Java Programming [CSE201] Enrolment No.: 23DCS056

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

**DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH**

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester: 3

Subject Code: 203

Academic year: 2024-2025

Part - 4

|  |  |
| --- | --- |
| **No.** | **Aim of the Practical** |
| 17. | 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  **PROGRAM CODE:**  class parent {      public void displayparent() {          System.out.println("this is parent class");      }  }  class child extends parent {      public void displaychild() {          System.out.println("this is child class");  **}**  }  public class Practical17 {      public static void main(String[] args) {          parent p1=new parent();          child c1= new child();          System.out.println("calling parent class method");          p1.displayparent();          System.out.println("calling inherited method from child class");          c1.displayparent();          System.out.println("calling child class method");          c1.displaychild();      }  }  **OUTPUT:**    **CONCLUSION:**  By performing this experiment we learnt how to use the concept of inheritance in java language. It uses the extends keyword to inherit a base class into a derived class, and the object of the derived class can access all the methods and data of the base class provided that they are inherited publicly or protected. |
| 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 (Times New Roman:14) :**  import java.util.\*;  class Member {      Scanner sc = new Scanner(System.in);      int age;      int phone;      String name;      String address;      int salary;      public void printSalary() {          System.out.println("The salary of the member is" + this.salary);      }      public void getdata() {          System.out.println("enter the age of member");          this.age=sc.nextInt();          System.out.println("enter the Name of member");          this.name=sc.next();          System.out.println("enter the phone number of member");          this.phone=sc.nextInt();          System.out.println("enter the address of member");          this.address=sc.next();          System.out.println("enter the salary of member");          this.salary=sc.nextInt();      }      public void printdata() {          System.out.println("The name of the member is " + this.name);          System.out.println("The age of the member is " + this.age);          System.out.println("The phone number of the member is " + this.phone);          System.out.println("The address of the member is " + this.address);          System.out.println("The salary of the member is "+this.salary);      }  }  class employee extends Member {      String specialization;  }  class manager extends Member {      // String specialization;      String department;  }    public class Practical18 {      public static void main(String[] args) {          employee e1 = new employee();          manager m1 = new manager();          e1.getdata();          e1.printdata();          m1.getdata();          m1.printdata();      }  }    **OUTPUT:**    **CONCLUSION:**  By performing this experiment we were able to create constructors and methods to input data for data members of derived class and base class. And further we can extend by creating a method which can display the data. |
| 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 parent class as 'super(s,s)'. Print the area and perimeter of a rectangle and a square. Also use array of objects.  **PROGRAM:**  class Rectangle {      int length, breadth;      public Rectangle(int l, int b) {          this.length = l;          this.breadth = b;      }      public void area() {          int area = this.length \* this.breadth;          System.out.println("the area is " + area);      }      public void perimeter() {          int perimeter = 2 \* (this.length + this.breadth);          System.out.println("the perimeter is " + perimeter);      }  }  class Square extends Rectangle {      public Square(int x) {          super(x, x);      }  }    public class Practical19 {      public static void main(String[] args) {          Square[] s1 = new Square[2];          s1[0] = new Square(10);            s1[1] = new Square(5);          s1[0].area();          s1[0].perimeter();          s1[1].area();          s1[1].perimeter();      }    }  **OUTPUT:**    **CONCLUSION:**  By this experiment we learned how constructors work in inheritance and how they are called. They are called by using the super keyword and passing the parameter given to the constructor of the derived class to the constructor of the base class. |
| 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:**  class Shape {      public void displayshape(){          System.out.println("This is Shape");      }  }  class Rectangle extends Shape {      public void displayRectangle() {          System.out.println("this is Rectangular shape");      }    }  class Circle extends Shape {      public void displaycircle() {          System.out.println("this is circular shape");      }  }  class Square extends Rectangle {      public void displaysquare() {          System.out.println("Square is rectangle");      }  }    public class Practical20 {      public static void main(String[] args) {          Square s1 = new Square();          s1.displayRectangle();          s1.displayshape();      }  }  **OUTPUT:**    **CONCLUSION:**  By performing this program we learnt how to perform hierarchal 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:**  class Degree {      public void getDegree() {          System.out.println("i got a degree");      }  }  class Undergraduate extends Degree {      public void getDegree() {          System.out.println("i am undergraduate");      }  }  class Postgraduate extends Degree {      public void getDegree() {          System.out.println("i am postgraduate");      }  }  public class Practical21 {      public static void main(String[] args) {          Degree d1 = new Degree();          Undergraduate u1 = new Undergraduate();          Postgraduate p1 = new Postgraduate();          d1.getDegree();          u1.getDegree();          p1.getDegree();        }  }  **OUTPUT:**    **CONCLUSION:**  By performing this practical in java we learnt how to create subclasses in java and call their methods. |
| 22 | Write a java that implements an interface Advanced Arithmetic which contains a method signature int divisor sum(int n). You need to write a class called My Calculator which implements the interface. Divisor Sum 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:**  import java.util.Scanner;  public class Practical22 {      public static void main(String[] args) {          MyCalculator mc = new MyCalculator();          Scanner sc = new Scanner(System.in);          int n;          n = sc.nextInt();          System.out.println("the sum of divisors of " + n + "is " + mc.divisor\_sum(n));          sc.close();      }  }  interface AdvancedArithmetic {      int divisor\_sum(int n);  }  class MyCalculator implements AdvancedArithmetic {          @Override      public int divisor\_sum(int n) {          int sum=0;          for (int i = 1; i < n; i++) {              if (n % i == 0) {                  sum += i;              }          }            return sum;        }  }  **Output:**    **Conclusion:**  By performing this practical we learnt how to implement interfaces and perform arithmetic calculations |
| 23 | Assume you want to capture shapes, which can be either circles (with a radiusand 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:**  interface Shape {  String getColor();  void draw();      default void describe() {  System.out.println("This is a shape with color " + getColor());  }  }      class Circle implements Shape {  private double radius;  private String color;    public Circle(double radius, String color) {  this.radius = radius;  this.color = color;  }    public double getRadius() {  return radius;  }    public void setRadius(double radius) {  this.radius = radius;  }      public String getColor() {  return color;  }      public void draw() {  System.out.println("Drawing a circle with radius " + radius + " and color " + color);  }      public void describe() {  System.out.println("This is a shape with color " + getColor());  }  }      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 getLength() {  return length;  }    public void setLength(double length) {  this.length = length;  }    public double getWidth() {  return width;  }    public void setWidth(double width) {  this.width = width;  }  public String getColor() {  return color;  }    public void draw() {  System.out.println("Drawing a rectangle with length " + length + ", width " + width +  " and color " + color);  }      public void describe() {  System.out.println("This is a shape with color " + getColor());  }  }    // Implement Sign class  class Sign {  private Shape shape;  private String text;    public Sign(Shape shape, String text) {  this.shape = shape;  this.text = text;  }    public Shape getShape() {  return shape;  }    public void setShape(Shape shape) {  this.shape = shape;  }    public String getText() {  return text;  }    public void setText(String text) {  this.text = text;  }    public void display() {  System.out.println("Sign Text: " + text);  shape.draw();  shape.describe();  }  }    public class practical23 {  public static void main(String[] args) {  Circle circle = new Circle(5.0, "Red");  Rectangle rectangle = new Rectangle(10.0, 4.0, "Blue");    Sign circleSign = new Sign(circle, "Welcome to the Event!");  Sign rectangleSign = new Sign(rectangle, "Campus Center Open!");    circleSign.display();  System.out.println();  rectangleSign.display();  }  }  **Output:**    **Conclusion:**  The code demonstrates polymorphism by using the Shape interface to handle different types of shapes (Circle and Rectangle) in a uniform way. The Sign class is designed to work with any object that implements the Shape interface. |