

**SEMESTER-II****COMPREHENSIVE JAVA PROGRAMMING: FROM BASIC TO  
ADVANCED**

Course Name: Comprehensive Java Programming: From Basic to Advanced	Course Code	L-T-P	Credits
	ETCCJP271	3-0-2	4
Type of Course:	Major		
Pre-requisite(s):	Prior experience with programming logic, object-oriented concepts, and data structures (in any language like C++ or Python).		

**Course Perspective:**

This course covers the fundamentals to advanced features of Java, ensuring students can develop industry-standard Java applications. It includes topics such as multithreading, Java Collections, JDBC, GUI development, RESTful APIs, security, microservices, and deployment. The hands-on approach emphasizes real-world problem-solving and equips students to build and deploy scalable and secure Java applications.

**Defined Course Outcomes**

CO1	Implementing core Java concepts, OOP principles, and Java Collections in real-world applications.
-----	---



CO 2	Developing and optimizing Java applications using advanced features like multithreading, exception handling, and functional programming.
CO 3	Designing industry-standard applications integrating GUI (JavaFX/Swing), databases (JDBC, Hibernate), and file handling.
CO 4	Building and deploying modern Java applications using Spring Boot, RESTful APIs, microservices, security, and cloud deployment.

**Course Outline:**

<b>Unit Number:1</b>	<b>Core Java &amp; Object-Oriented Programming for Real-World Applications</b>	<b>No. of hours:12</b>
----------------------	--	------------------------

**Topics Covered:**

- Introduction to Java, JVM, JDK, and JIT Compilation
- Object-Oriented Programming (OOP) in Java: Classes, Objects, Inheritance, Polymorphism, Abstraction, and Encapsulation
- Java Collections Framework (List, Set, Map, Queue) for Data Management
- Functional Programming in Java: Lambda Expressions, Streams API, Optional Class
- Java 8+ Features: Streams, Date & Time API, CompletableFuture
- **Real-World Applications:**
  - Data processing using Java Collections
  - Implementing microservices using Java Streams and functional programming



<b>Unit Number:2</b>	<b>Advanced Java Concepts and Optimized Performance</b>	<b>No. of hours:11</b>
----------------------	---	------------------------

**Topics Covered:**

- Multithreading & Concurrency: Thread creation, Executors, Callable & Future
- Synchronization, Locks, Atomic Variables, Fork-Join Framework
- Exception Handling: Best practices, logging, custom exceptions
- Reflection API: Dynamic method invocation, annotations processing
- Design Patterns in Java: Singleton, Factory, Observer, Dependency Injection

**Real-World Applications:**

- Stock market price analyzer using multithreading
- E-commerce order processing system using Factory Design Pattern

<b>Unit Number:3</b>	<b>GUI Development, Database Connectivity &amp; File Handling</b>	<b>No. of hours:11</b>
----------------------	---	------------------------

**Topics Covered:**

- GUI Programming: JavaFX & Swing, Event Handling, FXML, Styling with CSS
- Database Connectivity (JDBC & Hibernate): CRUD Operations, Connection Pooling, ORM Mapping
- File Handling & Serialization: Handling JSON, XML, CSV, and Object Serialization
- Email & SMS Notification Integration: Java Mail API, Twilio API

**Real-World Applications:**

- Bank Management System: A GUI-based system with database integration
- Automated Report Generator: Generates PDF/Excel reports for an enterprise



<b>Unit Number:4</b>	<b>RESTful APIs, Microservices &amp; Secure Application</b>	<b>No. of hours:11</b>
	<b>Deployment</b>	

**Topics Covered:**

- RESTful Web Services (Spring Boot): Creating, Consuming APIs, JSON Parsing, Swagger Documentation
- Security in Java: JWT Authentication, OAuth2, Secure REST APIs
- Role-Based Access Control (RBAC) using Spring Security
- Microservices Architecture: Building Microservices with Spring Boot, API Gateway (Zuul/Spring Cloud Gateway)
- Deployment & Cloud Integration: Dockerizing a Java Application, Deploying on AWS/GCP using Jenkins

**Real-World Applications:**

- Real-time order tracking system using REST APIs & WebSockets
- Secure online payment system implementing JWT authentication

**Text and Reference Books**

- Herbert Schildt – *Java: The Complete Reference*
- Cay Horstmann – *Core Java Volume I – Fundamentals*
- Kathy Sierra & Bert Bates – *Head First Java*
- Y. Daniel Liang – *Introduction to Java Programming*
- Oracle Java SE Documentation (Official)



## Learning Outcomes

### Inside the Classroom

1. Mastery of Core Java and OOP Principles:
  - o Apply fundamental object-oriented concepts like inheritance, polymorphism, abstraction, and encapsulation in real-world coding scenarios.
2. Proficiency in Collections and Functional Programming:
  - o Utilize Java Collections Framework and Java 8+ features such as Streams and Lambda expressions for effective data manipulation.
3. Multithreading and Concurrency:
  - o Implement multi-threaded applications using Java concurrency tools and understand synchronization mechanisms.
4. Robust Error and Exception Handling:
  - o Apply best practices for exception handling, logging, and debugging in scalable applications.
5. Design Patterns and Reflection:
  - o Use common design patterns and Java Reflection API to build extensible and dynamic applications.
6. Building GUIs and Managing Databases:
  - o Design user interfaces with JavaFX/Swing and connect applications to databases using JDBC and Hibernate ORM.
7. Creating RESTful APIs and Microservices:
  - o Develop and secure REST APIs using Spring Boot, implement role-based security, and



explore microservice architecture.

8. Deployment and Integration:

- Deploy applications using Docker, integrate CI/CD with Jenkins, and publish to cloud platforms like AWS/GCP.

## Outside the Classroom

1. Project-Based Learning:

- Build full-stack Java applications with GUI, database, and backend integration as part of personal or group projects.

2. Exploration of Advanced Libraries and Tools:

- Independently explore APIs, libraries (e.g., Twilio, Swagger), and frameworks not covered in-depth in lectures.

3. Hands-on Practice with Deployment:

- Gain experience by deploying personal projects on cloud platforms using Docker and CI/CD pipelines.

4. Self-Driven Certification Preparation:

- Prepare for certifications such as Oracle Certified Professional: Java SE Programmer, Spring Boot Microservices, or Docker Associate.

5. Peer Collaboration and Code Review:

- Collaborate with peers on GitHub, participate in team-based software design activities, and provide/receive feedback.

6. Problem Solving in Real-World Scenarios:

- Apply learned concepts to build systems like payment gateways, reporting tools, and



tracking systems based on real-world needs.

#### 7. Exposure to Industry Practices:

- Follow industry coding standards, apply secure coding principles, and practice writing production-grade documentation and APIs.

## Lab Assignment

S.No.	Lab Task
1	<p><b>LAB TASK 1: OOP and Exception Handling – Library Management System</b></p> <p><b>Objective:</b> To implement a library system using class-based design and exception handling.</p> <p><b>Activities:</b></p> <ul style="list-style-type: none"><li>• Create classes for Book, Member, and Library</li><li>• Implement operations like issue, return, fine calculation</li><li>• Handle runtime exceptions (e.g., book not found, max limit reached)</li></ul> <p><b>Tools:</b> Java (JDK), VS Code/IntelliJ</p> <p><b>Learning Focus:</b> Object-oriented design, modular classes, exception-driven flow</p>
2	<p><b>LAB TASK 2: Collections and File I/O – Product Catalog Application</b></p> <p><b>Objective:</b> To manage and persist product data using Java collection frameworks and file streams.</p> <p><b>Activities:</b></p> <ul style="list-style-type: none"><li>• Use ArrayList and HashMap to store and manage products</li><li>• Implement product search, update, delete functions</li></ul>



	<ul style="list-style-type: none"><li>Save/load data using object streams (serialization)</li></ul> <p><b>Tools:</b> Java I/O, Java Collections, IntelliJ</p> <p><b>Learning Focus:</b> Collections API, file handling, search optimization</p>
3	<p><b>LAB TASK 3: GUI with Multithreading – Bank Transaction Simulator</b></p> <p><b>Objective:</b> To design a multithreaded GUI simulating real-time bank transactions.</p> <p><b>Activities:</b></p> <ul style="list-style-type: none"><li>Build JavaFX or Swing interface for bank operations</li><li>Use threads for deposit/withdraw actions across accounts</li><li>Implement synchronization and GUI feedback for actions</li></ul> <p><b>Tools:</b> JavaFX/Swing, Threads, Event Listeners</p> <p><b>Learning Focus:</b> GUI design, event-driven programming, concurrency</p>
4	<p><b>LAB TASK 4: JDBC Integration – Student Portal CRUD System</b></p> <p><b>Objective:</b> To build a console-based or GUI system connected to a relational database.</p> <p><b>Activities:</b></p> <ul style="list-style-type: none"><li>Connect to MySQL using JDBC</li><li>Perform CRUD operations via user input</li><li>Use PreparedStatement for secure queries</li></ul> <p><b>Tools:</b> JDBC, MySQL/Oracle, IntelliJ</p> <p><b>Learning Focus:</b> Database integration, data persistence, security practices</p>
5	<p><b>Capstone Project: Java-Based Inventory &amp; Billing System</b></p>



	<p><b>Objective:</b></p> <p>To build a fully functional Java application for managing inventory, transactions, and billing using file I/O, database, and GUI integration.</p> <p><b>Project Tasks:</b></p> <ul style="list-style-type: none"><li>• Design reusable classes for products, bills, users</li><li>• GUI interface for inventory operations</li><li>• Save records in file/database</li><li>• Generate invoice in PDF/text format using Java I/O or external libraries</li></ul> <p><b>Learning Focus:</b> Full-cycle Java application, OOP design, GUI + DB integration</p> <p><b>Tools:</b> Java, JavaFX/Swing, JDBC, MySQL, Apache POI (optional)</p>
--	---