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:2024-25

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 theclass and call 1 - method of parent class by object of parent  **PROGRAM CODE:**      class Parent {          void displayParent() {              System.out.println("This is parent class");          }      }        class Child extends Parent {          void displayChild() {              System.out.println("This is child class");          }      }        public class pra17 {          public static void main(String[] args) {              Parent parentObj = new Parent();              parentObj.displayParent();                Child childObj = new Child();              childObj.displayChild();          }      }  **OUTPUT:**    **CONCLUSION:**  In this code snippet, you’ve demonstrated basic inheritance in Java. The Child class inherits the behavior (i.e., the displayParent() method) from the Parent class. When you create objects of these classes, you can call their specific methods. Remember that inheritance allows you to reuse code and establish relationships between classes. |
| 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 thesame.  **PROGRAM CODE:**  class Member {      String name;      int age;      String phNumber;      String adrs;      double salary;      void printSalary() {          System.out.println("Salary: " + salary);      }  }  class Employee extends Member {      String specialization;      String department;  }  class Manager extends Member {      String specialization;      String department;  }  public class pra18 {      public static void main(String[] args) {            Employee employee = new Employee();          employee.name = "prins";          employee.age = 18;          employee.phNumber = "0123456789";          employee.adrs = "gandhi chouk , junagadh";          employee.salary = 50000;          employee.specialization = "Software Engineering";          Manager manager = new Manager();          manager.name = "tirth";          manager.age = 19;          manager.phNumber = "1234567890";          manager.adrs = "mota varacha , surat";          manager.salary = 75000;          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.phNumber);          System.out.println("Address: " + employee.adrs);          employee.printSalary();          System.out.println("Specialization: " + employee.specialization);          System.out.println("Manager Details:");          System.out.println("Name: " + manager.name);          System.out.println("Age: " + manager.age);          System.out.println("Phone Number: " + manager.phNumber);          System.out.println("Address: " + manager.adrs);          manager.printSalary();          System.out.println("Department: " + manager.department);      }  }  **OUTPUT:**    **CONCLUSION:**  This code demonstrates inheritance in Java. Both Employee and Manager classes inherit common properties (such as name, age, and salary) from the Member class. By extending the Member class, you’ve avoided code duplication and established a relationship between these classes. |

|  |  |
| --- | --- |
| 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 CODE:**  class Rectangle {      int length;      int breadth;      Rectangle(int length, int breadth) {          this.length = length;          this.breadth = breadth;      }      void Area() {          int area = length \* breadth;          System.out.println("Area: " + area);      }      void Perimeter() {          int perimeter = 2 \* (length + breadth);          System.out.println("Perimeter: " + perimeter);      }  }  class Square extends Rectangle {      Square(int side) {          super(side, side);      }  }  public class pra19 {      public static void main(String[] args) {          Rectangle rect = new Rectangle(10, 5);          System.out.println("Rectangle:");          rect.Area();          rect.Perimeter();          Square squ = new Square(7);          System.out.println("Square:");          squ.Area();          squ.Perimeter();  }  }  **OUTPUT:**    **CONCLUSION:**   This code demonstrates inheritance and method overriding in Java. The Rectangle class calculates the area and perimeter of a rectangle, while the Square class inherits from Rectangle and specializes for squares (where length equals breadth). By using inheritance, you’ve reused code efficiently. |
| 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 shape() {          System.out.println("This is shape");      }  }  class Rectangle extends Shape {      void rectangle() {          System.out.println("This is rectangular shape");      }  }  class Circle extends Shape {      void circle() {          System.out.println("This is circular shape");      }  }  class Square extends Rectangle {      void square() {          System.out.println("Square is a rectangle");      }  }  public class pra20 {      public static void main(String[] args) {          Square square = new Square();          square.shape();            square.rectangle();      }  }  **OUTPUT:**    **CONCLUSION:**  This code illustrates inheritance in Java. The Square class inherits from Rectangle, which in turn inherits from Shape. When you create a Square object, you can call both the shape() method (inherited from Shape) and the rectangle() method (specific to Rectangle). Inheritance promotes code reuse and establishes relationships between classes. |
| 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 Degree {      void getDegree() {          System.out.println("I got a degree");      }  }  class UG extends Degree {      @Override      void getDegree() {          System.out.println("I am an Undergraduate");      }  }  class PG extends Degree {      @Override      void getDegree() {          System.out.println("I am a Postgraduate");      }  }  public class pra21 {      public static void main(String[] args) {          Degree degree = new Degree();          degree.getDegree();          UG undergraduate = new UG();          undergraduate.getDegree();          PG postgraduate = new PG();          postgraduate.getDegree();      }  }  **OUTPUT:**    **CONCLUSION:**  This code demonstrates method overriding and polymorphism in Java. The UG and PG classes override the getDegree() method inherited from the Degree class. When you create objects of these classes, the appropriate overridden method is called based on the actual type of the object. Polymorphism allows flexibility and dynamic behavior. |
| 22. | 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.  **PROGRAM CODE:** interface AdvancedArithmetic {      int divi\_sum(int n);  }  class MyCalc implements AdvancedArithmetic {      @Override      public int divi\_sum(int n) {          int sum = 0;          for (int i = 1; i <= n; i++) {              if (n % i == 0) {                  sum += i;              }          }          return sum;      }  }  public class pra22 {      public static void main(String[] args) {          MyCalc myCalc = new MyCalc();          int n = 6;          System.out.println("The sum of divisors of " + n + " is: " + myCalc.divi\_sum(n));      }  }  **OUTPUT:**    **CONCLUSION:**  This code defines an interface called AdvancedArithmetic with a method divi\_sum(int n). The MyCalc class implements this interface and calculates the sum of divisors for a given integer n. The result is printed in the main method. |
| 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 CODE:**  interface Shape {      String getColor();      double getArea();      double getPerimeter();      default void printDetails() {          System.out.println("Color: " + getColor());          System.out.println("Area: " + getArea());          System.out.println("Perimeter: " + getPerimeter());      }  }  class Circle implements Shape {      private double radius;      private String color;      Circle(double radius, String color) {          this.radius = radius;          this.color = color;      }      @Override      public String getColor() {          return color;      }      @Override      public double getArea() {          return Math.PI \* radius \* radius;      }      @Override      public double getPerimeter() {          return 2 \* Math.PI \* radius;      }  }  class Rectangle implements Shape {      private double length;      private double width;      private String color;      Rectangle(double length, double width, String color) {          this.length = length;          this.width = width;          this.color = color;      }      @Override      public String getColor() {          return color;      }      @Override      public double getArea() {          return length \* width;      }      @Override      public double getPerimeter() {          return 2 \* (length + width);      }  }  class Sign {      private Shape shape;      private String text;      Sign(Shape shape, String text) {          this.shape = shape;          this.text = text;      }      void printSignDetails() {          System.out.println("Sign Text: " + text);          shape.printDetails();      }  }  public class pra23 {      public static void main(String[] args) {          Circle circle = new Circle(5, "Red");          Rectangle rectangle = new Rectangle(4, 6, "Blue");          Sign circleSign = new Sign(circle, "Welcome to the Campus Center");          Sign rectangleSign = new Sign(rectangle, "Event Here");          System.out.println("Circle Sign Details:");          circleSign.printSignDetails();          System.out.println("\nRectangle Sign Details:");          rectangleSign.printSignDetails();      }  }  **OUTPUT:**    **CONCLUSION:**  This code showcases composition and interface implementation in Java. The Circle and Rectangle classes implement the Shape interface, providing methods for color, area, and perimeter. The Sign class combines a shape (either circle or rectangle) with text to create sign details. Composition allows flexibility in constructing complex objects. |