Serving Education for Last 40 Years

NEERAJ®

MCS-206
Object Oriented Programming Using Java

Chapter Wise Reference Book
Including Many Solved Sample Papers

Based on

I.G.N.O.U.

& Various Central, State & Other Open Universities

By: *Anand Prakash Srivastava*

NEERAJ PUBLICATIONS
(Publishers of Educational Books)

Mob.: 8510009872, 8510009878 E-mail: info@neerajbooks.com
Website: www.neerajbooks.com

MRP ₹ 300/-

Content

OBJECT ORIENTED PROGRAMMING USING JAVA

QUESTION PAPER

June – 2024

(Solved)

OBJECT ORIENTED PROGRAMMING USING JAVA

MCS-206

Time: 3 Hours] *[Maximum Marks : 100*

Note: *Question No. 1 is compulsory. Attempt any three questions from the rest.*

Q. 1. *(a)* What is a class? How a class is defined in Java? Explain the differences between public and private classes.

Ans. Ref.: See Chapter-1, Page No. 6, Q. No. 1.

Also Add:

A Class is Defined in Java:

```
// Example of a simple class in Java
public class Car {
   // Fields (attributes)
   String color;
   int speed;

   // Method (behavior)
   void drive() {
      System.out.println("Car is driving");
   }
}
```

- public class Car declares a class named Car.
- color and speed are instance variables.
- drive() is a method of the class.

Differences between public and private Classes:

Feature	Public Class	Private Class *(Not allowed for top-level classes)*
Access	Accessible from **anywhere** in the program.	**Not accessible** outside the class (only inner classes can be private).
Use Case	For top-level classes that should be used widely.	Used to **hide inner helper classes** inside outer classes.
Top-Level Class	Allowed	Not allowed
Inner Class	Allowed	Allowed

***(b)* What is a relational operator? List and explain the use of different relational operators in Java.**

Ans. Ref.: See Chapter-2, Page No. 20, Q. No. 4.

***(c)* Write a Java program which take radius of a circle as input and display its area. Define appropriate class and methods in the program.**

Ans. Ref.: See Chapter-2, Page No. 21, Q. No. 7.

***(d)* What is inheritance? How it is implemented in Java ? Define Vehicle class and inherit classes "Two wheeler vehicle" and "four wheeler vehicle" from vehicle class. Define appropriate constructors in all these three classes.**

Ans. Ref.: See Chapter-5, Page No. 43, 'Basics of Inheritance'.

Also Add:

How Inheritance is Implemented in Java:

In Java, inheritance is implemented using the extends keyword:

```
class Child extends Parent {
   // Child inherits fields and methods from Parent
}
```

Java Program with Inheritance: Vehicle Example:

```
// Base class
class Vehicle {
   String brand;
   // Constructor for Vehicle
   Vehicle(String brand) {
      this.brand = brand;
```

```
            System.out.println("Vehicle Brand: " + brand);
        }
    }

    // Derived class for two-wheeler
    class TwoWheeler extends Vehicle {
        int wheels;

        // Constructor for TwoWheeler
        TwoWheeler(String brand, int wheels) {
            super(brand);  // Call to parent constructor
            this.wheels = wheels;
             System.out.println("This is a Two-Wheeler with " + wheels + " wheels.");
        }
    }

    // Derived class for four-wheeler
    class FourWheeler extends Vehicle {
        int wheels;

        // Constructor for FourWheeler
        FourWheeler(String brand, int wheels) {
            super(brand);  // Call to parent constructor
            this.wheels = wheels;
             System.out.println("This is a Four-Wheeler with " + wheels + " wheels.");
        }
    }

    // Main class to test the hierarchy
    public class VehicleDemo {
        public static void main(String[] args) {
            // Create TwoWheeler object
        TwoWheeler bike = new TwoWheeler("Honda", 2);

            // Create FourWheeler object
        FourWheeler car = new FourWheeler("Toyota", 4);
        }
    }
```

Output:

```
Vehicle Brand: Honda
This is a Two-Wheeler with 2 wheels.
Vehicle Brand: Toyota
This is a Four-Wheeler with 4 wheels.
```

***(e)* What is File class in Java? Explain the use of any two methods in File class.**

Ans. Ref.: See Chapter-11, Page No. 92, 'File Class and Its Methods'.

***(f)* What is assertions in Java? Explain how assertions are enabled and disabled in Java.**

Ans. Ref.: See Chapter-7, Page No. 58, 'Assertion and its Use' and Page No. 60, Q. No. 1.

***(g)* Explain the use of JDBC in two-tier system. Also, explain the need of JDBC driver.**

Ans. Ref.: See Chapter-15, Page No. 126, 'Type 2/Java to Native API'.

Also Add:

Use of JDBC in Two-Tier System:

JDBC (Java Database Connectivity) is an API in Java that allows Java applications to interact with databases.

What is a Two-Tier Architecture?

A **two-tier architecture** consists of:

1. Client Tier: The Java application (front-end or business logic).

2. Database Tier: The back-end database (e.g., MySQL, Oracle).

In this model, the client directly communicates with the database using **JDBC**.

How JDBC Works in Two-Tier System:

1. The Java program uses JDBC to connect directly to the database.

2. JDBC sends SQL queries to the database.

3. The database processes the query and returns results.

4. The JDBC API receives the results and provides them to the Java program.

Example Flow:

```
// Sample JDBC steps in a two-tier system
Connection conn = DriverManager.getConnection(dbURL, user, pass);
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM users");
```

Q. 2. *(a)* Write a Java program to explain the use of "switch case" and "break" statements.

Ans. Ref.: See Chapter-3, Page No. 27, Q. No. 3.

***(b)* What is method overloading? Implement it with the help of a program?**

Ans. Ref.: See Chapter-6, Page No. 53, Q. No. 1.

***(c)* What is abstract class? List its advantages. In java how an abstract class is defined? Can you crate object from an abstract class? Also, explain differences between abstract class and interface.**

Ans. Ref.: See Chapter-6, Page No. 52-53, 'Abstract Class' and 'Application of Abstract Class'.

OBJECT ORIENTED PROGRAMMING USING JAVA

Introduction to Java

1

INTRODUCTION

Java is described as a modern, high-level, and evolutionary programming language. It emphasizes object-oriented concepts, with a strong focus on simplicity and portability. Historical background: Java was developed by James Gosling at Sun Microsystems in the early 1990s, originally named "Oak". Java's key features include adaptability, real-world entity modeling (objects and classes), and its use in various applications like mobile, desktop, and web applications.

CHAPTER AT A GLANCE

INTRODUCTION TO OBJECT ORIENTED PROGRAMMING

Java supports object-oriented concepts like classes and objects, which are fundamental to the language. It inherits these concepts from languages like C++. Key object-oriented principles such as classes, objects, inheritance, and encapsulation are crucial for learning Java.

Objects and Class

Objects have states and behaviors, while classes define these states and behaviors for objects of the same type. Objects represent real-world entities (e.g., cars, dogs). Classes are blueprints for objects, defining data fields and methods.

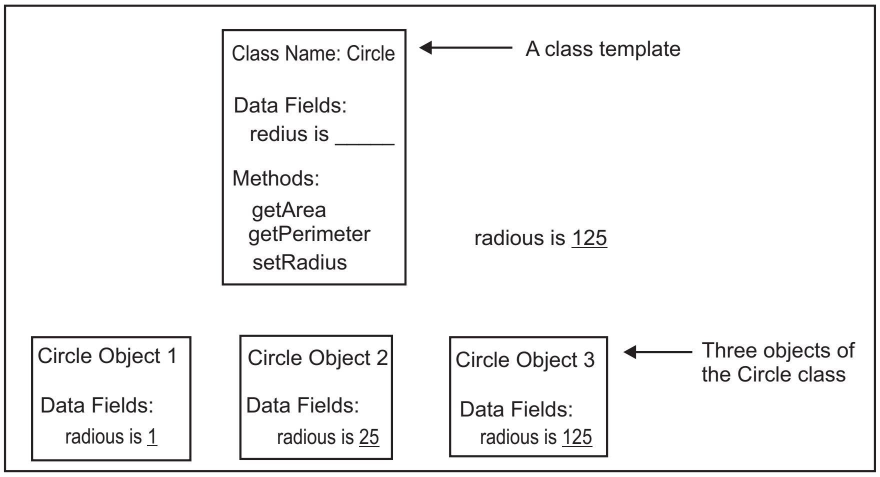

Fig.: A class template along with objects

Data Hiding

Protects data from external interference by restricting access to it and only allowing interaction through getter and setter methods.

Abstraction

Hides unnecessary details, showing only essential information (e.g., using a method without needing to understand its implementation).

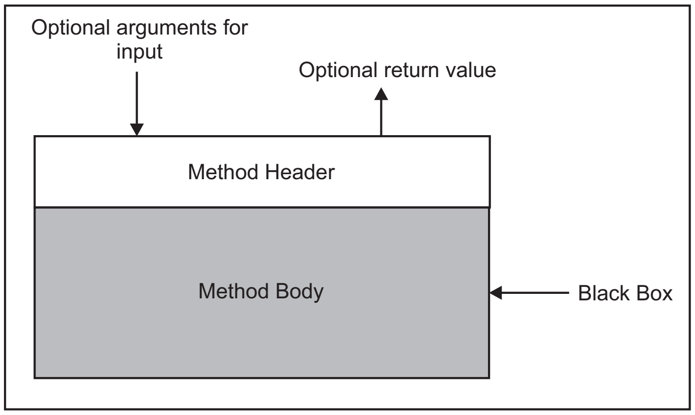

Fig. 1: Method body as a Black Box

Encapsulation

Bundling data and methods, ensuring that data is only accessible in a controlled manner.

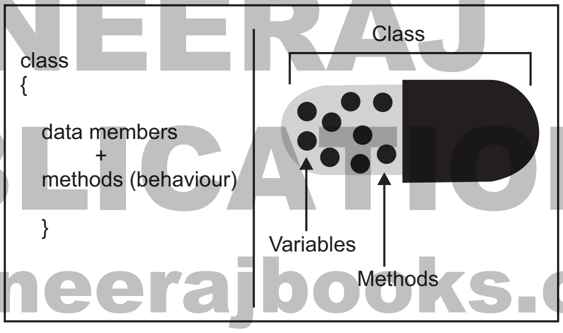

Fig. 2: Encapsulation in Java

Inheritance

Allows one class to inherit properties and methods from another, promoting code reuse. Figure illustrates inheritance, where three derived classes inherit features A and B from the base class and add their own unique features (C, D, F). This demonstrates code reusability and functionality extension. Further details on inheritance in Java will be covered in the next block of the course.

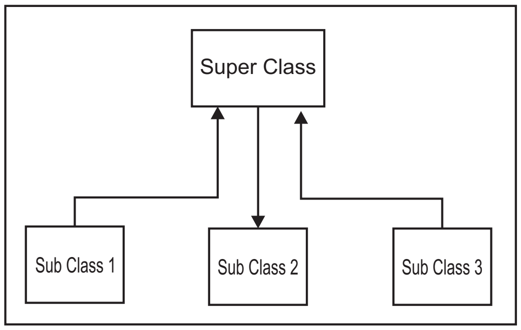

Fig. 3: Reprsentation of Inheritance

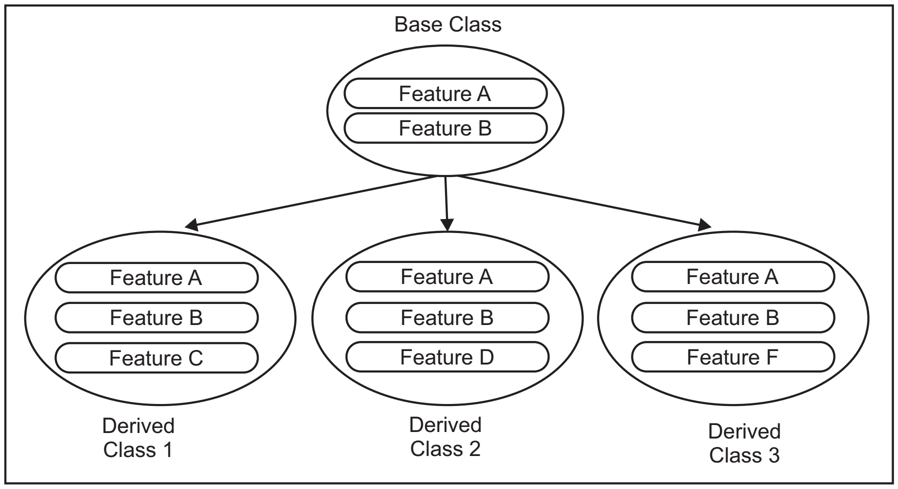

Fig. 4: Inheritance Hierarchy

Polymorphism

Allows objects to take many forms, enabling a single interface to be used for different actions (e.g., method overloading and overriding). Figure shows function overriding, where the same method name (draw()) is used for different shapes (Triangle, Circle, Rectangle) with different implementations. This is an example of polymorphism.

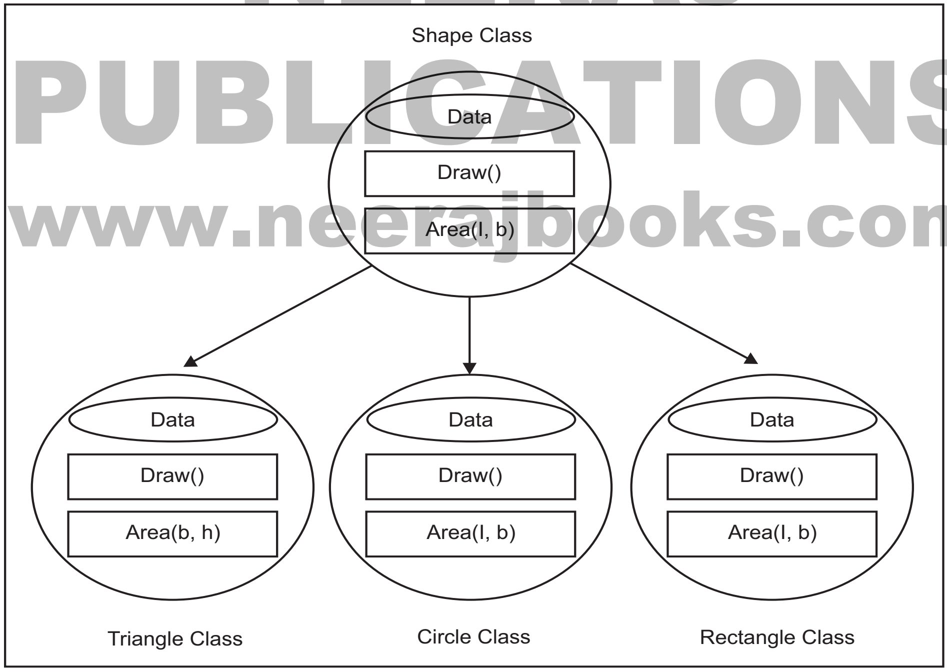

Fig. 5: Polymorphism

Polymorphism has two types:

***(i)* Static Polymorphism (Compile-time):** Achieved through method overloading, where methods have the same name but differ in parameters or return types, decided at compile time.

***(ii)* Dynamic Polymorphism (Run-time):** Achieved through method overriding, where the method to be called is decided by the JVM at runtime, based on the object type.

FEATURES OF JAVA

1. Object-Oriented: Everything in Java is treated as an object.

2. Simple: Java's syntax is user-friendly and easy to learn, especially for those familiar with C++.

3. Secure: Java supports virus-free development and runs within a secure virtual machine.

4. Portable: Java is platform-independent due to its use of bytecode, which can run on any device with a JVM. Java provides three types of portability:

***(a)* Source Code Portability:** Java programs produce identical results on any CPU, OS, or Java compiler.

***(b)* CPU Architecture Portability:** Java compilers produce bytecode that can run on any CPU with a JVM.

***(c)* OS/GUI:** Java libraries offer a virtual OS/GUI, allowing programs to work across different platforms with consistent functionality.

5. Robust: Java has strong memory management, automatic garbage collection, and exception handling.

6. Multithreaded: Java supports multithreading, making it ideal for networked applications.

7. Architecture-Neutral: Java can run on any platform due to the JVM.

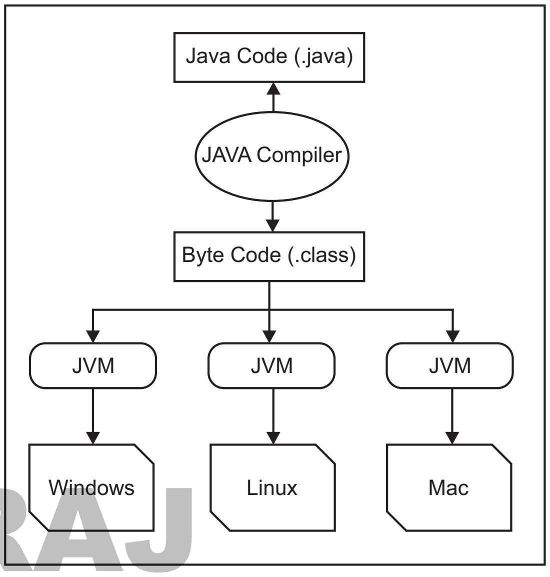

Fig. 6: Java is Architecture-neutral

8. Interpreted and High-Performance: Java uses JIT compilers to improve performance while retaining portability.

9. Distributed: Supports the development of distributed applications through TCP/IP protocols.

10. Dynamic: Java can dynamically load and link classes at runtime.

Use of Java in Industry

Java is widely used in the IT industry for various applications like web development, mobile apps (e.g., Android), scientific applications, and big data tools (e.g., Hadoop). It is also used in popular services like Twitter and for building servers and simulators.

JAVA DEVELOPMENT ENVIRONMENT AND TOOLS

Java programs are developed using the Java Development Kit (JDK), which includes tools for compiling and running Java applications. The development environment can be command-line-based or use an Integrated Development Environment (IDE) like Eclipse or NetBeans for easier program creation, compilation, and execution.

JDK

JDK is a software package that provides the tools necessary to develop Java applications. It includes the Java Runtime Environment (JRE), a compiler, and other development tools for creating Java programs.

JRE

JRE is a software layer that provides the necessary libraries and resources to run Java programs. It includes the JVM and class libraries required for execution, but does not contain development tools like the JDK.

JIT and JVM

JIT (Just-In-Time) Compiler: A component of the JVM that compiles bytecode into machine code at runtime, improving performance.

JVM (Java Virtual Machine): A virtual machine that executes Java bytecode, making Java platform-independent by translating bytecode into machine-specific code.