



KAMALDNP

kamaldnp.com • IT Training, Varanasi

PDF created by Raj Kamal using AI

Object Oriented Programming Using Java — Complete Syllabus

BTEUP Diploma in Computer Science & Engineering • 5th Semester • Session 2026-27
Subject 5.1 (Practicum) • Total 100 Marks • 3 Credits • L 1 - P 4

Examination Scheme

Code	Paper	Type	L - P	Credits	Internal	External	Total
5.1	Object Oriented Programming Using Java	Practicum	1 - 4	3	60	Practical 40 (3 hrs)	100
Java Total				3	60	40	100

What This Subject Aims To Do

To offer the modern programming language Java so students can implement object-orientation practically and program in object-oriented technology using Java. By the end you should be able to: install a Java IDE, compiler and Java virtual machines and understand basic Java concepts; implement basic Java programs; explain and implement object-oriented programming; explain and implement abstract classes, interfaces and polymorphism; and understand exceptions and develop multithreaded programs.

Unit-Wise Complete Topic List

Unit	Topic	Periods	Marks Share
1	Introduction and Features	2	15%
2	Language Constructs	3	15%

3	Classes and Objects	3	30%
4	Abstract Class, Interface and Polymorphism	3	20%
5	Exception Handling and Multithreading	3	20%

Unit 1 — Introduction and Features (2 Periods, 15%)

Topics: Fundamentals of object-oriented programming — procedure-oriented vs object-oriented programming (OOP); OOP concepts — classes, object, object reference, abstraction, encapsulation, inheritance, polymorphism; introduction of Eclipse (IDE) for developing programs in Java.

In simple words: Why OOP exists, how it differs from the older procedure-based style, and the core vocabulary — class, object, inheritance, polymorphism — plus your first tool, the Eclipse IDE.

Unit 2 — Language Constructs (3 Periods, 15%)

Topics: Variables; data types — integer, floating point, character, boolean; operators; iteration and jump statements; conditional expressions; input using Scanner class and output statements; loops; switch-case; arrays; introduction to methods.

In simple words: The Java basics: storing values, making decisions, repeating work, reading input, and grouping code into methods.

Unit 3 — Classes and Objects (3 Periods, 30%)

Topics: Class fundamentals; constructors; declaring objects (object and object reference); creating and accessing variables and methods; static and non-static variables/methods; defining packages; creating and accessing a package; importing packages; understanding CLASSPATH; auto-boxing; String and StringBuffer. Inheritance: definition; protected/private/public data; constructor chaining; order of invocation; types of inheritance (single, multilevel, hierarchical, hybrid); access control (private vs public vs protected vs default).

In simple words: The heart of the subject — and the heaviest-weighted unit. How to build classes, how objects are created, how code is organised into packages, and how one class inherits from another.

Unit 4 — Abstract Class, Interface and Polymorphism (3 Periods, 20%)

Topics: Defining an interface; key points of abstract class and interface; difference between an abstract class and an interface; implementation of multiple inheritance through interface; method and constructor overloading; method overriding; up-casting and down-casting.

In simple words: How Java handles the tricky parts: partial blueprints (abstract classes), pure contracts (interfaces), and the same method name behaving differently — polymorphism.

Unit 5 — Exception Handling and Multithreading (3 Periods, 20%)

Topics: Definition of exception handling; implementation of keywords try, catch, finally, throw and throws; built-in exceptions; difference between multi-threading and multi-tasking; thread life cycle; creating threads; thread priorities.

In simple words: How programs survive errors gracefully instead of crashing, and how one program can do several things at the same time with threads.

Practical Work

- WAP to create a simple class to find out the area and perimeter of a rectangle and a box, using super and this keywords.
- WAP to design a class Account using inheritance and static members, showing all functions of a bank (withdrawal, deposit).
- WAP to design a class using abstract methods and classes.
- WAP to design a String class that performs string methods (equal, reverse the string, change case).
- Consider a class of Cars under which Santro Xing, Alto and Wagon R are individual objects, each with model, year of manufacture, colour, top speed, etc. as properties and Create(), Sold() and display() as methods of the Car class.
- In a software company, Software Engineers, Sr. Software Engineers, Module Leads, Technical Leads, Project Leads, Project Managers, Program Managers and Directors are all employees but differ in work, perks, roles and responsibilities. Create an Employee base class providing common behaviours/properties that all employees must have.
- Using multiple inheritance, create classes: Shape, Circle, Square, Cube, Sphere and Cylinder, with specified class variables, constructors and base classes, with Area and/or Volume methods.
- WAP to handle exceptions using try and multiple catch blocks.
- WAP that implements nested try statements.

- WAP to create a package that accesses members of an external class as well as of the same package.
- WAP that shows the partial implementation of an interface.
- WAP to create a thread that implements the Runnable interface.

Reference Books (as listed in the syllabus)

- Programming with Java: A Primer — E. Balagurusamy
- Head First Java — Kathy Sierra & Bert Bates, O'Reilly
- OCA Java SE Programmer I Certification Guide — Mala Gupta, Wiley
- Programmer's Guide to Java SE 8 — Khalid A. Mughal, Pearson

KAMALDNP • kamaldnp.com • support@kamaldnp.com • +91 8318013442 • Varanasi, UP

Source: BTEUP 3rd Year curriculum (folder2627), approved in C.D.C. meeting held on 16.07.2026. Topic lists follow the official syllabus; short explanations are the author's own plain-English summaries, not syllabus text.

Please confirm with your own polytechnic for any local changes.

