Ques-1 Write a program declaring a class Rectangle with data member‟s length and breadth and member functions Input, Output and CalcArea.

import java.util.Scanner;

public class Rectangle {

private double length;

private double breadth;

public void input() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the length of the rectangle: ");

length = scanner.nextDouble();

System.out.print("Enter the breadth of the rectangle: ");

breadth = scanner.nextDouble();

}

public void output() {

System.out.println("Length: " + length);

System.out.println("Breadth: " + breadth);

}

public void calcArea() {

double area = length \* breadth;

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

}

public static void main(String[] args) {

Rectangle rectangle = new Rectangle();

rectangle.input();

rectangle.output();

rectangle.calcArea();

}

}

Ques-2 Write a program to demonstrate use of method overloading to calculate area of square, rectangle and triangle.

public class AreaCalculator {

public double calculateArea(double side) {

return side \* side; // Area of a square

}

public double calculateArea(double length, double breadth) {

return length \* breadth; // Area of a rectangle

}

public double calculateArea(double base, double height) {

return 0.5 \* base \* height; // Area of a triangle

}

public static void main(String[] args) {

AreaCalculator calculator = new AreaCalculator();

// Calculate the area of a square

double squareArea = calculator.calculateArea(5.0);

System.out.println("Area of square: " + squareArea);

// Calculate the area of a rectangle

double rectangleArea = calculator.calculateArea(4.0, 6.0);

System.out.println("Area of rectangle: " + rectangleArea);

// Calculate the area of a triangle

double triangleArea = calculator.calculateArea(3.0, 4.0);

System.out.println("Area of triangle: " + triangleArea);

}

}

Ques-3 Write a program to demonstrate the use of static variable, static method and static block.

public class StaticDemo {

static int count; // Static variable

static {

count = 0;

System.out.println("Static block executed.");

}

public StaticDemo() {

count++;

}

public static void staticMethod() {

System.out.println("Static method called.");

}

public static void main(String[] args) {

StaticDemo obj1 = new StaticDemo();

StaticDemo obj2 = new StaticDemo();

System.out.println("Count: " + count); // Accessing static variable

staticMethod(); // Invoking static method

}

}

Ques-4 Write a program to demonstrate concept of ``this``.

public class Person {

private String name;

private int age;

public Person(String name, int age) {

this.name = name; // using 'this' to refer to the instance variable

this.age = age;

}

public void printDetails() {

System.out.println("Name: " + this.name); // using 'this' to refer to the instance variable

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

}

public static void main(String[] args) {

Person person = new Person("John Doe", 25);

person.printDetails();

}

}

Ques-5 Write a program to demonstrate multi-level and hierarchical

// Vehicle class (base class)

class Vehicle {

void displayInfo() {

System.out.println("This is a vehicle.");

}

}

// Car class (derived from Vehicle)

class Car extends Vehicle {

void displayInfo() {

System.out.println("This is a car.");

}

}

// SportsCar class (derived from Car)

class SportsCar extends Car {

void displayInfo() {

System.out.println("This is a sports car.");

}

}

// Bike class (derived from Vehicle)

class Bike extends Vehicle {

void displayInfo() {

System.out.println("This is a bike.");

}

}

public class InheritanceDemo {

public static void main(String[] args) {

// Multi-level inheritance

SportsCar sportsCar = new SportsCar();

sportsCar.displayInfo(); // Output: This is a sports car.

// Hierarchical inheritance

Bike bike = new Bike();

bike.displayInfo(); // Output: This is a bike.

Car car = new Car();

car.displayInfo(); // Output: This is a car.

}

}

Ques-6 Write a program to use super() to invoke base class constructor.

class Person {

private String name;

public Person(String name) {

this.name = name;

}

public void displayInfo() {

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

}

}

class Student extends Person {

private int rollNo;

public Student(String name, int rollNo) {

super(name); // invoking the base class constructor using super()

this.rollNo = rollNo;

}

public void displayInfo() {

super.displayInfo(); // invoking the displayInfo() method of the base class using super()

System.out.println("Roll No: " + rollNo);

}

}

public class SuperDemo {

public static void main(String[] args) {

Student student = new Student("John Doe", 101);

student.displayInfo();

}

}

Ques-7 Write a program to demonstrate run-time polymorphism.

class Animal {

public void makeSound() {

System.out.println("The animal makes a sound.");

}

}

class Dog extends Animal {

public void makeSound() {

System.out.println("The dog barks.");

}

}

class Cat extends Animal {

public void makeSound() {

System.out.println("The cat meows.");

}

}

public class PolymorphismDemo {

public static void main(String[] args) {

Animal animal1 = new Animal();

Animal animal2 = new Dog();

Animal animal3 = new Cat();

animal1.makeSound(); // Output: The animal makes a sound.

animal2.makeSound(); // Output: The dog barks.

animal3.makeSound(); // Output: The cat meows.

}

}

Ques-8 Write a program to demonstrate the concept of aggregation.

class Address {

private String street;

private String city;

private String state;

public Address(String street, String city, String state) {

this.street = street;

this.city = city;

this.state = state;

}

public String getStreet() {

return street;

}

public String getCity() {

return city;

}

public String getState() {

return state;

}

}

class Employee {

private String name;

private int age;

private Address address; // Aggregation

public Employee(String name, int age, Address address) {

this.name = name;

this.age = age;

this.address = address;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

public Address getAddress() {

return address;

}

}

public class AggregationDemo {

public static void main(String[] args) {

Address address = new Address("123 Main St", "Cityville", "State");

Employee employee = new Employee("John Doe", 30, address);

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

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

System.out.println("Employee Address: " + employee.getAddress().getStreet() +

", " + employee.getAddress().getCity() +

", " + employee.getAddress().getState());

}

}

Ques-9 Write a program to demonstrate the concept of abstract class with constructor and ``final`` method.

abstract class Shape {

private String color;

public Shape(String color) {

this.color = color;

}

public String getColor() {

return color;

}

public abstract double calculateArea();

public final void displayInfo() {

System.out.println("Color: " + getColor());

System.out.println("Area: " + calculateArea());

}

}

class Circle extends Shape {

private double radius;

public Circle(String color, double radius) {

super(color);

this.radius = radius;

}

public double calculateArea() {

return Math.PI \* radius \* radius;

}

}

class Rectangle extends Shape {

private double length;

private double width;

public Rectangle(String color, double length, double width) {

super(color);

this.length = length;

this.width = width;

}

public double calculateArea() {

return length \* width;

}

}

public class AbstractClassDemo {

public static void main(String[] args) {

Circle circle = new Circle("Red", 5.0);

circle.displayInfo();

System.out.println();

Rectangle rectangle = new Rectangle("Blue", 4.0, 6.0);

rectangle.displayInfo();

}

}

Ques-10 Write a program to demonstrate the concept of interface when two interfaces have unique methods and same data members.

interface Animal {

int age = 10; // Common data member

void eat(); // Unique method

}

interface Mammal {

void sleep(); // Unique method

}

class Dog implements Animal, Mammal {

public void eat() {

System.out.println("The dog is eating.");

}

public void sleep() {

System.out.println("The dog is sleeping.");

}

public void displayInfo() {

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

}

}

public class InterfaceDemo {

public static void main(String[] args) {

Dog dog = new Dog();

dog.eat(); // Output: The dog is eating.

dog.sleep(); // Output: The dog is sleeping.

dog.displayInfo(); // Output: Age: 10

}

}

Ques-11 Write a program to demonstrate checked exception during file handling.

import java.io.File;

import java.io.FileReader;

import java.io.FileNotFoundException;

import java.io.IOException;

public class FileHandlingDemo {

public static void main(String[] args) {

File file = new File("input.txt");

try {

FileReader fr = new FileReader(file);

// Perform file operations here

fr.close();

} catch (FileNotFoundException e) {

System.out.println("File not found: " + e.getMessage());

} catch (IOException e) {

System.out.println("Error while reading file: " + e.getMessage());

}

}

}

Ques-12 Write a program to demonstrate unchecked exception

public class UncheckedExceptionDemo {

public static void main(String[] args) {

int[] numbers = {1, 2, 3, 4, 5};

int index = 6; // Trying to access an index out of bounds

try {

int result = numbers[index];

System.out.println("Value at index " + index + ": " + result);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Error: Index out of bounds!");

}

// This statement will not be executed due to the exception

System.out.println("Program continues...");

}

}

Ques-13 Write a program to demonstrate creation of multiple child threads.

Ques-14 Write a program to use Byte stream class to read from a text file and display the content on the output screen.

Ques-15 Write a program to demonstrate any event handling.

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class EventHandlingDemo extends JFrame implements ActionListener {

private JButton button;

public EventHandlingDemo() {

// Create a button

button = new JButton("Click Me");

// Register ActionListener

button.addActionListener(this);

// Add the button to the frame

add(button);

// Set frame properties

setTitle("Event Handling Demo");

setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == button) {

JOptionPane.showMessageDialog(this, "Button Clicked!");

}

}

public static void main(String[] args) {

new EventHandlingDemo();

}

}