MockAttempt

TECH ACADEMY • OFFICIAL COURSE SYLLABUS BROCHURE

MockAttempt Academy Pvt. Ltd. • Live Architect Mentorship, AI Assessments & Client Production

Admissions & Support:
support@mockattempt.com
Phone / WhatsApp: +91 98765 43210
Course Code: JAVA • **Updated:** 01 Nov 2026
https://mockattempt.com/courses/java-training

JAVA • PROGRAMMING & BACKEND **BEGINNER TO JOB-READY JAVA DEVELOPER**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

Java Training

Master Java from JVM internals, OOPs, Strings, Exceptions & Multithreading to Collections, Generics, Java 8+ Streams & MySQL JDBC — with runnable Java programs, 10×50 MCQ Mock Tests per topic, and Live Voice + Gemini AI Technical Interviews at the most affordable fee.

Duration: 16–18 Weeks • Live + Hands-On Projects (209 Hrs) • **Modules:** 16 • **Live Topics:** 53 • **Labs & Tasks:** 106 • **Mocks / AI Vivas:** 386 / 386 • **Projects:** 16 Module + 1 Live Client Prod

Fixed Non-Negotiable Tuition Fee: Base Fee **₹3,499** + 18% GST (₹630) = **Total Incl. GST: ₹4,129**

Live Batch Tracks: Weekday Live Online Batch (Mon – Fri) (Monday to Friday (5 Live Interactive Classes / Week) • 1.5 Hours / Day (60 Mins Live Architecture & Coding + 30 Mins Hands-On Lab) • 17–19 Weeks Fast-Track) | Weekend Intensive Live Online Batch (Sat – Sun) (Saturday & Sunday (Designed for Working Professionals & College Students) • 3.0 Hours / Day (2 Hrs Deep-Dive Live Coding + 1 Hr Guided Project Build) • 19–21 Weeks Weekend Track)

Core Technologies & Engineering Stack: Java 21 LTS • JVM & Memory Model • OOPs & Solid • Multithreading • Collections & Generics • Lambda & Stream API • JDBC & MySQL

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Understand JDK, JRE, JVM ClassLoader, Heap vs Stack memory, and JIT compilation internals
2. Write clean object-oriented Java programs using Inheritance, Polymorphism, Interfaces, Exceptions, and Multithreading
3. Master ArrayList, HashSet, HashMap, Generics, Java 8+ Lambdas/Streams, and SQL-injection-safe MySQL JDBC
4. Practice 350 Timed Mock Exams (17,500 Java MCQs) and 350 Spoken Voice + Gemini AI Technical Interviews
5. Java 21 LTS features in production: Records, Sealed Classes, Pattern Matching, Streams, and Optional safety.
6. High-throughput concurrency using Project Loom Virtual Threads, `CompletableFuture`, and lock-free collections.
7. JVM memory model literacy (Heap generations, G1/ZGC garbage collection, thread dumps, and leak profiling).
8. Clean SOLID object-oriented design, Maven multi-module builds, and pooled transactional JDBC data access.

Eligibility & Prerequisites: Zero prior Java experience required — starts from JDK setup, bytecode compilation, and Hello World • Any basic laptop/PC capable of running JDK 17/21 and VS Code / IntelliJ Community Edition • No prior Java industry experience required — Module 1 starts from environment setup, core mental models, and plain-English fundamentals. • Basic computer & internet familiarity on a Windows, macOS, or Linux laptop (8 GB+ RAM recommended for smooth live coding labs). • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews.

2. DETAILED MODULE & TOPIC-WISE SYLLABUS (16 MODULES • 53 LIVE TOPICS)

MODULE 1 OF 16 • 12 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: Java Fundamentals, JVM/JDK/JRE Architecture, Data Types & Operators

Master Java platform independence (WORA), JDK vs JRE vs JVM internals, primitive vs reference types, type promotion, and operators.

Objectives: ✓ Understand bytecode compilation & JVM ClassLoader/JIT • ✓ Use all 8 primitive types without overflow traps • ✓ Apply arithmetic, bitwise, shift, and short-circuit operators

1.1 Features of Java — Platform Independence, Bytecode & Object-Oriented Design [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Features of Java — Platform Independence, Bytecode & Object-Oriented Design in real Java systems — specifically: Write Once Run Anywhere (WORA), j...

1.2 JVM Architecture — ClassLoader, Heap, Stack, Metaspace & JIT Compiler [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply JVM Architecture — ClassLoader, Heap, Stack, Metaspace & JIT Compiler in real Java systems — specifically: Bootstrap/Application ClassLoaders, Byt...

1.3 Java Data Types — 8 Primitives, Wrapper Autoboxing & Type Casting [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Data Types — 8 Primitives, Wrapper Autoboxing & Type Casting in real Java systems — specifically: Exact bit widths of byte, short, int, long,...

1.4 Java Operators — Arithmetic, Bitwise, Shift (>> vs >>>), Short-Circuit & Ternary [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Operators — Arithmetic, Bitwise, Shift (>> vs >>>), Short-Circuit & Ternary in real Java systems — specifically: Pre/post increment (++i vs i...

Module 1 Hands-On Capstone Project (100 Marks): Smart CLI GST Invoice, Loan EMI & Income Tax Calculator in Java — Build an interactive console billing & financial calculator in Java 21 using Scanner, formatted printf tables, compound/EMI math, and strict numeric range validation.

MODULE 2 OF 16 • 12 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: Java Control Flow, Switch Expressions, Loops & Arrays

Write structured algorithmic logic with if-else ladders, modern Java switch expressions (->), labeled loops, and 1D/2D/jagged arrays.

Objectives: ✓ Prevent switch fall-through bugs using modern arrow switch • ✓ Write efficient for, enhanced for-each, and while loops • ✓ Manipulate 1D/2D arrays and java.util.Arrays binarySearch/sort

2.1 Java Conditional Branching — if, else-if Ladders & Guard Clauses [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Conditional Branching — if, else-if Ladders & Guard Clauses in real Java systems — specifically: Boolean condition requirements in Java (no t...

2.2 Java Switch Statements & Modern Java 14+ Switch Expressions [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Switch Statements & Modern Java 14+ Switch Expressions in real Java systems — specifically: Classic switch with break vs fall-through, suppor...

2.3 Java Loops — for, Enhanced for-each, while, do-while & Labeled break [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Loops — for, Enhanced for-each, while, do-while & Labeled break in real Java systems — specifically: Indexed iteration, read-only enhanced fo...

2.4 Java Arrays — 1D, Multidimensional, Jagged Arrays & java.util.Arrays [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Arrays — 1D, Multidimensional, Jagged Arrays & java.util.Arrays in real Java systems — specifically: Contiguous heap allocation, default zero...

Module 2 Hands-On Capstone Project (100 Marks): Batch Exam Scorecard & 2D Matrix Analytics Engine — Develop a Java console application that stores multi-subject marks for a cohort in a 2D array, computes student-wise percentages, toppers, subject averages, and binary-searches scores using...

MODULE 3 OF 16 • 12 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: Java OOPs Foundations — Classes, Constructors, static & this

Build modular object-oriented Java programs using classes, overloaded constructors, constructor chaining (this()), and static class members.

Objectives: ✓ Design cohesive classes with fields, methods, and constructors • ✓ Chain overloaded constructors cleanly using this(...) • ✓ Distinguish per-instance Heap fields from per-class static Metaspace members

3.1 Java Classes & Objects — Fields, Methods, Object Creation & References [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Classes & Objects — Fields, Methods, Object Creation & References in real Java systems — specifically: Class blueprints, new heap allocation,...

3.2 Constructors in Java — Default, Parameterized, Overloading & Copy Pattern [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Constructors in Java — Default, Parameterized, Overloading & Copy Pattern in real Java systems — specifically: Implicit no-arg constructor rule, c...

3.3 The static Keyword — Static Variables, Methods, Blocks & Nested Classes [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply The static Keyword — Static Variables, Methods, Blocks & Nested Classes in real Java systems — specifically: Class-level state shared across insta...

3.4 The this Keyword — Shadowing Resolution, Constructor Chaining & Fluent APIs [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply The this Keyword — Shadowing Resolution, Constructor Chaining & Fluent APIs in real Java systems — specifically: Disambiguating shadowed fields (t...

Module 3 Hands-On Capstone Project (100 Marks): Automated Student Admission & Unique Roll-Number Generator — Create an OOP student enrollment system where a static sequence counter auto-issues unique student IDs (MAA-1001, MAA-1002...), overloaded constructors support scholarship/regular admissions...

MODULE 4 OF 16 • 15 LIVE + LAB HOURS • 5 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: Java OOPs Pillars — Inheritance, Polymorphism, Abstraction & Encapsulation

Master extends, super(), compile-time overloading vs runtime dynamic method dispatch, abstract classes, Java 8+ interfaces, and immutable encapsulation.

Objectives: ✓ Extend base classes cleanly using super(...) and @Override • ✓ Understand vtable dynamic dispatch & covariant return types • ✓ Design decoupled contracts using interfaces, abstract classes, and encapsulated JavaBeans/Records

4.1 Inheritance in Java — extends, super Keyword, IS-A vs HAS-A Composition [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Inheritance in Java — extends, super Keyword, IS-A vs HAS-A Composition in real Java systems — specifically: Single/multilevel/hierarchical inheri...

4.2 Polymorphism — Method Overloading vs Method Overriding & Dynamic Dispatch [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Polymorphism — Method Overloading vs Method Overriding & Dynamic Dispatch in real Java systems — specifically: Compile-time overloading resolution...

4.3 Abstract Classes & Template Method Pattern in Java [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Abstract Classes & Template Method Pattern in Java in real Java systems — specifically: Partial abstraction (0 to 100%), mixing abstract methods w...

4.4 Java Interfaces — Multiple Implementation, default & static Methods [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Interfaces — Multiple Implementation, default & static Methods in real Java systems — specifically: Contract-driven architecture, implicit pu...

4.5 Encapsulation, Access Modifiers & Immutable Classes in Java [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Encapsulation, Access Modifiers & Immutable Classes in Java in real Java systems — specifically: Four visibility levels (private, default/package-...

Module 4 Hands-On Capstone Project (100 Marks): Polymorphic Payment Gateway (UPI, Card, NetBanking) & Audit Ledger — Architect an extensible Java payment processing engine using a PaymentGateway interface, an AbstractPayment base class for common validation/receipts, concrete UpiPayment and CardPayment sub...

MODULE 5 OF 16 • 13 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: Java Strings, StringBuilder, Regex & Exception Handling

Understand String Constant Pool interning, O(n) StringBuilder mutation, java.util.regex Pattern/Matcher, and Checked vs Unchecked exceptions.

Objectives: ✓ Master String Pool == vs .equals() and StringBuilder performance • ✓ Validate emails, PAN/mobile numbers, and logs with compiled Pattern • ✓ Handle Checked & Unchecked exceptions cleanly with try-with-resources

5.1 Java Strings — Immutability, String Constant Pool & == vs .equals() [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Strings — Immutability, String Constant Pool & == vs .equals() in real Java systems — specifically: Why java.lang.String is final and immutab...

5.2 StringBuilder vs StringBuffer — Mutable Text Manipulation in O(n) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply StringBuilder vs StringBuffer — Mutable Text Manipulation in O(n) in real Java systems — specifically: Dynamic resizable buffer capacity, .append(...

5.3 Java Regular Expressions — Pattern, Matcher & Text Extraction [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Regular Expressions — Pattern, Matcher & Text Extraction in real Java systems — specifically: java.util.regex.Pattern compilation, Matcher.fi...

5.4 Exception Handling — Checked vs Unchecked, try-with-resources & Custom Exceptions [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Exception Handling — Checked vs Unchecked, try-with-resources & Custom Exceptions in real Java systems — specifically: Throwable -> Error vs Excep...

Module 5 Hands-On Capstone Project (100 Marks): Fault-Tolerant KYC Document Validator & Log Masking Engine — Build a Java validation & PII-masking service that validates emails, Indian mobile numbers, and student IDs using compiled Regex Patterns, masks sensitive digits via StringBuilder, and throw...

MODULE 6 OF 16 • 13 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Java Multithreading, Synchronization & Executor Thread Pools

Write safe concurrent Java systems using Thread/Runnable/Callable, thread lifecycle states, synchronized monitors, volatile, and ExecutorService pools.

Objectives: ✓ Prefer Runnable/Callable composition over extending Thread • ✓ Eliminate race conditions & deadlocks via synchronized/locks/AtomicInteger • ✓ Scale concurrent tasks with ExecutorService and Future

6.1 Multithreading & Thread Lifecycle — Runnable, start() vs run() & join() [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Multithreading & Thread Lifecycle — Runnable, start() vs run() & join() in real Java systems — specifically: NEW -> RUNNABLE -> BLOCKED/WAITING/TI...

6.2 Synchronization, Intrinsic Monitors, volatile & Race Condition Prevention [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Synchronization, Intrinsic Monitors, volatile & Race Condition Prevention in real Java systems — specifically: Atomicity & visibility hazards (cou...

6.3 ExecutorService Thread Pools, Callable<V>, Future & CompletableFuture [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply ExecutorService Thread Pools, Callable<V>, Future & CompletableFuture in real Java systems — specifically: Reusing worker threads via Executors.ne...

Module 6 Hands-On Capstone Project (100 Marks): Thread-Safe Flash Sale Seat Booking & ExecutorService Worker Engine — Simulate 200 concurrent student booking requests competing for 25 limited workshop seats using an ExecutorService thread pool, synchronized/ReentrantLock seat allocation, Callable<String> re...

MODULE 7 OF 16 • 11 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: Java File I/O, Buffered Streams & Object Serialization

Process files and binary/character streams efficiently using java.io, java.nio.file.Files, BufferedReader, and Serializable + transient.

Objectives: ✓ Distinguish InputStream/OutputStream (bytes) from Reader/Writer (UTF-8 chars) • ✓ Boost disk I/O throughput 10x+ with BufferedReader/BufferedWriter • ✓ Control object serialization safely with serialVersionUID and transient

7.1 File Handling in Java — File vs java.nio.file.Path & Files API [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply File Handling in Java — File vs java.nio.file.Path & Files API in real Java systems — specifically: Creating, inspecting, reading, and writing fil...

7.2 BufferedReader & BufferedWriter — High-Throughput Character Streaming [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply BufferedReader & BufferedWriter — High-Throughput Character Streaming in real Java systems — specifically: Decorator pattern wrapping Reader/Input...

7.3 Java Serialization — Serializable, serialVersionUID & transient Keyword [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Serialization — Serializable, serialVersionUID & transient Keyword in real Java systems — specifically: Converting object graphs to byte stre...

Module 7 Hands-On Capstone Project (100 Marks): Buffered CSV Attendance Importer & Binary Snapshot Serializer — Create a Java file-processing utility that streams student attendance CSV logs via BufferedReader inside try-with-resources, flags < 75% attendance alerts, and persists binary state snapshot...

MODULE 8 OF 16 • 15 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: Java Collections Framework, HashMap Internals & Generics

Master dynamic lists, hash sets, Java 8+ treeified HashMap internals (hashCode/equals contract), Comparable/Comparator, and bounded Generics.

Objectives: ✓ Choose optimal data structures (ArrayList vs LinkedList, HashSet vs TreeSet) • ✓ Explain HashMap bucket indexing, load factor 0.75, and Red-Black treeification • ✓ Write compile-time safe Generic APIs using PECS (? extends T / ? super T)

8.1 Java ArrayList vs LinkedList — Dynamic Resizing, Fail-Fast Iterators & Complexity [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java ArrayList vs LinkedList — Dynamic Resizing, Fail-Fast Iterators & Complexity in real Java systems — specifically: Object[] backing array, 1.5...

8.2 Java HashSet, LinkedHashSet & TreeSet — Uniqueness & Equals/HashCode Contract [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java HashSet, LinkedHashSet & TreeSet — Uniqueness & Equals/HashCode Contract in real Java systems — specifically: How HashSet delegates to a back...

8.3 Java HashMap Internals — Bucket Hashing, Load Factor 0.75 & Treeification [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java HashMap Internals — Bucket Hashing, Load Factor 0.75 & Treeification in real Java systems — specifically: Node<K,V>[] array, `(n - 1) & hash`...

8.4 Java Generics — Type Parameters, Bounded Wildcards (PECS) & Type Erasure [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Generics — Type Parameters, Bounded Wildcards (PECS) & Type Erasure in real Java systems — specifically: Compile-time type safety without Cla...

Module 8 Hands-On Capstone Project (100 Marks): Generic In-Memory Course Search, Deduplicated Tag Index & Leaderboard Engine — Build an in-memory catalog & student leaderboard engine in Java using HashMap<String, Course> for O(1) code lookup, HashSet<String> for unique skill tags, ArrayList + Comparator chaining for...

MODULE 9 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: Modern Java (Java 8–21) — Lambda Expressions & Stream API

Write concise, declarative data pipelines using @FunctionalInterface, Predicate/Function/Consumer/Supplier, method references, Streams, Collectors, and Optional.

Objectives: ✓ Implement functional interfaces cleanly with lambdas & method references (::) • ✓ Chain lazy intermediate Stream operations (filter, map, flatMap, sorted) with Collectors • ✓ Eliminate NullPointerException chains using Optional<T>

9.1 Java Lambda Expressions, @FunctionalInterface & Method References (::) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Lambda Expressions, @FunctionalInterface & Method References (::) in real Java systems — specifically: Single Abstract Method (SAM) interface...

9.2 Java Stream API, Lazy Evaluation, Collectors & Optional<T> [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java Stream API, Lazy Evaluation, Collectors & Optional<T> in real Java systems — specifically: Lazy intermediate operations (filter, map, flatMap...

Module 9 Hands-On Capstone Project (100 Marks): Declarative E-Commerce Analytics & Grouping Engine with Java Streams — Build an analytical reporting engine over 1,000+ order records using pure Java Stream pipelines: compute category-wise revenue (`Collectors.groupingBy` + `summingDouble`), top 3 customers, p...

MODULE 10 OF 16 • 11 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: Java JDBC Database Connectivity (MySQL), PreparedStatement & Transactions

Connect Java backend services to MySQL/MariaDB safely using JDBC Type-4 drivers, SQL-injection-proof PreparedStatement, batch updates, and ACID transactions.

Objectives: ✓ Configure JDBC URLs, Connection, PreparedStatement, and ResultSet in try-with-resources • ✓ Eliminate SQL injection 100% via ? parameter binding • ✓ Execute atomic multi-table ACID transactions using setAutoCommit(false), commit(), and rollback()

10.1 Java JDBC Architecture & MySQL Database Connectivity [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Java JDBC Architecture & MySQL Database Connectivity in real Java systems — specifically: JDBC API architecture, Type-4 Pure Java MySQL Connector/...

10.2 JDBC PreparedStatement, SQL Injection Defense, Batching & ACID Transactions [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply JDBC PreparedStatement, SQL Injection Defense, Batching & ACID Transactions in real Java systems — specifically: Precompiled parameterized SQL (`?...

Module 10 Hands-On Capstone Project (100 Marks): JDBC + MySQL Student Admission, Fee Ledger & Certificate DAO Layer — Build a clean Data Access Object (DAO) persistence layer in Java 21 connecting to MySQL/MariaDB via JDBC, supporting parameterized student CRUD, batch mock-question insertion (`addBatch`/`ex...

MODULE 11 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Modern Java 17/21 Features: Records, Sealed Types & Pattern Matching

Production-grade Java Training engineering module covering Modern Java 17/21 Features: Records, Sealed Types & Pattern Matching with hands-on labs and capstone build.

Objectives: ✓ Immutable Java Records & Compact Canonical Constructors — Purpose: Validate invariants once at construction and eliminate verbose POJO boilerplate. • ✓ Sealed Interfaces & Exhaustive Switch Pattern Matching — Purpose: Guarantee every transaction state is handled at compile time without default-branch bugs. • ✓ Text Blocks, Optional Chains & java.time UTC APIs — Purpose: Write clean SQL/JSON literals and timezone-safe billing schedules in enterprise Java.

11.1 Immutable Java Records & Compact Canonical Constructors [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Immutable Java Records & Compact Canonical Constructors in real Java systems — specifically: Validate invariants once at construction and eliminat...

11.2 Sealed Interfaces & Exhaustive Switch Pattern Matching [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Sealed Interfaces & Exhaustive Switch Pattern Matching in real Java systems — specifically: Guarantee every transaction state is handled at compil...

11.3 Text Blocks, Optional Chains & java.time UTC APIs [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Text Blocks, Optional Chains & java.time UTC APIs in real Java systems — specifically: Write clean SQL/JSON literals and timezone-safe billing sch...

Module 11 Hands-On Capstone Project (100 Marks): Type-Safe Banking Domain Engine in Modern Java 21 — Design, implement, test, and document 'Type-Safe Banking Domain Engine in Modern Java 21' applying every production concept from Module 11 (Modern Java 17/21 Features: Records, Sealed Types...

MODULE 12 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Java Concurrency, ExecutorService, Locks & Virtual Threads

Production-grade Java Training engineering module covering Java Concurrency, ExecutorService, Locks & Virtual Threads with hands-on labs and capstone build.

Objectives: ✓ ThreadPoolExecutor Sizing, Callable/Future & CompletableFuture — Purpose: Execute parallel I/O tasks with strict timeouts instead of spawning unbounded raw Threads. • ✓ ConcurrentHashMap, AtomicReference & ReentrantReadWriteLock — Purpose: Protect shared in-memory caches from race conditions and deadlocks under heavy load. • ✓ Java 21 Virtual Threads (Project Loom) for Blocking I/O Scale — Purpose: Handle tens of thousands of concurrent socket/JDBC operations with lightweight virtual threads.

12.1 ThreadPoolExecutor Sizing, Callable/Future & CompletableFuture [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply ThreadPoolExecutor Sizing, Callable/Future & CompletableFuture in real Java systems — specifically: Execute parallel I/O tasks with strict timeout...

12.2 ConcurrentHashMap, AtomicReference & ReentrantReadWriteLock [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply ConcurrentHashMap, AtomicReference & ReentrantReadWriteLock in real Java systems — specifically: Protect shared in-memory caches from race conditi...

12.3 Java 21 Virtual Threads (Project Loom) for Blocking I/O Scale [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Java 21 Virtual Threads (Project Loom) for Blocking I/O Scale in real Java systems — specifically: Handle tens of thousands of concurrent socket/J...

Module 12 Hands-On Capstone Project (100 Marks): High-Throughput Multi-Threaded Order Matching Engine — Design, implement, test, and document 'High-Throughput Multi-Threaded Order Matching Engine' applying every production concept from Module 12 (Java Concurrency, ExecutorService, Locks & Virt...

MODULE 13 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

JVM Memory Architecture, Garbage Collection & Performance Tuning

Production-grade Java Training engineering module covering JVM Memory Architecture, Garbage Collection & Performance Tuning with hands-on labs and capstone build.

Objectives: ✓ Stack Frames vs Heap Generations (Eden, Survivor, Old Gen) — Purpose: Prevent OutOfMemoryError leaks caused by unbounded static maps and unclosed streams. • ✓ G1GC & ZGC Tuning Flags plus Heap/Thread Dump Analysis — Purpose: Diagnose CPU spikes and deadlocks rapidly using jcmd, jstack, and Flight Recorder. • ✓ JDBC Connection Pooling (HikariCP), PreparedStatements & Batching — Purpose: Execute parameterized SQL 20x faster while completely blocking SQL injection.

13.1 Stack Frames vs Heap Generations (Eden, Survivor, Old Gen) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Stack Frames vs Heap Generations (Eden, Survivor, Old Gen) in real Java systems — specifically: Prevent OutOfMemoryError leaks caused by unbounded...

13.2 G1GC & ZGC Tuning Flags plus Heap/Thread Dump Analysis [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply G1GC & ZGC Tuning Flags plus Heap/Thread Dump Analysis in real Java systems — specifically: Diagnose CPU spikes and deadlocks rapidly using jcmd,...

13.3 JDBC Connection Pooling (HikariCP), PreparedStatements & Batching [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply JDBC Connection Pooling (HikariCP), PreparedStatements & Batching in real Java systems — specifically: Execute parameterized SQL 20x faster while...

Module 13 Hands-On Capstone Project (100 Marks): Low-Latency JVM Profiler & Leak Diagnostic Lab — Design, implement, test, and document 'Low-Latency JVM Profiler & Leak Diagnostic Lab' applying every production concept from Module 13 (JVM Memory Architecture, Garbage Collection & Perform...

MODULE 14 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

SOLID Design Principles, GoF Patterns, Maven & JUnit 5/Mockito

Production-grade Java Training engineering module covering SOLID Design Principles, GoF Patterns, Maven & JUnit 5/Mockito with hands-on labs and capstone build.

Objectives: ✓ SOLID Principles & Dependency Inversion in Real Java Codebases — Purpose: Keep business rules decoupled so swapping SMS/email/payment providers takes minutes. • ✓ Factory, Builder, Strategy & Observer Patterns in Production — Purpose: Replace sprawling 500-line if/else chains with clean, extensible polymorphic strategies. • ✓ Maven Multi-Module Builds, JUnit 5 Assertions & Mockito Stubbing — Purpose: Verify every Java service edge case automatically in CI before packaging JAR artifacts.

14.1 SOLID Principles & Dependency Inversion in Real Java Codebases [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply SOLID Principles & Dependency Inversion in Real Java Codebases in real Java systems — specifically: Keep business rules decoupled so swapping SMS/...

14.2 Factory, Builder, Strategy & Observer Patterns in Production [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Factory, Builder, Strategy & Observer Patterns in Production in real Java systems — specifically: Replace sprawling 500-line if/else chains with c...

14.3 Maven Multi-Module Builds, JUnit 5 Assertions & Mockito Stubbing [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Maven Multi-Module Builds, JUnit 5 Assertions & Mockito Stubbing in real Java systems — specifically: Verify every Java service edge case automati...

Module 14 Hands-On Capstone Project (100 Marks): Modular Enterprise Invoice & Notification Library with JUnit 5 — Design, implement, test, and document 'Modular Enterprise Invoice & Notification Library with JUnit 5' applying every production concept from Module 14 (SOLID Design Principles, GoF Patterns...

MODULE 15 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Java 21 Modern Language Features: Records, Sealed Types & Virtual Threads

Self-explanatory list-and-bullet production mastery module covering Java 21 Modern Language Features: Records, Sealed Types & Virtual Threads in Java Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ Immutable Java Records, Compact Constructors & Defensive Validation — Purpose: Model domain DTOs and value objects concisely with zero boilerplate mutability bugs. • ✓ Sealed Classes, Exhaustive Switch Pattern Matching & Null-Safe Design — Purpose: Let the Java compiler guarantee every business state transition is handled explicitly. • ✓ Java 21 Virtual Threads (Project Loom) & Structured Concurrency — Purpose: Scale I/O-bound services to tens of thousands of concurrent tasks with simple readable code. • ✓ Build, verify, and document 'High-Concurrency Java 21 Virtual-Thread Order Processor & Pattern-Matched Domain' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + Google Gemini AI Technical Interviews.

15.1 Immutable Java Records, Compact Constructors & Defensive Validation [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Immutable Java Records, Compact Constructors & Defensive Validation in real Java systems — specifically: Model domain DTOs and value objects conci...

15.2 Sealed Classes, Exhaustive Switch Pattern Matching & Null-Safe Design [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Sealed Classes, Exhaustive Switch Pattern Matching & Null-Safe Design in real Java systems — specifically: Let the Java compiler guarantee every b...

15.3 Java 21 Virtual Threads (Project Loom) & Structured Concurrency [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Java 21 Virtual Threads (Project Loom) & Structured Concurrency in real Java systems — specifically: Scale I/O-bound services to tens of thousands...

Module 15 Hands-On Capstone Project (100 Marks): High-Concurrency Java 21 Virtual-Thread Order Processor & Pattern-Matched Domain — Architect, implement, test, and document 'High-Concurrency Java 21 Virtual-Thread Order Processor & Pattern-Matched Domain' applying every bullet-wise engineering concept from Module 15 (Jav...

MODULE 16 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

JVM Internals, JUnit 5 Testing Discipline & Enterprise Java Interviews

Self-explanatory list-and-bullet production mastery module covering JVM Internals, JUnit 5 Testing Discipline & Enterprise Java Interviews in Java Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ ClassLoading, Stack vs Heap Memory, String Pool & Garbage Collection Mechanics — Purpose: Explain how the JVM allocates, interns, and reclaims memory under real enterprise workloads. • ✓ Clean Unit Testing with JUnit 5 Parameterized Tests, Assertions & Test Doubles — Purpose: Verify business rules and boundary exceptions automatically on every Maven/Gradle build. • ✓ Core Java Design Patterns, Concurrency Traps & SDE Interview Readiness — Purpose: Solve equals/hashCode contracts, deadlock prevention, and SOLID refactoring challenges. • ✓ Build, verify, and document 'JUnit 5 + Mockito Verification Suite & JVM GC Diagnostic Lab' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + Google Gemini AI Technical Interviews.

16.1 ClassLoading, Stack vs Heap Memory, String Pool & Garbage Collection Mechanics [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply ClassLoading, Stack vs Heap Memory, String Pool & Garbage Collection Mechanics in real Java systems — specifically: Explain how the JVM allocates,...

16.2 Clean Unit Testing with JUnit 5 Parameterized Tests, Assertions & Test Doubles [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Clean Unit Testing with JUnit 5 Parameterized Tests, Assertions & Test Doubles in real Java systems — specifically: Verify business rules and boun...

16.3 Core Java Design Patterns, Concurrency Traps & SDE Interview Readiness [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Core Java Design Patterns, Concurrency Traps & SDE Interview Readiness in real Java systems — specifically: Solve equals/hashCode contracts, deadl...

Module 16 Hands-On Capstone Project (100 Marks): JUnit 5 + Mockito Verification Suite & JVM GC Diagnostic Lab — Architect, implement, test, and document 'JUnit 5 + Mockito Verification Suite & JVM GC Diagnostic Lab' applying every bullet-wise engineering concept from Module 16 (JVM Internals, JUnit 5...

3. LIVE CLIENT PRODUCTION PROJECT, GLOBAL CERTIFICATIONS & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: CoreBank Concurrent Account Ledger, Multithreaded Batch & JDBC Audit Engine • **Confirmed Role:** Core & Advanced Java 21 Engineering Intern

Domain & Production Infra: BFSI Core Banking & Financial Compliance Client (Java 21 LTS + Virtual Threads + Maven + HikariCP JDBC + JUnit 5 Linux Server)

- Designed clean SOLID domain models using Java 21 records, sealed interfaces, generics, and immutable value objects.
- Implemented thread-safe concurrent settlement workers with ExecutorService, Virtual Threads, and CompletableFuture.
- Built transactional JDBC repositories with prepared statements, HikariCP connection pooling, and batch audit logging.
- Verified business invariants with JUnit 5 test suites and deployed packaged JAR services to live client production.

Aligned Global Vendor Certifications: **1Z0-811** (Oracle Certified Foundations Associate, Java) • **1Z0-830** (Oracle Certified Professional: Java SE 21 Developer) • **1Z0-829** (Oracle Certified Professional: Java SE 17 Developer)

Target Job Roles: Java Software Engineer, Associate Enterprise Backend Developer, Core Java & JDBC Developer, Graduate Engineer Trainee (Java)

Realistic Annual CTC Scope: **Fresher / Entry-Level (0–1 Yr):** ₹4.8 LPA – ₹8.5 LPA **Mid-Level Java Backend Engineer (1–3 Yrs):** ₹9.0 LPA – ₹18.0 LPA **Senior Java Architect / Staff Engineer (3–6+ Yrs):** ₹19.0 LPA – ₹38.0+ LPA

MockAttempt Academy Pvt. Ltd. • Email: support@mockattempt.com • Phone/WhatsApp: +91 98765 43210 • Web: https://mockattempt.com/courses/java-training

Try 2 Free Topic-Wise 50-MCQ Mock Tests & Spoken Gemini AI Interviews daily on every topic above before enrolling.