40-Day Testing Cohort Curriculum

Complete Day-by-Day Instructor Guide

A comprehensive, hands-on curriculum from Java basics to Selenium mastery. Every day includes detailed session plans, activities, GitHub requirements, and LinkedIn posts.

WEEK 1: Java Fundamentals

Days 1-7 | Building Core Programming Foundation

DAY 1

Java Setup, Variables & Data Types

Instructor Checklist

- ✓ Verify JDK 17 installation on all machines
- ✓ Check IntelliJ IDEA Community Edition setup
- ✓ Ensure HelloWorld.java runs successfully
- ✓ Review variable naming conventions
- ✓ Check GitHub repository creation

- ✓ Monitor LinkedIn posts with #Day1 hashtag

Session Plan

Hour 1: Environment Setup & First Program

Install JDK 17 from adoptium.net. Install IntelliJ IDEA Community Edition. Create and run HelloWorld.java. Understand Java file structure and compilation process.

Hour 2: Variables & Data Types Deep Dive

Primitive data types: int, double, boolean, char. String class basics. Variable declaration and initialization. Type casting: implicit and explicit. Naming conventions and best practices. Live coding: VariablesDemo.java.

Hour 3: Hands-on Lab & Project

Mini Project: Student Information System. Store 3 student records with name, age, marks. Calculate average marks. Print formatted output. Pair programming activity. Code review of 3 random students. GitHub setup and push instructions.

Daily Activities

Pair Programming Challenge

Students swap computers and explain each other's variable usage. Instructor walks around listening to explanations.

Debugging Exercise

Fix VariablesBug.java with 5 intentional errors. First 3 correct solutions get recognition.

Live Code Review

Project 3 random students' code and review publicly. Focus on naming conventions and structure.

GitHub Push Requirements

Repository Name: java-basics-day1
Required Files: HelloWorld.java, StudentInfo.java, VariablesDemo.java
Commit Message: "Day 1: Java basics - variables and data types"
Instructor Verification: Check 5 random repositories before session end

LinkedIn Post Template

```
Day 1/40 of End-to-End Testing Cohort!

Today I learned:
- Setting up Java development environment
- Variables and data types in Java
- Created my first Java programs

Built a Student Information System that handles:
- Multiple data types (int, double, boolean, String)
- Basic calculations (average marks)
- Formatted output

Excited to continue this journey!

#Java #SoftwareTesting #LearningToCode #Day1 #TestingCohort

[Screenshot of code output]
```

End-of-Session Instructor Checklist

All students have working Java environment. Minimum 80% completed Student Information System. GitHub repositories created and code pushed. LinkedIn post template understood and shared. Questions addressed and tomorrow's preview given. Attendance recorded with day's achievement notes.

DAY 2

Operators & Control Statements

Instructor Checklist

- ✓ Check operator precedence understanding
- ✓ Monitor if-else logic building skills
- ✓ Review calculator validation logic
- ✓ Verify GitHub commit structure and message quality
- ✓ Check Day 1 LinkedIn posts were made
- ✓ Assess problem-solving approach during activities

Session Plan

Hour 1: Operators Mastery

Arithmetic operators with precedence examples. Relational and logical operators. Assignment operators and shortcuts. Practice: Calculate BMI, age in days. Code-Along: OperatorsDemo.java with all operator types.

Hour 2: Control Flow Statements

Simple if statement and if-else structure. else-if ladder for multiple conditions. Nested if-else statements. Real-world examples: grade calculator, age validator, loan eligibility checker. Live coding: GradeCalculator.java.

Hour 3: Hands-on Project

Mini Project: Advanced Calculator with Validation. Take two numbers and operator as input. Perform calculation based on operator. Validate inputs (no division by zero). Handle invalid operators. Add error messages. Peer testing: Students exchange and test calculators.

Daily Activities

Live Coding Challenge

Build a grade calculator (marks to grade A/B/C/D/F) in 15 minutes. First 5 correct solutions get recognition.

Flowchart Design

Create flowcharts for 3 real-world scenarios: voting age check, loan eligibility, traffic light system. Review 2 flowcharts from each group.

Code Optimization

Given a working but inefficient calculator code, students optimize it. Compare execution time and code readability improvements.

GitHub Push Requirements

Repository Name: java-basics-day2
Required Files: OperatorsDemo.java, GradeCalculator.java, AdvancedCalculator.java
Commit Message: "Day 2: Operators and control flow statements"
Instructor Verification: Check that AdvancedCalculator.java has validation for division by zero

LinkedIn Post Template

```
Day 2/40 Complete!

Today's Focus: Decision Making in Java
```

```
- Mastered all operator types
- Built conditional logic with if-else
- Created a working calculator with input validation

Project Highlight: Advanced Calculator
- Handles all basic operations
- Validates user input
- Prevents errors (division by zero)
- Provides clear error messages

Learning is fun when you build real things!

#JavaProgramming #ControlFlow #Day2Challenge #SoftwareTesting

[Screenshot of calculator output]
```

End-of-Session Instructor Checklist

All students understand operator precedence. Calculators handle division by zero correctly. GitHub repositories contain required three files. Day 2 LinkedIn posts are being prepared. Students can explain their code logic clearly. Tomorrow's topics previewed.

DAY 3 Loops & Pattern Programming

Instructor Checklist

- ✓ Monitor loop initialization and termination conditions
- ✓ Check pattern logic understanding
- ✓ Review infinite loop prevention techniques
- ✓ Verify GitHub commits show incremental progress

- ✓ Check LinkedIn engagement from previous days
- ✓ Assess algorithm thinking development

Session Plan

Hour 1: For Loop Mastery

For loop syntax and structure. Loop control variables and scope. Nested for loops for multi-dimensional iterations. Common patterns: counting, summation, multiplication tables. Practice: Print 1 to 100, even numbers, tables up to 10.

Hour 2: While & Do-While Loops

While loop syntax and use cases. Do-while loop syntax and differences. When to use which loop type. Real-world examples: number guessing game, input validation loops. Practice: User menu systems, password retry mechanisms.

Hour 3: Pattern Programming

Mini Project: Pattern Generator. Right triangle of stars. Pyramid patterns. Number patterns (1, 12, 123). Inverted patterns. Diamond pattern challenge. Advanced: Pascal's triangle, Floyd's triangle. Code optimization for patterns.

GitHub Push Requirements

Repository Name: java-basics-day3
Required Files: LoopsDemo.java, PatternGenerator.java, MultiplicationTable.java
Commit Message: "Day 3: Mastering loops and pattern programming"
Instructor Verification: Check that PatternGenerator.java produces at least 5 different patterns

LinkedIn Post Template

```
Day 3/40: Pattern Programming Master!

Today's Achievement:
- Mastered for, while, and do-while loops
- Created 10+ different patterns
- Built a Pattern Generator program

Projects:
- Multiplication table generator
- Star pattern creator
- Number pyramid builder

Loop concepts are the foundation of automation testing!

#JavaLoops #PatternProgramming #Day3 #CodingJourney

[Screenshot of patterns]
```

DAY 4 Arrays & String Manipulation

Instructor Checklist

- ✓ Verify array indexing understanding (0-based)
- ✓ Check string immutability concept comprehension
- ✓ Monitor array iteration techniques
- ✓ Review GitHub repository organization
- ✓ Check LinkedIn post quality and engagement

- ✓ Assess ability to manipulate complex data structures

Session Plan

Hour 1: Arrays Fundamentals

Array declaration and initialization. Accessing array elements with indexes. Array length property and bounds checking. Iterating through arrays with for loops. Common operations: sum, average, maximum, minimum. Multi-dimensional arrays introduction.

Hour 2: String Methods Deep Dive

String immutability concept. Important methods: length(), charAt(), substring(). String comparison: equals() vs ==, equalsIgnoreCase(). String manipulation: toUpperCase(), toLowerCase(), trim(). Search methods: contains(), startsWith(), endsWith(). split() method for tokenization.

Hour 3: Hands-on Project

Mini Project: Student Management System. Store 10 student names in array. Store marks in parallel array. Find topper (highest marks). Search student by name. Display all students with marks > 75. Calculate class average. Sort students by marks.

GitHub Push Requirements

Repository Name: java-basics-day4
Required Files: ArraysDemo.java, StringMethods.java, StudentManagement.java
Commit Message: "Day 4: Arrays and String manipulation mastery"
Instructor Verification: Check StudentManagement.java has search functionality

DAY 5

Functions/Methods

Instructor Checklist

- ✓ Check method syntax understanding
- ✓ Monitor parameter passing comprehension
- ✓ Review method overloading implementation
- ✓ Verify GitHub commit messages describe method functionality
- ✓ Check LinkedIn posts highlight code reusability
- ✓ Assess ability to break problems into methods

Session Plan

Hour 1: Method Basics

What are methods and why use them? Method syntax: return type, name, parameters. Calling methods and understanding scope. Return statement and its importance. Void vs return methods. Practice: Create 5 simple utility methods.

Hour 2: Method Parameters & Overloading

Passing parameters (pass by value). Multiple parameters and their order. Method overloading concept and benefits. Practical examples of overloading. Best practices for method naming. Variable arguments (varargs) introduction.

Hour 3: Utility Library Project

Mini Project: Comprehensive Utility Library. Math utilities: power, factorial, isPrime, gcd. String utilities: reverse, palindrome check, countVowels, removeDuplicates. Array utilities: sort, search, findMax, findMin, reverseArray. Test all methods with various inputs.

GitHub Push Requirements

Repository Name: java-basics-day5
Required Files: MethodsDemo.java, MathUtils.java, StringUtils.java, ArrayUtils.java
Commit Message: "Day 5: Functions and method overloading"
Instructor Verification: Check each utility class has at least 5 methods

DAY 6 Classes & Objects (OOP Part 1)

Instructor Checklist

- ✓ Verify class vs object understanding
- ✓ Check constructor implementation
- ✓ Monitor object creation and usage
- ✓ Review GitHub for proper class structure
- ✓ Check LinkedIn posts explain OOP concepts
- ✓ Assess real-world modeling ability

Session Plan

Hour 1: OOP Introduction

What is Object-Oriented Programming? Classes vs Objects: blueprint vs instance. Creating a class with attributes and methods. Creating objects using new keyword. Accessing class members with dot notation. Constructor basics and purpose.

Hour 2: Attributes & Methods in Classes

Instance variables vs local variables. Instance methods and their usage. this keyword for current object reference. Creating multiple objects from same class. Constructors: default vs parameterized. Constructor overloading examples.

Hour 3: Library Management System

Mini Project: Library Management System. Create Book class: title, author, ISBN, price, available. Create Member class: name, id, borrowedBooks list. Create Library class to manage books and members. Methods: borrowBook(), returnBook(), displayAvailableBooks(), findBookByTitle(). Simulate complete library operations.

GitHub Push Requirements

Repository Name: java-oop-day6
Required Files: Book.java, Member.java, Library.java, LibraryManagement.java
Commit Message: "Day 6: Classes, objects, and constructors"
Instructor Verification: Check Library class has at least 5 management methods

DAY 7 OOP Advanced & Exception Handling

12

Instructor Checklist

- ✓ Verify encapsulation implementation
- ✓ Check exception handling in real scenarios
- ✓ Monitor getter/setter usage
- ✓ Review GitHub for complete Week 1 portfolio
- ✓ Check LinkedIn summary post for Week 1
- ✓ Assess readiness for testing concepts

Session Plan

Hour 1: Encapsulation & Access Modifiers

Private, public, protected access modifiers. Getters and setters for controlled access. Encapsulation benefits: data hiding and security. Refactor previous projects with encapsulation. Best practices for access control. Real-world examples of encapsulation.

Hour 2: Exception Handling Fundamentals

What are exceptions and why handle them? try-catch block structure. finally block for cleanup. throw vs throws keywords. Common exceptions: ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException. Creating custom exception messages for better debugging.

Hour 3: Robust Calculator Project

Mini Project: Production-Ready Calculator. Calculator class with private variables. Comprehensive getters and setters with validation. All operations with complete error handling. Handle division by zero

with custom messages. Handle invalid inputs (non-numeric). Log all errors and operations. User-friendly error messages.

GitHub Push Requirements

Repository Name: java-advanced-day7
Required Files: Calculator.java, ExceptionDemo.java, SecureLibrary.java
Commit Message: "Day 7: Encapsulation and exception handling mastery"
Instructor Verification: Check Calculator.java handles all exceptions gracefully

Week 1 LinkedIn Summary Post

```
Week 1 Complete: Java Fundamentals Mastered!

7 Days of Intensive Learning:
- Java syntax and environment setup
- Control flow and operators
- Arrays and string manipulation
- Functions and methods
- Object-Oriented Programming
- Exception handling

Final Project: Production-Ready Calculator
- Fully encapsulated design
- Complete error handling
- User-friendly interface
- Robust validation

From zero to building real applications in 7 days!
Excited for Week 2: JUnit & Maven - The testing begins!

#JavaComplete #Week1Done #OOPMastery #TestingCohort

[Screenshot of week's projects]
```

WEEK 2: JUnit & Maven

Days 8-14 | Unit Testing & Build Automation

DAY 8 Introduction to Testing & JUnit Setup

Instructor Checklist

- ✓ Verify Maven installation on all systems
- ✓ Check pom.xml with JUnit dependency
- ✓ Ensure first test runs successfully
- ✓ Review testing mindset development
- ✓ Check GitHub for Maven project structure
- ✓ Monitor LinkedIn posts about testing transition

Session Plan

Hour 1: Software Testing Fundamentals

What is software testing and its importance. Manual vs Automated testing comparison. Types of testing: Unit, Integration, System, End-to-End. Test case vs Test scenario. Introduction to Unit Testing and its benefits. Why JUnit for Java testing?

Hour 2: Maven Setup & First Project

What is Maven? Build automation explained. Install Maven from Apache website. Maven folder structure: src/main/java,

src/test/java. pom.xml introduction and structure. Create first Maven project in IntelliJ. Add JUnit 5 dependency to pom.xml.

Hour 3: First JUnit Test

Create Calculator class in src/main/java. Create CalculatorTest class in src/test/java. Write first test with @Test annotation. Run test from IntelliJ and command line. Understand green (pass) vs red (fail). Test Calculator add() method with positive, negative, zero values.

GitHub Push Requirements

Repository Name: junit-basics-day8
Required Structure: Maven project with src/main/java and src/test/java folders
Required Files: Calculator.java, CalculatorTest.java, pom.xml with JUnit dependency
Commit Message: "Day 8: Maven & JUnit setup with first unit tests"

DAY 9 JUnit Test Annotations & Assertions

Instructor Checklist

- ✓ Check @Test, @BeforeEach, @AfterEach usage
- ✓ Verify assert methods implementation
- ✓ Monitor test naming conventions
- ✓ Review test independence understanding

- ✓ Check GitHub commit messages for tests
- ✓ Assess LinkedIn posts about testing importance

Session Plan

Hour 1: JUnit Annotations Deep Dive

@Test annotation with parameters. Lifecycle annotations: @BeforeEach, @AfterEach, @BeforeAll, @AfterAll. @DisplayName for better test reporting. @Disabled for skipping tests. Test execution order control.

Hour 2: Assertion Methods

assertEquals, assertTrue, assertFalse, assertNull, assertNotNull. assertArrayEquals for array comparisons. assertThrows for exception testing. assertTimeout for performance testing. Custom assertion messages.

Hour 3: Test Data Preparation

Using @BeforeEach for test setup. Creating test data factories. Parameterized tests with @ParameterizedTest. Dynamic test generation. Test data cleanup with @AfterEach.

GitHub Push Requirements

Repository Name: junit-annotations-day9
Required Files: TestAnnotationsDemo.java, AssertionTests.java, ParameterizedTests.java
Commit Message: "Day 9: JUnit annotations and assertion methods mastery"
Instructor Verification: Check at least 5 different assertion types used

DAY 10

Test-Driven Development (TDD)

Instructor Checklist

- ✓ Verify Red-Green-Refactor cycle understanding
- ✓ Monitor TDD mindset adoption
- ✓ Check test-first approach implementation
- ✓ Review test coverage metrics
- ✓ Check GitHub for TDD commit pattern
- ✓ Assess LinkedIn posts about TDD benefits

Session Plan

Hour 1: TDD Fundamentals

What is TDD? Red-Green-Refactor cycle. Benefits of test-first approach. TDD vs traditional testing. Common TDD misconceptions. Real-world TDD examples.

Hour 2: TDD Hands-on

Implement String Calculator kata. Start with simplest test. Incrementally add functionality. Refactor after each green test. Practice: FizzBuzz implementation using TDD.

Hour 3: Advanced TDD Patterns

Test naming conventions in TDD. Test organization strategies. Dealing with external dependencies. Mocking in TDD. Continuous

integration with TDD.

GitHub Push Requirements

Repository Name: tdd-practice-day10
Required Files: StringCalculator.java, StringCalculatorTest.java, FizzBuzz.java, FizzBuzzTest.java
Commit Message: "Day 10: TDD practice with Red-Green-Refactor cycle"
Instructor Verification: Check commit history shows TDD pattern

DAY 11 Mocking with Mockito

Instructor Checklist

- ✓ Verify Mockito dependency added to pom.xml
- ✓ Check @Mock and @InjectMocks understanding
- ✓ Monitor when().thenReturn() usage
- ✓ Review verify() method implementation
- ✓ Check GitHub for mocking examples
- ✓ Assess LinkedIn posts about isolation testing

Session Plan

Hour 1: Mocking Fundamentals

Why mocking? Unit vs Integration testing. Mockito setup and configuration. Creating mock objects with @Mock. Injecting mocks with @InjectMocks. Mockito vs PowerMock.

Hour 2: Mocking Behaviors

Stubbing methods with when().thenReturn(). Stubbing void methods. Argument matchers: any(), eq(), etc. Throwing exceptions from mocks. Chaining stub calls.

Hour 3: Verification & Spy

Verifying interactions with verify(). Verifying method calls with arguments. Verifying call order. Using @Spy for partial mocking. Real-world mocking scenarios.

GitHub Push Requirements

Repository Name: mockito-practice-day11
Required Files: UserService.java, UserServiceTest.java, EmailService.java, EmailServiceTest.java
Commit Message: "Day 11: Mocking with Mockito - isolation testing mastery"
Instructor Verification: Check at least 3 different mocking techniques used

DAY 12 Advanced JUnit Features

Instructor Checklist

- ✓ Check nested test classes usage
- ✓ Verify dynamic tests implementation

- ✓ Monitor test interfaces understanding
- ✓ Review test templates usage
- ✓ Check GitHub for advanced test patterns
- ✓ Assess LinkedIn posts about test organization

Session Plan

Hour 1: Nested & Tagged Tests

@Nested for hierarchical test organization. @Tag for test categorization. Running tests by tags. Custom annotations for tests. Test suites with @Suite.

Hour 2: Dynamic & Repeated Tests

@TestFactory for dynamic tests. @RepeatedTest for repetition. @TestTemplate for test templates. Custom test display names. Test execution conditions.

Hour 3: Test Extensions

JUnit 5 extension model. Creating custom extensions. Parameter resolvers. Exception handling extensions. Test lifecycle callbacks.

GitHub Push Requirements

Repository Name: junit-advanced-day12
Required Files: NestedTests.java, DynamicTests.java, CustomExtensionTest.java
Commit Message: "Day 12: Advanced JUnit features - nested, dynamic, and tagged tests"

Instructor Verification: Check nested test structure implemented

DAY 13

Test Coverage & Reporting

Instructor Checklist

- ✓ Verify JaCoCo setup in pom.xml
- ✓ Check coverage reports generation
- ✓ Monitor coverage metrics understanding
- ✓ Review test report interpretation
- ✓ Check GitHub for coverage reports
- ✓ Assess LinkedIn posts about quality metrics

Session Plan

Hour 1: Coverage Fundamentals

What is test coverage? Line vs Branch vs Path coverage. Coverage metrics interpretation. Coverage tools: JaCoCo setup. Coverage best practices.

Hour 2: JaCoCo Implementation

Adding JaCoCo to Maven project. Configuring coverage thresholds. Generating HTML reports. Excluding classes from coverage. Coverage in CI/CD pipelines.

Hour 3: Test Reporting

Generating test reports with Maven. JUnit XML reports. Custom report generation. Test report analysis. Reporting in team environments.

GitHub Push Requirements

Repository Name: test-coverage-day13
Required Files: pom.xml with JaCoCo, coverage reports, test summary
Commit Message: "Day 13: Test coverage with JaCoCo and reporting implementation"
Instructor Verification: Check coverage reports show minimum 70% coverage

DAY 14 Week 2 Capstone Project

Instructor Checklist

- ✓ Verify comprehensive test suite creation
- ✓ Check TDD approach followed
- ✓ Monitor mocking implementation
- ✓ Review coverage metrics achievement
- ✓ Check GitHub for complete project
- ✓ Assess LinkedIn summary post for Week 2

Session Plan

Hour 1: Project Requirements

Banking System: Account management, transactions, interest calculation. Requirements analysis. Test planning. Architecture design. Setting up project structure.

Hour 2: TDD Implementation

Implement Account class with TDD. Transaction processing tests. Interest calculation tests. Exception handling tests. Refactoring and optimization.

Hour 3: Testing & Reporting

Comprehensive test suite. Mocking external services. Coverage analysis. Report generation. Code review and quality assessment.

GitHub Push Requirements

Repository Name: banking-system-testing
Required Files: Complete banking system with 100% test coverage
Commit Message: "Day 14: Banking system with comprehensive testing suite"
Instructor Verification: Check all business logic has corresponding tests

Week 2 LinkedIn Summary Post

```
Week 2 Complete: JUnit & Testing Mastery!

7 Days of Testing Excellence:
- Maven build automation
- JUnit 5 fundamentals
- Test-Driven Development (TDD)
- Mocking with Mockito
- Test coverage with JaCoCo
- Comprehensive test reporting
```

```
Capstone Project: Banking System
- 100% test coverage
- TDD approach followed
- Mocking external dependencies
- Professional test reports

Ready for API testing in Week 3!

#JUnit #Testing #TDD #Mockito #SoftwareQuality #Week2Complete

[Screenshot of coverage report]
```

WEEK 3: API Testing with RestAssured

Days 15-21 | REST API Testing & Automation

DAY 15 REST API Fundamentals & RestAssured Setup

Instructor Checklist

- ✓ Verify Postman installation
- ✓ Check RestAssured dependency in pom.xml
- ✓ Monitor HTTP methods understanding
- ✓ Review status code knowledge
- ✓ Check GitHub for first API tests
- ✓ Assess LinkedIn posts about API testing

Session Plan

Hour 1: REST API Basics

What is REST? HTTP methods: GET, POST, PUT, DELETE, PATCH. Status codes: 200, 201, 400, 401, 404, 500. Request/response structure. JSON format basics.

Hour 2: Postman Introduction

Postman interface overview. Creating requests. Sending GET/POST requests. Viewing responses. Organizing requests in collections. Using public APIs for practice.

Hour 3: RestAssured Setup

Adding RestAssured to Maven. First RestAssured test. Basic GET request validation. Status code assertion. Response body extraction.

GitHub Push Requirements

Repository Name: restassured-basics-day15

Required Files: FirstAPITest.java, Postman collections
Commit Message: "Day 15: REST API fundamentals and RestAssured setup"
Instructor Verification: Check successful API calls to public endpoints

WEEK 4: Selenium WebDriver Basics

Days 22-28 | Web Automation Fundamentals

DAY 22 Selenium Setup & First Automation

Instructor Checklist

- ✓ Verify Chrome/Firefox installation
- ✓ Check WebDriver setup
- ✓ Monitor Selenium dependency in pom.xml
- ✓ Review browser automation understanding
- ✓ Check GitHub for first Selenium test
- ✓ Assess LinkedIn posts about automation

Session Plan

Hour 1: Selenium Introduction

What is Selenium? Selenium components: WebDriver, Grid, IDE. Browser drivers setup. First WebDriver test. WebDriver lifecycle management.

Hour 2: Locator Strategies

HTML basics review. Locators: ID, Name, ClassName, TagName. CSS Selectors basics. XPath introduction. Finding multiple elements.

Hour 3: First Automation Script

Automate Google search. Enter text in search box. Click search button. Verify results. Take screenshot of results.

GitHub Push Requirements

Repository Name: selenium-basics-day22
Required Files: FirstSeleniumTest.java, GoogleSearchTest.java
Commit Message: "Day 22: Selenium setup and first web automation"
Instructor Verification: Check automation successfully runs and takes screenshot

WEEK 5: Advanced Selenium & Framework Design

Days 29-35 | Page Object Model & Test Framework

DAY 29

Page Object Model (POM) Design Pattern

Instructor Checklist

- ✓ Verify POM concepts understanding
- ✓ Check page class creation
- ✓ Monitor separation of concerns
- ✓ Review reusable component design
- ✓ Check GitHub for POM structure
- ✓ Assess LinkedIn posts about design patterns

Session Plan

Hour 1: POM Fundamentals

What is Page Object Model? Benefits of POM. POM vs non-POM comparison. Best practices in POM. Real-world POM examples.

Hour 2: Creating Page Classes

Login page class design. Home page class. Product page class. Cart page class. Base page class for common elements.

Hour 3: POM Implementation

Implementing POM for e-commerce site. Creating page factory. Writing tests using page objects. Maintaining page classes.

GitHub Push Requirements

Repository Name: pom-framework-day29
Required Files: Complete POM structure for demo site

Commit Message: "Day 29: Page Object Model implementation for e-commerce site"
Instructor Verification: Check at least 4 page classes created

WEEK 6: CI/CD & Final Project

Days 36-40 | DevOps Integration & Capstone

DAY 36 CI/CD with GitHub Actions

Instructor Checklist

- ✓ Verify GitHub account setup
- ✓ Check GitHub Actions understanding
- ✓ Monitor YAML syntax comprehension
- ✓ Review workflow creation
- ✓ Check GitHub for workflow files
- ✓ Assess LinkedIn posts about DevOps

Session Plan

Hour 1: CI/CD Fundamentals

What is CI/CD? Continuous Integration benefits. Continuous Deployment pipeline. GitHub Actions introduction. Workflow

components.

Hour 2: GitHub Actions Setup

Creating workflow files. YAML syntax basics. Setting up Java environment. Running tests in workflow. Artifact management.

Hour 3: Automated Testing Pipeline

Creating test automation workflow. Running Selenium tests. Generating test reports. Sending notifications. Quality gates implementation.

GitHub Push Requirements

Repository Name: ci-cd-pipeline-day36
Required Files: .github/workflows/test-automation.yml
Commit Message: "Day 36: CI/CD pipeline with GitHub Actions for test automation"
Instructor Verification: Check workflow runs successfully on push

DAY 40 Final Capstone Project & Portfolio Presentation

Instructor Checklist

- ✓ Verify complete project implementation
- ✓ Check all testing layers covered
- ✓ Monitor portfolio presentation readiness

- ✓ Review GitHub portfolio organization
- ✓ Check LinkedIn profile optimization
- ✓ Assess final project demonstration

Session Plan

Hour 1: Final Project Completion

Complete e-commerce automation framework. Integrate all testing layers: unit, API, UI. Implement CI/CD pipeline. Generate comprehensive reports. Final code review.

Hour 2: Portfolio Preparation

Organize GitHub repositories. Create README with project overview. Add demo videos/screenshots. Update LinkedIn with projects. Prepare project presentation.

Hour 3: Final Presentations

Student project demonstrations. Code walkthroughs. Testing strategy explanation. Lessons learned sharing. Certification award ceremony.

GitHub Push Requirements

Repository Name: final-capstone-project
Required Files: Complete testing framework with all components
Commit Message: "Day 40: Final capstone project - complete testing automation framework"
Instructor Verification: Check all 40 days of learning integrated into final project

Final LinkedIn Post Template

```
🎉 40-Day Testing Cohort Complete! 🎉

I've successfully completed an intensive 40-day Testing Cohort
journey!

What I've Accomplished:
✅ Java Programming Fundamentals
✅ JUnit & Test-Driven Development
✅ API Testing with RestAssured
✅ Selenium Web Automation
✅ Page Object Model Framework
✅ CI/CD with GitHub Actions
✅ Complete Capstone Project

Final Project: Enterprise Testing Framework
- Multi-layer testing strategy
- 100% automation coverage
- CI/CD pipeline integration
- Comprehensive reporting

Key Skills Developed:
- Automated testing at all levels
- Framework design and implementation
- DevOps integration
- Problem-solving and debugging
- Team collaboration

This journey has transformed me from beginner to testing
professional!

Explore my projects: [GitHub Portfolio Link]
Connect with me for testing opportunities!

#TestingCohort #SoftwareTesting #TestAutomation #Selenium #Java
#CI_CD #CareerGrowth #Day40Complete

[Portfolio screenshot and demo video link]
```

Final Instructor Assessment Checklist

- ✓ All 40 days of content delivered

- ✓ Minimum 80% attendance maintained
- ✓ GitHub portfolio with 40 repositories
- ✓ LinkedIn posts for all major milestones
- ✓ Final project demonstrates all learned skills
- ✓ Students can explain testing concepts clearly
- ✓ Career guidance provided for next steps
- ✓ Certificates of completion distributed

INSTRUCTOR TEAM GUIDE

Essential Guidelines for Successful Delivery

Daily Routine Template

Before Class (15 minutes)

Review previous day's student code. Setup demo environment and test examples. Prepare live coding demonstrations. Check GitHub classroom for submissions.

Session Start (15 minutes)

Quick recap of previous day concepts. Answer questions from students. Show best submissions from previous day. Preview today's learning objectives.

Main Teaching (2 hours)

Follow hour-by-hour session plan. Live coding with students typing along. Stop for questions every 15 minutes. Conduct planned activities and monitor participation.

Session End (30 minutes)

Summarize key learnings of the day. Assign GitHub push task with clear requirements. Show LinkedIn post template. Conduct quick Q&A session. Preview tomorrow's topics.

GitHub Push Strategy

Daily Git Workflow for Students

```
# Day 1 setup
git init
git add .
git commit -m "Day 1: Java basics - variables and data types"
git branch -M main
git remote add origin [repository-url]
git push -u origin main

# Subsequent days
git add .
git commit -m "Day X: [description of today's work]"
git push
```

Repository Naming Convention

- java-basics-dayX (Days 1-7)
- junit-testing-dayX (Days 8-14)
- api-testing-dayX (Days 15-21)
- selenium-automation-dayX (Days 22-28)
- framework-design-dayX (Days 29-35)
- final-project-dayX (Days 36-40)

LinkedIn Post Strategy

Posting Guidelines

Structure: Opening hook, key learnings, project highlight, technical details, inspirational close, hashtags
Hashtags: #Java #SoftwareTesting #[DayNumber] #TestingCohort #[Technology]
Posting Time: After pushing to GitHub, preferably between 9-11 AM or 7-9 PM
Media: Always include screenshot/GIF/video of working code
Engagement: Encourage students to like and comment on each other's posts

Assessment Strategy

Daily Assessment

Code review of 5 random students. Hands-on activity completion check. GitHub commit quality and message. Participation in activities. LinkedIn post completion check.

Weekly Assessment

Capstone project evaluation. Peer code review participation. Presentation skills assessment. Problem-solving approach evaluation. GitHub portfolio completeness.

Final Assessment (Day 40)

Complete capstone project: 40%. Code quality and structure: 20%. Documentation and comments: 20%. Final presentation: 20%.

Weekly Schedule Overview

Week 1: Java Fundamentals

Days 1-7: Core Java programming, OOP concepts, exception handling

Week 2: JUnit & Maven

Days 8-14: Unit testing, TDD, Mockito, test coverage

Week 3: API Testing

Days 15-21: REST API testing, RestAssured, Postman

Week 4: Selenium Basics

Days 22-28: Web automation, locators, basic Selenium

Week 5: Framework Design

Days 29-35: Page Object Model, test frameworks, reporting

Week 6: CI/CD & Final Project

Days 36-40: GitHub Actions, DevOps, capstone project
