Java Course Curriculum by 10Pie

👉 Whether you want to start your IT career or looking for guidance on how to plan your future career path, 10Pie is your learning hub for tech career knowledge.

Here's a Java course syllabus at a glance:

Sl No.	Module Name	Topic Names	Projects to Practice
1	**Introduction to Java**	- History and Features of Java - Java Development Kit (JDK) - Java Runtime Environment (JRE) - Java Virtual Machine (JVM) - Hello Java Program - Java Identifiers and Keywords - Java Data Types - Java Operators - Java Control Statements	1. Basic Calculator 2. Student Grade System
2	**Object-Oriented Programming in Java**	- Classes and Objects - Constructors - Methods - Inheritance - Polymorphism - Abstract Classes and Methods - Interfaces - Packages - Exception Handling - Inner Classes	1. Library Management System 2. Bank Account Management
3	**Java Collections Framework**	- Collection Interfaces - Collection Classes - Generics - Streams API	1. Contact Management System 2. To-Do List Application
4	**Java I/O and NIO**	- File I/O - Serialization and Deserialization - NIO (New I/O)	1. File-Based Student Record System 2. Simple Note-Taking Application
5	**Multithreading**	- Thread Creation - Thread Synchronization - Thread Coordination - Concurrent Collections - Executor Framework - Atomic Variables	1. Multi-threaded Chat Application 2. Producer-Consumer Problem

6	**Java Database Connectivity (JDBC)**	- JDBC Architecture - Executing SQL Queries - ResultSet Interface - Transaction Management - RowSet Interface	1. Simple JDBC CRUD Application 2. Employee Management System
7	**Java Enterprise Edition (Java EE)**	- Servlets - JavaServer Pages (JSP) - JavaServer Faces (JSF) - Enterprise JavaBeans (EJB) - Java Persistence API (JPA) - Web Services	1. Simple Web Application with Servlets 2. JSP-based Online Quiz Application
8	**Spring Framework**	- Spring Core - Spring MVC - Spring Boot - Spring Data - Spring Security - Spring Cloud	1. Spring Boot CRUD Application 2. Spring MVC Web Application
9	**Java Frameworks and Libraries**	- Hibernate - Maven - JUnit - Log4j - Java EE Servers	1. Hibernate-based E-commerce Application 2. JUnit Testing for a Library Management System
10	**Advanced Java Concepts**	- Design Patterns - Java 8 Features - Java 9 Modules - Java 10 Local Variable Type Inference - Java 11 HTTP Client - Java 12 Switch Expressions - Java 13 Text Blocks	1. Design Patterns Showcase 2. Java 8 Features Implementation

Module 1: Introduction to Java

History and Features of Java

- Object-Oriented Programming (OOP)
- Platform Independence (Write Once, Run Anywhere)
- Automatic Memory Management
- Garbage Collection

Java Development Kit (JDK)

- JDK Components: javac, java, javadoc, etc.
- JDK Installation and Configuration

Java Runtime Environment (JRE)

- JRE Components: Java Virtual Machine (JVM), Java Class Library
- JRE Installation and Configuration

Java Virtual Machine (JVM)

- JVM Architecture
- JVM Memory Management: Heap, Stack, Method Area, etc.
- JVM Execution Engine

Hello Java Program

- Syntax and Structure
- Compiling and Running Java Programs

Java Identifiers and Keywords

- Naming Conventions
- Reserved Keywords

Java Data Types

- Primitive Data Types: byte, short, int, long, float, double, char, boolean
- Reference Data Types: Classes, Interfaces, Arrays

Java Operators

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Ternary Operator

Java Control Statements

- Selection Statements: if-else, switch
- Iteration Statements: for, while, do-while
- Jump Statements: break, continue, return

⭐ **Hands-on projects to practice:**

- **Basic Calculator:** Create a console-based calculator that performs basic arithmetic operations (addition, subtraction, multiplication, division) using methods.

- **Student Grade System:** Develop a program that takes student names and their grades as input and calculates the average grade, determining if the student has passed or failed.
- **Number Guessing Game:** Implement a simple game where the computer randomly selects a number, and the user has to guess it. Provide hints if the guess is too high or too low.

Module 2: Object-Oriented Programming in Java

Classes and Objects

- Class Definition and Instantiation
- Instance Variables and Methods
- Access Modifiers: public, private, protected, default

Constructors

- Default Constructor
- Parameterized Constructor
- Constructor Overloading
- Constructor Chaining

Methods

- Method Signature and Return Types
- Method Overloading
- Method Overriding
- Static Methods
- Final Methods

Inheritance

- Superclass and Subclass
- Types of Inheritance: Single, Multilevel, Hierarchical, Multiple
- Method Overriding and Super Keyword

Polymorphism

- Method Overloading and Overriding
- Dynamic Method Dispatch
- Abstract Classes and Methods

Interfaces

- Interface Definition and Implementation
- Default and Static Methods

- Multiple Inheritance using Interfaces

Packages

- Package Definition and Naming Conventions
- Importing Packages
- Access Protection in Packages

Exception Handling

- Types of Exceptions: Checked and Unchecked Exceptions
- try-catch-finally Block
- Throw and Throws Keywords
- Custom Exception Creation

Inner Classes

- Member Inner Class
- Local Inner Class
- Anonymous Inner Class
- Static Nested Class

⭐ **Hands-on projects to practice:**

- **Library Management System:** Create a system that manages books in a library, including adding, removing, and searching for books using classes and objects.
- **Bank Account Management:** Develop a program that simulates bank account operations (deposit, withdraw, check balance) using constructors and methods.
- **Simple E-commerce Application:** Build a console application that allows users to add products to a cart, calculate total prices, and manage inventory using OOP principles.

Module 3: Java Collections Framework

Collection Interfaces

- List Interface: ArrayList, LinkedList
- Set Interface: HashSet, TreeSet, LinkedHashSet
- Queue Interface: PriorityQueue
- Deque Interface: ArrayDeque
- Map Interface: HashMap, TreeMap, LinkedHashMap

Collection Classes

- Utility Classes: Collections, Arrays
- Comparable and Comparator Interfaces

Generics

- Generic Classes and Methods
- Type Parameters
- Bounded Type Parameters
- Wildcard Types

Streams API

- Stream Creation
- Intermediate Operations: filter, map, flatMap, sorted, distinct
- Terminal Operations: forEach, collect, reduce, match, count
- Parallel Streams

⭐ **Hands-on projects to practice:**

- **Contact Management System:** Implement a program that stores and manages contacts using ArrayList and HashMap, allowing users to add, remove, and search for contacts.
- **To-Do List Application:** Create a console-based to-do list where users can add tasks, mark them as completed, and view all tasks using collections.
- **Student Record System:** Develop a system that stores student records using a List of objects and allows sorting and filtering based on different criteria.

Module 4: Java I/O and NIO

File I/O

- File and Directory Operations
- FileInputStream and FileOutputStream
- FileReader and FileWriter
- BufferedReader and BufferedWriter

Serialization and Deserialization

- ObjectInputStream and ObjectOutputStream
- Serializable Interface
- Transient and Static Keywords

NIO (New I/O)

- Buffers: ByteBuffer, CharBuffer, IntBuffer, etc.
- Channels: FileChannel, SocketChannel, ServerSocketChannel
- Selectors and SelectionKeys
- Path and Paths
- Files and StandardOpenOption

⭐ **Hands-on projects to practice:**

- **File-Based Student Record System:** Create a program that reads and writes student records to a text file, allowing users to add, view, and delete records.
- **Simple Note-Taking Application:** Develop a console application that allows users to create, read, update, and delete notes, saving them to a file.
- **Data Serialization Example:** Implement a program that serializes and deserializes an object (e.g., a student) to and from a file.

Module 5: Multithreading

Thread Creation

- Extending Thread Class
- Implementing Runnable Interface
- Thread Priorities

Thread Synchronization

- Synchronized Methods and Blocks
- Deadlock and Starvation
- Reentrant Locks

Thread Coordination

- wait(), notify(), and notifyAll() Methods
- Condition Interface

Concurrent Collections

- ConcurrentHashMap
- CopyOnWriteArrayList
- BlockingQueue and its Implementations

Executor Framework

- Executor Interface
- ExecutorService and Callable Interface
- Future and FutureTask

Atomic Variables

- AtomicInteger, AtomicLong, AtomicReference
- Atomic Operations

⭐ **Hands-on projects to practice:**

- **Multi-threaded Chat Application:** Build a simple console-based chat application that allows multiple users to send and receive messages simultaneously using threads.
- **Producer-Consumer Problem:** Implement a solution to the producer-consumer problem using threads and synchronization techniques.
- **Download Manager:** Create a program that simulates downloading files using multiple threads, showing progress for each download.

Module 6: Java Database Connectivity (JDBC)

JDBC Architecture

- JDBC Drivers
- DriverManager Class
- Connection Interface

Executing SQL Queries

- Statement Interface
- PreparedStatement Interface
- CallableStatement Interface

ResultSet Interface

- Navigating ResultSet
- Updating ResultSet

Transaction Management

- setAutoCommit(), commit(), and rollback() Methods
- Savepoints

RowSet Interface

- JdbcRowSet
- CachedRowSet
- WebRowSet

⭐ Hands-on projects to practice:

- **Simple JDBC CRUD Application:** Develop a console application that performs Create, Read, Update, and Delete operations on a database table (e.g., student records).
- **Employee Management System:** Create a program that manages employee records using JDBC, allowing users to add, update, and delete employee data.
- **Library Database System:** Build a library management system that interacts with a database to manage books, authors, and borrowers.

Module 7: Java Enterprise Edition (Java EE)

Servlets

- Servlet Life Cycle
- HttpServletRequest and HttpServletResponse
- Servlet Filters and Listeners

JavaServer Pages (JSP)

- JSP Life Cycle
- Scripting Elements: Declarations, Expressions, Scriptlets
- JSP Directives: page, include, taglib
- JSP Actions

JavaServer Faces (JSF)

- JSF Architecture
- Managed Beans and Scopes
- JSF Components and Renderers
- Navigation and Templating

Enterprise JavaBeans (EJB)

- Session Beans: Stateless, Stateful, Singleton
- Message-Driven Beans
- EJB Container and EJB Lifecycle

Java Persistence API (JPA)

- Entity Classes and Annotations
- EntityManager and EntityManagerFactory
- JPQL (Java Persistence Query Language)

Web Services

- SOAP Web Services
- RESTful Web Services
- JAX-WS and JAX-RS APIs

⭐ Hands-on projects to practice:

- **Simple Web Application with Servlets:** Create a web application using servlets that handles user requests and responses, displaying dynamic content.
- **JSP-based Online Quiz Application:** Develop a web-based quiz application using JSP that allows users to take quizzes and view results.

- **RESTful Web Service:** Implement a RESTful web service using JAX-RS that provides CRUD operations for a resource (e.g., products).

Module 8: Spring Framework

Spring Core

- Inversion of Control (IoC) and Dependency Injection (DI)
- ApplicationContext and BeanFactory
- Bean Scopes and Lifecycle

Spring MVC

- DispatcherServlet and HandlerMapping
- @Controller and @RequestMapping Annotations
- Model and ModelAndView
- Spring MVC Configuration

Spring Boot

- Auto-configuration
- Embedded Servers
- Spring Initializr
- Spring Boot Actuator

Spring Data

- Spring Data JPA
- Spring Data MongoDB
- Spring Data Redis

Spring Security

- Authentication and Authorization
- Web Security Configuration
- Method Security

Spring Cloud

- Service Discovery with Eureka
- Load Balancing with Ribbon
- Circuit Breaker with Hystrix

⭐ **Hands-on projects to practice:**

- **Spring Boot CRUD Application:** Build a simple Spring Boot application that performs CRUD operations on a database using Spring Data JPA.
- **Spring MVC Web Application:** Create a web application using Spring MVC that handles user requests and displays data using Thymeleaf.
- **Secure REST API with Spring Security:** Develop a REST API secured with Spring Security that requires authentication for accessing certain endpoints.

Module 9: Java Frameworks and Libraries

Hibernate

- Object-Relational Mapping (ORM)
- SessionFactory and Session
- HQL (Hibernate Query Language)
- Criteria API

Maven

- Project Object Model (POM)
- Dependency Management
- Build Automation

JUnit

- Unit Testing
- Assertions and Annotations
- Test Suites and Runners

Log4j

- Logging Levels
- Appenders and Layouts
- Configuration Files

Java EE Servers

- Apache Tomcat
- JBoss/WildFly
- GlassFish
- IBM WebSphere

⭐ Hands-on projects to practice:

- **Hibernate-based E-commerce Application:** Implement an e-commerce application using Hibernate for ORM, managing products and orders.

- **JUnit Testing for a Library Management System:** Create unit tests for the library management system using JUnit to ensure code correctness.
- **Logging Implementation with Log4j:** Develop a console application that uses Log4j to log various events and errors.

Module 10: Advanced Java Concepts

Design Patterns

- Creational Patterns: Singleton, Factory, Abstract Factory
- Structural Patterns: Adapter, Decorator, Proxy
- Behavioral Patterns: Observer, Strategy, Command

Java 8 Features

- Lambda Expressions
- Method References
- Functional Interfaces
- Streams API
- Date and Time API

Java 9 Modules

- Module System
- Jigsaw Project
- Modular JDK

Java 10 Local Variable Type Inference

- var Keyword
- Type Inference in Lambda Expressions

Java 11 HTTP Client

- HttpClient API
- WebSocket API

Java 12 Switch Expressions

- Enhanced switch Statements
- Pattern Matching for instanceof

Java 13 Text Blocks

- Multiline String Literals
- Escape Sequences

⭐ **Hands-on projects to practice:**

- **Design Patterns Showcase:** Implement various design patterns (e.g., Singleton, Factory, Observer) in a sample application to demonstrate their use.
- **Java 8 Features Implementation:** Create a program that utilizes Java 8 features such as streams, lambda expressions, and the new Date and Time API.
- **Microservices Architecture Example:** Develop a simple microservices-based application that demonstrates service discovery and communication between services.

Java course subjects and topics to learn

As of 2024, with the rise of new tech and AI, technology leaders believe that there's a skills gap within their teams. This is creating a bigger impact, demanding a requirement of adaptive Java skills.

These are the most important Java course subjects and topics you must learn as per Statista and a skills gap report:

AI and machine learning with Java

Java includes important libraries like Deeplearning4j, Weka, and TensorFlow (via Java API). These can be effectively used to develop AI and machine learning models. This is an important subject to learn as learning AI with Java is essential for students to close the skills gap.

You can get started by following Deeplearning4j's Quickstart guide to implement neural networks and other AI models using Java. This helps you create small AI projects, like image recognition or text classification.

Data Analytics

Since Java is commonly used for large-scale processing frameworks, it is highly relevant for data analytics. A skill gap in this area means there's a strong demand for developers who understand how to build large-scale, high-performance data systems.

Some important tools:

- Apache Hadoop
- Apache Spark
- Java Streams API

You can learn Apache Spark using the Spark documentation for Java. Once you build a hands-on project, you can opt for online courses on big data and Apache Spark from platforms like Udemy or Coursera.

Back-end development

Java has been one of the top choices to build back-end systems due to its scalability and frameworks like Spring Boot. This is an important skill for students to be able to build good applications for enterprises. Consider learning tools like Spring Boot, Java Persistence API, and RESTful APIs.

Follow this Spring Boot guide to master it and create a basic REST API, integrating it with a database using JPA. Then, you can work on a small project like an e-commerce back-end to further understand microservices and back-end architecture.

Cybersecurity with Java

As discussed above, Java is used to develop secure applications, especially in sectors that need enterprise-level security. Keep track of best practices in security, such as:

- Encryption and SSL
- Secure coding
- Java Security APIs
- OWASP Top Ten

These help you work in fields like financial services and for the government. Once you learn Java Security, practice securing data with encryption techniques. You can even work through a cybersecurity course to understand best practices and common mistakes.

Full-stack development

Java is flexible enough that it allows both back-end and front-end development. Tools like Spring Boot and Thymeleaf for back-end and JavaScript frameworks (React/Angular) for front-end equip you best for a Java developer role.

Lastly, build a full-stack web application that performs CRUD (create, read, update, delete) operations.

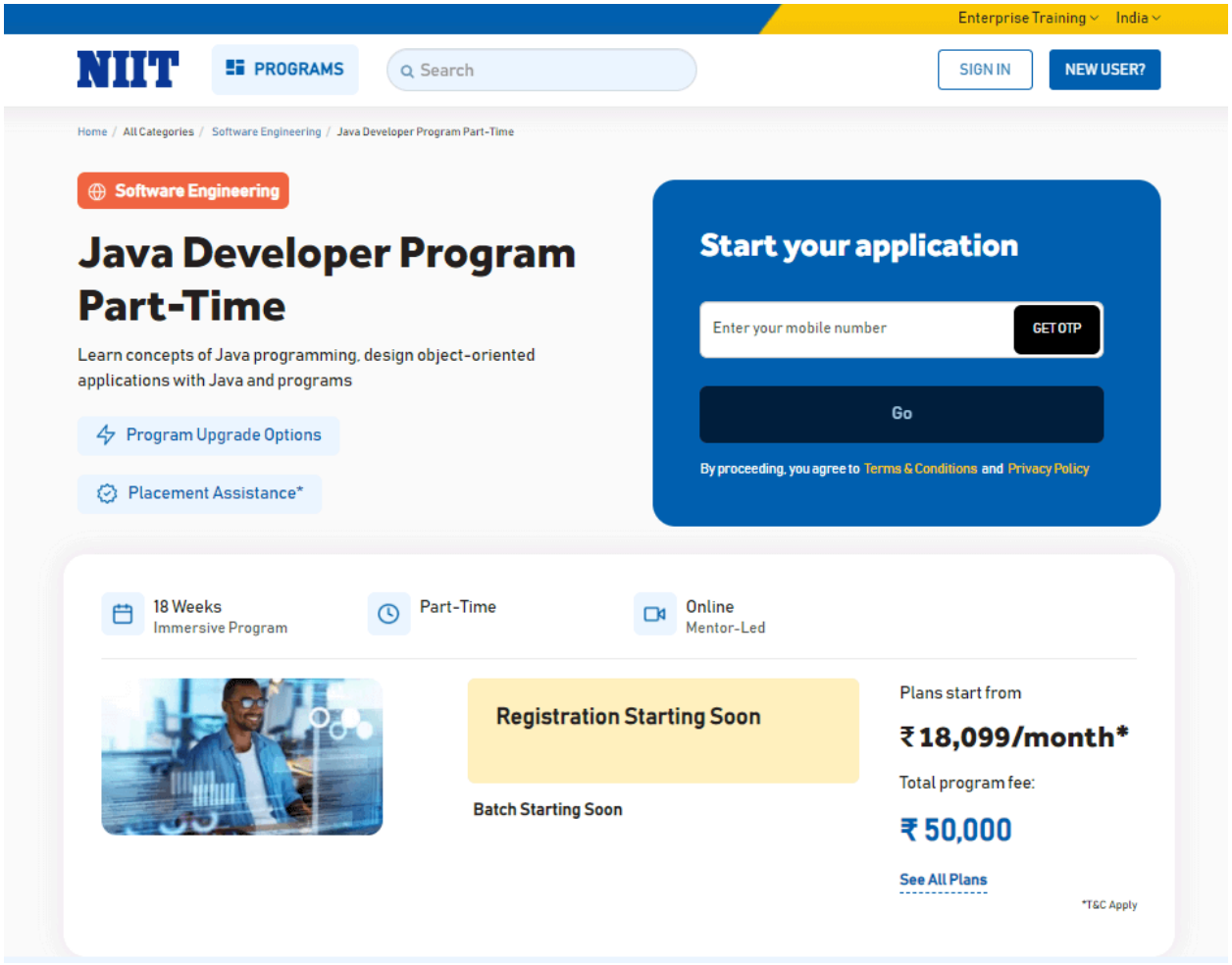
Java Course Fees and Duration 2024

There are various types of Java courses, such as obtaining the skill set through formal education or online courses. Whatever the format is, you must choose those that provide in-depth Java programming courses. They must prepare you for various careers in tech.

What is the course fee for Java Courses?

The course fee of Java courses ranges between ₹7000 to ₹1,00,000. The fees majorly depend on the duration of the course, placement, projects, internships, and accreditation.

For example, Codegnan offers a one-month foundational course between ₹20,000 to ₹25,000, while NIIT provides a 4-month part-time Java Developer program starting from ₹50,000.



Java course duration

You can expect a Java course to last from one month to 6 months, on average. Some institutions or courses offer different packages for different levels of coaching. For instance, foundational Java courses are one month to three months long, while more advanced Java courses last more than four months.

Related read:

- 10 BEST Java Training Courses in Bangalore
- 10 Java training courses in Hyderabad (2024)

Who is eligible for Java courses?

For online Java courses, there are typically no strict eligibility criteria. However, having the following knowledge and skills can be beneficial:

- Basic computer knowledge and familiarity with programming concepts.
- Prior experience in any programming language (like C, C++, or Python), although not mandatory.

Additionally, an educational background in computer science, information technology, or a related field is often preferred, but not strictly necessary.