Java\_practical-3

Q.1 Write a Java program to define a Person class with attributes name and age, and a subclass Student inheriting from Person with additional attributes grade and school ?

class Person {

String name;

int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

}

class Student extends Person {

int grade;

String school;

public Student(String name, int age, int grade, String school) {

super(name, age);

this.grade = grade;

this.school = school;

}

public int getGrade() {

return grade;

}

public String getSchool() {

return school;

}

}

public class Main {

public static void main(String[] args) {

// Creating a Person object

Person person = new Person("John", 30);

System.out.println("Person: ");

System.out.println("Name: " + person.getName());

System.out.println("Age: " + person.getAge());

// Creating a Student object

Student student = new Student("Alice", 18, 12, "High School");

System.out.println("\nStudent: ");

System.out.println("Name: " + student.getName());

System.out.println("Age: " + student.getAge());

System.out.println("Grade: " + student.getGrade());

System.out.println("School: " + student.getSchool());

}

}

Q.2 Write a java program for Method overloading

public class MethodOverloadingExample {

// Method to add two integers

public static int add(int a, int b) {

return a + b;

}

// Method to add three integers

public static int add(int a, int b, int c) {

return a + b + c;

}

// Method to add two doubles

public static double add(double a, double b) {

return a + b;

}

// Method to concatenate two strings

public static String add(String a, String b) {

return a + " " + b;

}

public static void main(String[] args) {

System.out.println("Addition of 5 and 3: " + add(5, 3));

System.out.println("Addition of 5, 3, and 2: " + add(5, 3, 2));

System.out.println("Addition of 2.5 and 3.5: " + add(2.5, 3.5));

System.out.println("Concatenation of 'Hello' and 'World': " + add("Hello", "World"));

}

}

//output

Addition of 5 and 3: 8

Addition of 5, 3, and 2: 10

Addition of 2.5 and 3.5: 6.0

Concatenation of 'Hello' and 'World': Hello World

Q. Write a java program for Constructor overloading ?

class xyz123 {

private String name;

private int age;

// Constructor with no parameters

public xyz123() {

name = "Unknown";

age = 0;

}

// Constructor with one parameter

public xyz123(String n) {

name = n;

age = 0;

}

// Constructor with two parameters

public xyz123(String n, int a) {

name = n;

age = a;

}

// Method to display name and age

public void display() {

System.out.println("Name: " + name);

System.out.println("Age: " + age);

}

}

// xyz.java

public class Main {

public static void main(String[] args) {

// Creating objects using different constructors

xyz123 xyz1 = new xyz123();

xyz123 obj2 = new xyz123("John");

xyz123 obj3 = new xyz123("Alice", 25);

System.out.println("Object 1:");

xyz1.display();

System.out.println("\nObject 2:");

obj2.display();

System.out.println("\nObject 3:");

obj3.display();

}

}

//output

Object 1:

Name: Unknown

Age: 0

Object 2:

Name: John

Age: 0

Object 3:

Name: Alice

Age: 25

Q. Write a java program to represent Abstract class with example. ?

abstract class Shape {

// Abstract method

abstract double area();

}

// Concrete subclass

class Circle extends Shape {

double radius;

// Constructor

public Circle(double radius) {

this.radius = radius;

}

// Implementation of abstract method

@Override

double area() {

return Math.PI \* radius \* radius;

}

}

public class Main {

public static void main(String[] args) {

// Creating an object of the concrete subclass Circle

Circle circle = new Circle(5);

// Calling the abstract method area()

double circleArea = circle.area();

// Displaying the area of the circle

System.out.println("Area of the circle: " + circleArea);

}

}

//output

Area of the circle: 78.53981633974483

**Q. Write a java program to implement Interface using extends keyword ?**

// Interface

interface Shape {

// Abstract method

double area();

}

// Interface extension

interface Drawable extends Shape {

// Abstract method

void display();

}

// Concrete class implementing both Shape and Drawable interfaces

class Circle implements Drawable {

double radius;

// Constructor

public Circle(double radius) {

this.radius = radius;

}

// Implementation of abstract method area() from Shape interface

@Override

public double area() {

return Math.PI \* radius \* radius;

}

// Implementation of abstract method display() from Drawable interface

@Override

public void display() {

System.out.println("Displaying a circle's area");

}

}

public class Main {

public static void main(String[] args) {

// Creating an object of the concrete class Circle

Circle circle = new Circle(5);

circle.display();

System.out.println("Area of the circle: " + circle.area());

}

}

Q. Write a java program to create inner classes ?

public class OuterClass {

    private int outerData;

    public OuterClass(int data) {

        this.outerData = data;

    }

    // Inner class

    public class InnerClass {

        private int innerData;

        public InnerClass(int data) {

            this.innerData = data;

        }

        public void display() {

            System.out.println("Outer Data: " + outerData);

            System.out.println("Inner Data: " + innerData);

        }

    }

        public static void main(String[] args) {

        OuterClass outerObj = new OuterClass(10);

        OuterClass.InnerClass innerObj = outerObj.new InnerClass(20);

        innerObj.display();

    }

}