

CHAROTAR UNIVERSITY OF SCIENCE TECHNOLOGY
DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY &
RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3rd

Subject Code: CSE201

Academic year: 2024

SET- 4

N o	Aim of the Practical
17.	<p>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> class parents { void display() { System.out.println("this is parents class. "); } } class child extends parents { void show() { System.out.println("this is child class."); } } class inheritance { public static void main(String[] args) { parents p1=new parents(); p1.display(); child c1=new child(); c1.show(); System.out.println("\n23DCS27_Astha Sodvadia"); } } </pre>

OUTPUT:

```
this is parents class.  
this is child class.  
  
23DCS27_Astha Sodvadia
```

CONCLUSION:

This demonstrates the basic concept of inheritance in Java, where a subclass can inherit methods from its parent class, but still, you can create and use the parent class independently.

- 18** Create a class named 'Member' having the following members: Data members
- 1 - Name
 - 2 - Age
 - 3 - Phone number
 - 4 - Address
 - 5 – Salary
- 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.

PROGRAM CODE :

```
import java.util.*;

class mem
{
    String name;
    String add;
    int age;
    double pnum;
    float salary;

    void getdata()
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter your name : ");
        name = sc.nextLine();
        System.out.print("Enter your age : ");
        age = sc.nextInt();
        System.out.print("Enter your phone number : ");
        pnum = sc.nextDouble();
        System.out.print("Enter your address : ");
        add = sc.nextLine();
        sc.nextLine();
        System.out.print("Enter your salary : ");
        salary = sc.nextFloat();
    }
}
```

```
void display()
{
System.out.println("Name is: " + name);
System.out.println("Age is : "+age);
System.out.println("Phone number is : "+pnum);
System.out.println("Address is : "+add);
System.out.println("Salary is : "+salary);
}
}

class emloyee extends mem
{
String sp="";
void getdata()
{
super.getdata();
Scanner sc = new Scanner(System.in);
System.out.print("Your specialization : ");
sp = sc.nextLine();
}
void display()
{
super.display();
System.out.print("Your specialization : "+sp);
}
}

class manager extends mem
{
String dp="";
void getdata()
{
super.getdata();
Scanner sc = new Scanner(System.in);
System.out.print("Your department : ");
dp=sc.nextLine();
}
void display()
{
super.display();
System.out.print("Your department : "+dp);
}
}

class member
```

```
{
public static void main(String[] args)
{

employee e1 = new employee();
manager m1=new manager();

System.out.println("Enter employee details:");
e1.getdata();
//e1.display();

System.out.println("\n\n\nEnter manager details:");
m1.getdata();
//m1.display();
System.out.println("\n23DCS27_Astha Sodvadia");
}
}
```

OUTPUT:

```
Enter employee details:  
Enter your name : astha  
Enter your age : 19  
Enter your phone number : 5456885456  
Enter your address : surat  
Enter your salary : 200000  
Your specialization : handling
```

```
Enter manager details:  
Enter your name : manasvi  
Enter your age : 18  
Enter your phone number : 456123895  
Enter your address : junagadh  
Enter your salary : 250000  
Your department : manager
```

```
23DCS27_Astha Sodvadia
```

CONCLUSION:

By creating objects of the Employee and Manager classes, we can assign values to their attributes and print them. This demonstrates inheritance in Java, where subclasses can use properties and methods from a parent class while also having their own specific properties.

- 19** 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 parents class as 'super(s,s)'. print the area and Perimeter of a rectangle and a square. Also use array of objects.

PROGRAM CODE :

```
import java.util.Scanner;

class rectangle
{
    Scanner sc = new Scanner(System.in);

    double l,b;
    double area,pmeter;
    rectangle()
    {
        l=0.0;
        b=0.0;
        area=0.0;
        pmeter=0.0;
    }

    void getdata()
    {
        System.out.println("\nRECTANGLE");
        System.out.print("Enter length : ");
        l=sc.nextDouble();
        System.out.print("Enter breadth : ");
        b=sc.nextDouble();
        area = l*b;
        System.out.println("Area rectangle is : "+area);
        pmeter=2*(l+b);
        System.out.println("Perimeter rectangle is : "+pmeter);
    }

    void display()
    {
```



```
System.out.println("length : "+l);
System.out.println("breadth : "+b);
System.out.println("Area : "+area);
System.out.println("Perimeter : "+pmeter);
}
}

class square extends rectangle
{
square()
{
super();
}
Scanner sc = new Scanner(System.in);

void getdata()
{
System.out.println("\n\nSQUARE");
System.out.print("Enter length : ");
l=sc.nextDouble();
area = l*l;
pmeter=4*l;
System.out.println("Your Area is : "+area);
System.out.println("Your Perimeter is : "+pmeter);
}

void display()
{
System.out.println("length : "+l);
System.out.println("Area : "+area);
System.out.println("Perimeter : "+pmeter);
}
}

class perimeter
{
public static void main(String[] args)
{
rectangle r1=new rectangle();
r1.getdata();
r1.display();

square s1=new square();
s1.getdata();
s1.display();
}
```

```
System.out.println("\n23DCS27_Astha Sodvadia");  
}  
}
```

OUTPUT:

```
RECTANGLE  
Enter length : 2  
Enter breadth : 7  
Area rectangle is : 14.0  
Perimeter rectangle is : 18.0  
length : 2.0  
breadth : 7.0  
Area : 14.0  
Perimeter :18.0  
  
SQUARE  
Enter length : 4  
Your Area is : 16.0  
Your Perimeter is : 16.0  
length : 4.0  
Area : 16.0  
Perimeter : 16.0  
  
23DCS27_Astha Sodvadia
```

CONCLUSION:

This example highlights the concepts of inheritance, constructor chaining, and polymorphism in Java. By using an array of objects, we can manage different shapes uniformly and utilize the shared methods from the parent class (`Rectangle`) in both the rectangle and square instances.

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

PROGRAM CODE :

```
class shape
{
    void printdata()
    {
        System.out.println("This is shape.");
    }
}

class rectangle extends shape
{
    void printdata()
    {
        System.out.println("This is rectangular shape.");
    }
}

class circle extends shape
{
    void printdata()
    {
        System.out.println("This is circular shape.");
    }
}

class square extends rectangle
{
    void printdata()
    {
        System.out.println("Square is a rectangle.");
    }
}
```

```
}

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

shape s1=new shape();
s1.printdata();

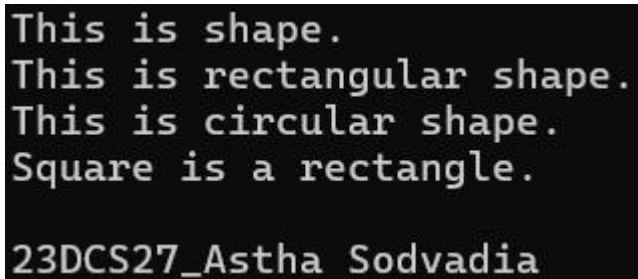
rectangle r1=new rectangle();
r1.printdata();

circle c1=new circle();
c1.printdata();

square sq=new square();
sq.printdata();

System.out.println("\n23DCS27_Astha Sodvadia");
}
}
```

OUTPUT:

A screenshot of a terminal window with a black background and white text. The output consists of five lines: "This is shape.", "This is rectangular shape.", "This is circular shape.", "Square is a rectangle.", and "23DCS27_Astha Sodvadia".

```
This is shape.
This is rectangular shape.
This is circular shape.
Square is a rectangle.

23DCS27_Astha Sodvadia
```

CONCLUSION:

By creating an object of the Square class, we can call methods from both the Shape class and the Rectangle class because Square inherits from Rectangle, which in turn inherits from Shape. This demonstrates the concept of inheritance in Java.

- 21** 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 Degrees
{
void printdata()
{
System.out.println("I got a degree.");
}
}

class undergraduate extends Degrees
{
void printdata()
{
System.out.println("I am an Undergraduate.");
}
}

class postgraduate extends Degrees
{
void printdata()
{
System.out.println("I am a Postgraduate.");
}
}

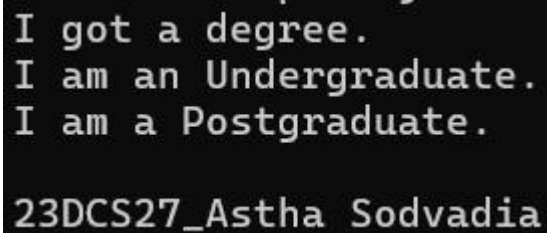
public class degree
{
public static void main(String[] args)
{
Degrees d1=new Degrees();
d1.printdata();

undergraduate u1 = new undergraduate();
u1.printdata();

postgraduate p1 =new postgraduate();
```

```
p1.printdata();

System.out.println("\n23DCS27_Astha Sodvadia");
}
}
```

OUTPUT:A screenshot of a terminal window showing the output of a Java program. The text is displayed in a monospaced font on a dark background. The output consists of four lines: "I got a degree.", "I am an Undergraduate.", "I am a Postgraduate.", and "23DCS27_Astha Sodvadia".

```
I got a degree.
I am an Undergraduate.
I am a Postgraduate.

23DCS27_Astha Sodvadia
```

CONCLUSION:

This code demonstrates method overriding in Java, where the Undergraduate and Postgraduate classes override the `getDegree()` method of the Degree class to provide specific outputs. Each class prints a message related to the type of degree it represents. This showcases how subclasses can customize inherited behavior by overriding methods from a parent class.

- 22** Write a java that implements an interface AdvancedArithmetic which contains a method signature `int divisor_sum(int n)`. You need to write a class called `MyCalculator` 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.

PROGRAM CODE :

```
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++) {
            if (n % i == 0) {
                sum += i;
            }
        }
        return sum;
    }
}

public class prac22 {
    public static void main(String[] args) {
        MyCalculator my_calculator = new MyCalculator();
        System.out.print("I implemented: ");
        int n = 6;
        System.out.println(my_calculator.divisor_sum(n));
    }
}
```

OUTPUT:

```
I implemented: 12
```

CONCLUSION:

This code demonstrates the implementation of an interface in Java. The AdvanceArithmetic interface defines a method divisor_sum(int n), which is implemented by the MyCalculator class. The program calculates and returns the sum of all divisors of a given number, showcasing how interfaces can be used to define a contract for classes to implement specific functions.

- 23** 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.

PROGRAM CODE :

```
import java.util.*;

interface Shape {
    double area();

    default void displayDetails() {
        System.out.println("This is a shape.");
    }
}

class Circle implements Shape {
    private double radius;
    private String color;

    public Circle(double radius, String color) {
        this.radius = radius;
        this.color = color;
    }

    public double area() {
        return Math.PI * radius * radius;
    }

    public void displayDetails() {
        System.out.println("This is a circle with radius " + radius + " and color "
+ color);
    }
}
```

```
class Rectangle implements Shape {
    private double length;
    private double width;
    private String color;

    public Rectangle(double length, double width, String color) {
        this.length = length;
        this.width = width;
        this.color = color;
    }

    public double area() {
        return length * width;
    }

    public void displayDetails() {
        System.out.println("This is a rectangle with length " + length + ", width "
+ width + ", and color " + color);
    }
}

class Sign {
    private Shape shape;
    private String text;

    public Sign(Shape shape, String text) {
        this.shape = shape;
        this.text = text;
    }

    public void displaySign() {
        shape.displayDetails();
        System.out.println("Text: " + text);
    }
}

public class prac23 {
    public static void main(String[] args) {
```

```
Circle circle = new Circle(5.0, "Red");
Rectangle rectangle = new Rectangle(4.0, 6.0, "Blue");

Sign circleSign = new Sign(circle, "Hello, World!");
Sign rectangleSign = new Sign(rectangle, "Welcome to the Campus
Center!");

circleSign.displaySign();
rectangleSign.displaySign();
}
```

OUTPUT:

```
This is a circle with radius 5.0 and color Red
Text: Hello, World!
This is a rectangle with length 4.0, width 6.0, and color Blue
Text: Welcome to the Campus Center!
```

CONCLUSION:

This code demonstrates the use of interfaces, composition, and default methods in Java. The shape interface defines the methods color() and area(), and a default method info(), which are implemented by the Circle and Rectangle classes. The sign class uses composition to associate a shape with text, and its display() method prints both the sign's text and shape information.