Over 35 Years Of Technology Training

Document Generated: 04/26/2026

Learning Style: Virtual Classroom

Technology:

Difficulty: Beginner

Course Duration: 3 Days

Next Course Date: May 11, 2026

Test Driven Development (TDD) and Unit Testing Essentials (TT3503)

About This Course:

Test Driven Development (TDD) and Unit Testing Essentials is a three-day, comprehensive hands-on test-driven development / JUnit / TDD training course

geared for developers who need to get up and running with essential Test-driven development programming skills using JUnit and various open-source testing frameworks. Throughout the course you'll learn, explore and gain practical experience working with best practices for writing great programs in Java, using test-driven development techniques. This course quickly introduces you to the core features and benefits of JUnit. You'll leave this course armed with the skills required to leverage solid test driven development and unit testing techniques, using the latest industry techniques and best practices.

Throughout the course, you will work on a project with labs specifically oriented towards using TDD to implement a complex and multi-faceted web application that uses a database in its final form.

Course Objectives:

Working in a hands-on learning environment, guided by our expert team, you'll explore:

- The role of Unit Testing in software development and testing
- How to write effective Unit Testing
- The properties of effective unit tests
- The benefits of the test-first and Test-Driven Development
- Techniques and practices to aid in the successful adoption of Test-Driven Development
- JUnit and the JUnit Test Runner interface.
- How to use JUnit to drive the implementation of Java code.
- The role of debugging when done in conjunction with tests.
- The fundamentals of the TDD using Java, as well as its importance, uses, strengths and weaknesses.
- Understand how JUnit affects your perspective on development and increases your focus on a task.
- Good JUnit coding style.
- How to Create well-structured JUnit programs.
- How to Compile and execute programs using JUnit and DBUnit
- How to extend testing with mock objects using Mockito.
- Refactoring techniques available to make code as reusable/robust as

possible.

- Various testing techniques.

Audience:

- This programming course is for students with hands-on Java development experience. Attending titles may include Software Developers and Programmers, Agile Practitioners, Quality Assurance Professionals, Software Testers, Product Owners, Project Managers, IT Managers or Software Engineers.

Prerequisites:

Take Before: Students should have development skills at least equivalent to the following course(s) or should have attended as a pre-requisite:

- TT2104 Core Java Programming for OO Experienced Developers – 4 days

Course Outline:

Session: Introducing Test-driven Development

Lesson: Test-Driven Development

- Rationale for TDD
- The process of TDD
- Advantages to TDD
- Side-effects of TDD
- Tools to support TDD
- Tutorial: Setup IntelliJ for Using Maven

Session: Unit Testing using JUnit

Lesson: Unit Testing Fundamentals

- Purpose of Unit Testing

- Good Unit Tests
- Test Stages
- Unit Testing Vs Integration Testing
- Understanding Unit Testing Frameworks

Lesson: Jumpstart: JUnit 5.x

- Understand and work with the features of JUnit
- Write unit tests using @Test annotation
- Test Result Verification (Assertions)
- Manage fixtures using @BeforeEach, @AfterEach, @BeforeAll and @AfterAll annotations
- Maven setup using Surefire plugin
- Lab: Demo JUnit
- Lab: Build JUnit Case Study
- Lab: Jumpstart JUnit

Lesson: Annotations

- Use @DisplayName to specify a custom name for the test
- Check for exceptions thrown by test
- Use @Disabled to prevent a test class or method from running
- Use timeouts to fail test that take longer than required
- Test Execution Order
- Lab: Working with @Test Annotation

Lesson: Hamcrest

- Learn the notation of assertThat
- Know the objective of Hamcrest library

- Use Hamcrest's logical and object matchers
- Use Hamcrest's number and collection matchers
- Lab: Working with Hamcrest

Lesson: Parameterized Tests

- The @ParameterizedTest annotation
- A parameterized test to test code under several conditions
- Define different sources for test data (@ValueSource, @CsvSource, @CsvFileSource,@EnumSource, @MethodSource, @ArgumentSource)
- Lab: Working with Parameterized Tests

Lesson: Advanced Features

- JUnit 4 vs JUnit 5
- Nested Unit Tests
- Repeated Tests
- JUnit Extensions
- ExecutionConditions
- Lambda Support
- Grouped Assertions
- Lab: Working with Advanced Features

Lesson: JUnit Best Practices

- "Good" Tests
- Bad Smell
- White-Box Unit Testing
- Black-Box Unit Testing
- Automation and Coverage

Session: Mocking

Lesson: Mocking of Components

- Why We use Test Dummies
- Working with Mock Objects
- Using Mocks with the User Interface
- Mock Object Strategies

Lesson: Mock Objects and Mockito

- Mockito Description and Features
- Mockito Object Lifecycle
- JUnit 5 and Mockito Dependency Injection
- Stubs Using ArgumentMatchers
- Verifying Behavior in Mockito
- Partial Mock Objects
- The Spy annotation
- Lab: Mock Object and Mockito

Lesson: PowerMock

- PowerMock Description and Features
- Using PowerMockito
- @PrepareForTest
- Mocking a final class or final method
- Mocking a Static Method

Session: Advanced Topics

Lesson: State-based vs. Interaction-based Testing

- State-based Testing

- Interaction-based Testing
- Mock Objects Support Each Approach
- Three Areas to Check in a Test
- Lab: Interaction-based Testing

Lesson: Improving Code Quality Through Refactoring

- Refactoring Overview
- Refactoring and Testing
- Refactoring to Design Patterns
- Lab: Refactoring
- Lab: Best Practices - Refactoring Tests

Lesson: Database Testing: DbUnit

- Setting up DbUnit
- Defining a Dataset File in XML, CSV or Excel
- Writing a DbUnit Test Class
- Assert the results
- Use the FailureHandler and ValueComparer
- Using Date and Time in test sets
- Export a data set
- Lab: Introduction to DbUnit
- Lab: DbUnit Assertions
- Lab: Selenium and DbUnit