breadth; public Rectangle(int length, int breadth) { this.length = length; } </pre>

	<pre> this.breadth = breadth; } public void printArea() { int area = length * breadth; System.out.println("Area: " + area); } public void printPerimeter() { int perimeter = 2 * (length + breadth); System.out.println("Perimeter: " + perimeter); } } class Square extends Rectangle { public Square(int side) { super(side, side); } } </pre> <p><u>OUTPUT:</u></p>  <p style="text-align: center;"><u>OUTPUT: PRACTICAL-19</u></p> <p><u>CONCLUSION:</u></p> <p>This Java program demonstrates polymorphism using inheritance. The Rectangle class is extended by the Square class, allowing both to be treated as Rectangle objects. An array of Rectangle objects is used to store both Rectangle and Square instances. The program iterates through the array, calling methods to print the area and perimeter of each shape.</p>
20.	<p><u>AIM :</u> Create a class named 'Shape' with a method to print "This is This is shape". Then create two other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to print "This is rectangular shape" and "This is circular shape" respectively. Create a subclass 'Square' of 'Rectangle' having a method to print "Square</p>

is a rectangle". Now call the method of 'Shape' and 'Rectangle' class by the object of 'Square' class.

PROGRAM CODE :

```
class Shape{
    public void printS()
    {
        System.out.println("This is This shape");
    }
}
class Rectangle extends Shape{
    public void printR()
    {
        System.out.println("This is Rectangle shape");
    }
}
class Circle extends Shape{
    public void printC()
    {
        System.out.println("This is Circle shape");
    }
}
class Square extends Rectangle{
    public void printSq()
    {
        printS();
        printR();
        System.out.println("Square is Reactangle ");
    }
}
public class Prac_20{
    public static void main(String[] args)
    {
        Shape s = new Shape();
        Square A =new Square();
        A.printSq();
        System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 ");
    }
}
```

OUTPUT:

```
PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\Part4\"
This is This shape
This is Rectangle shape
Square is Reactangle

Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\Part4>
```

OUTPUT: PRACTICAL-20

CONCLUSION:

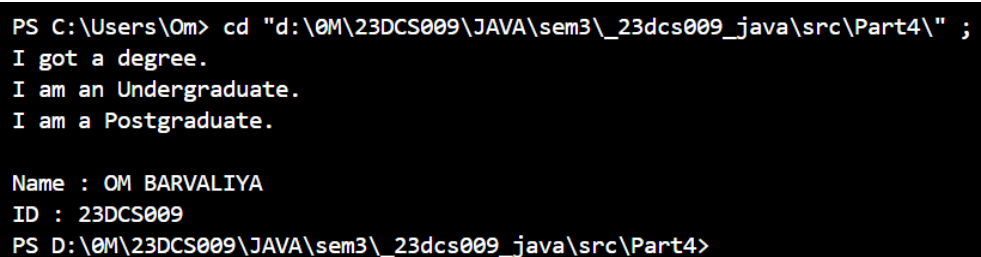
This Java program demonstrates inheritance and method overriding. The Shape class is extended by Rectangle and Circle, each adding their own print methods. The Square class extends Rectangle and overrides its methods, showcasing multi-level inheritance. The program illustrates how derived classes can call methods from their parent classes.

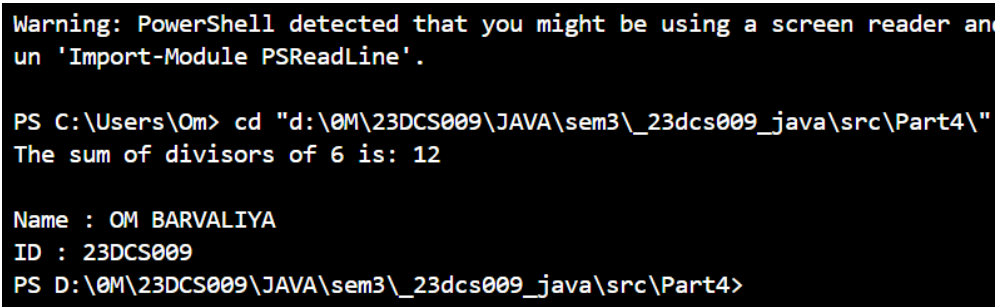
21. **AIM:** Create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.

PROGRAM CODE :

```
class Degree{
    public void getDegree()
    {
        System.out.println("I got a degree.");
    }
}
class Undergraduate extends Degree {
    public void getDegree()
    {
        System.out.println("I am an Undergraduate.");
    }
}
class Postgraduate extends Degree{
    public void getDegree()
    {
        System.out.println("I am a Postgraduate.");
    }
}

public class Prac_21 {
```

	<pre> public static void main(String[] args) { Degree d = new Degree(); Undergraduate u = new Undergraduate(); Postgraduate p = new Postgraduate(); d.getDegree(); u.getDegree(); p.getDegree(); System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 "); } </pre> <p><u>OUTPUT:</u></p>  <p><u>OUTPUT: PRACTICAL-21</u></p> <p><u>CONCLUSION:</u></p> <p>This Java program demonstrates method overriding in inheritance. The Degree class has a method getDegree which is overridden by Undergraduate and Postgraduate classes. Each subclass provides its own implementation of the getDegree method. The main method creates instances of each class and calls their respective getDegree methods, showcasing polymorphism.</p>
22.	<p><u>AIM:</u> Write a java that implements an interface AdvancedArithmetic which contains amethod signature int divisor_sum(int n). You need to write a class calledMyCalculator which implements the interface. divisorSum function just takes an integer as input and return the sum of all its divisors. For example, divisors of 6 are 1, 2, 3 and 6, so divisor_sum should return 12. The value of n will be at most 1000.</p> <p><u>PROGRAM CODE :</u></p> <pre> interface AdvancedArithmetic { int divisor_sum(int n); } class MyCalculator implements AdvancedArithmetic { public int divisor_sum(int n) { int sum = 0; for (int i = 1; i <= n; i++) { </pre>

	<pre> if (n % i == 0) { sum += i; } } return sum; } } public class Prac_22 { public static void main(String[] args) { MyCalculator myCalculator = new MyCalculator(); int n = 6; int result = myCalculator.divisor_sum(n); System.out.println("The sum of divisors of " + n + " is: " + result); System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 "); } } </pre> <p><u>OUTPUT:</u></p>  <p style="text-align: center;"><u>OUTPUT: PRACTICAL-22</u></p> <p><u>CONCLUSION:</u></p> <p>This Java program demonstrates the implementation of an interface. The AdvancedArithmetic interface is implemented by the MyCalculator class, which provides the divisor_sum method. The main method creates an instance of MyCalculator and calculates the sum of divisors for a given number. The result is printed, showcasing the functionality of the implemented method.</p>
23.	<p><u>AIM :</u> Assume you want to capture shapes, which can be either circles (with a radius and a color) or rectangles (with a length, width, and color). You also want to be able to create signs (to post in the campus center, for example), each of which has a shape (for the background of the sign) and the text (a String) to put on the sign. Create classes and interfaces for circles, rectangles, shapes, and signs. Write a program that illustrates the significance of interface default method.</p>

PROGRAM CODE :

```

interface Shape{
    void print();
}
class Circle implements Shape{
    int radius;
    String color;
    Circle(int radius, String color){
        this.radius = radius;
        this.color = color;
    }
    public void print(){
        System.out.println("Radius : "+radius+" Color : "+color);
    }
}
class Rectangle implements Shape{
    int length;
    int width;
    String color;
    Rectangle(int length, int width, String color){
        this.length = length;
        this.width = width;
        this.color = color;
    }
    public void print(){
        System.out.println("Length : "+length+" Width : "+width+" Color : "+color);
    }
}
class Sign{
    Shape s;
    String text;
    Sign(Shape s, String text){
        this.s = s;
        this.text = text;
    }
    void print(){
        s.print();
        System.out.println("Text : "+text);
    }
}
public class Prac_23{
    public static void main(String[] args) {
        Circle c = new Circle(10, "Red");
        Rectangle r = new Rectangle(10, 20, "Blue");
        Sign s = new Sign(c, "Circle Sign");
        s.print();
        Sign s1 = new Sign(r, "Rectangle Sign");
    }
}

```

```
s1.print();
System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 ");

}
}
```

OUTPUT:

```
PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\Part4\" ;
Radius : 10 Color : Red
Text : Circle Sign
Length : 10 Width : 20 Color : Blue
Text : Rectangle Sign

Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\Part4>
```

OUTPUT: PRACTICAL-23

CONCLUSION:

This Java program demonstrates the use of interfaces and polymorphism. The Shape interface is implemented by the Circle and Rectangle classes, each providing their own print method. Instances of Circle and Rectangle can be treated as Shape objects. This allows for flexible and interchangeable use of different shapes in the program.