Test Driven Development (TDD) and Unit Testing Essentials | Mocking, JUnit, Refactoring, Best Practices & More (TT3503)

Category: Software Development & Programming | **Vendor:** Trivera Tech

Duration: 24.00 hours (3 days) **19.5 CPD Hours** **Rating:** ⭐ 4.6 (5,878 reviews)

Course Information

Delivery Format: Instructor Led - Online

Course Overview

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 ll 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.

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 ll 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.

Who Should Attend

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.

Learning Outcomes

Upon successful completion of this course, participants will be able to:

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.

Detailed Course Outline

Test-Driven Development

- Rationale for TDD
- The process of TDD
- Advantages to TDD
- Side-effects of TDD
- Tools to support TDD
-

Unit Testing Fundamentals

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

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

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

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

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)

Advanced Features

- JUnit 4 vs JUnit 5
- Nested Unit Tests
- Repeated Tests
- JUnit Extensions
- ExecutionConditions
- Lambda Support
- Grouped Assertions

JUnit Best Practices

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

Mocking of Components

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

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

PowerMock

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

State-based vs. Interaction-based Testing

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

Improving Code Quality Through Refactoring

- Refactoring Overview
- Refactoring and Testing
- Refactoring to Design Patterns

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

, 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 OR Using Eclipse Tutorial

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, @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, @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

Mock Objects

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 (OPTIONAL): Selenium and DbUnit

Additional Course Details

Nexus Humans Test Driven Development (TDD) and Unit Testing Essentials | Mocking, JUnit, Refactoring, Best Practices & More (TT3503) training program is a workshop that presents an invigorating mix of sessions, lessons, and masterclasses meticulously crafted to propel your learning expedition forward.

This immersive bootcamp-style experience boasts interactive lectures, hands-on labs, and collaborative hackathons, all strategically designed to fortify fundamental concepts.

Guided by seasoned coaches, each session offers priceless insights and practical skills crucial for honing your expertise. Whether you're stepping into the realm of professional skills or a seasoned professional, this comprehensive course ensures you're equipped with the knowledge and prowess necessary for success.

While we feel this is the best course for the Test Driven Development (TDD) and Unit Testing Essentials | Mocking, JUnit, Refactoring, Best Practices & More (TT3503) course and one of our Top 10 we encourage you to read the course outline to make sure it is the right content for you.

Additionally, private sessions, closed classes or dedicated events are available both live online and at our training centres in Dublin and London, as well as at your offices anywhere in the UK, Ireland or across EMEA.

Frequently Asked Questions

Q: What delivery options are available for Test Driven Development (TDD) and Unit Testing Essentials | Mocking, JUnit, Refactoring, Best Practices & More (TT3503)?

We offer multiple delivery formats:

- Live Instructor-Led Classroom Online (Virtual/Live Online)
- Traditional Instructor-Led Classroom Training (ILT)
- On-site delivery at your offices anywhere in United Kingdom
- Private dedicated courses customized for your team

Q: How many CPD hours does this course provide?

The 3-day Test Driven Development (TDD) and Unit Testing Essentials | Mocking, JUnit, Refactoring, Best Practices & More (TT3503) course provides up to 19.5 CPD hours of structured learning. CPD certificates can be provided upon request.

Q: What is the duration of the Test Driven Development (TDD) and Unit Testing Essentials | Mocking, JUnit, Refactoring, Best Practices & More (TT3503) training?

The training takes place over 3 day(s), with each day lasting approximately 24.00 hours including breaks for lunch and refreshments.

Q: Do you provide corporate training for Test Driven Development (TDD) and Unit Testing Essentials | Mocking, JUnit, Refactoring, Best Practices & More (TT3503)?

Yes, we provide corporate training, dedicated training, and closed classes for Test Driven Development (TDD) and Unit Testing Essentials | Mocking, JUnit, Refactoring, Best Practices & More (TT3503). Training can take place anywhere in United Kingdom including London, Manchester, Birmingham, Edinburgh, or live online allowing teams from across United Kingdom or internationally to attend.

Q: What related terms do people search for?

Popular related searches include: Java; TDD; Unit Testing

Q: Why choose Nexus Human for Test Driven Development (TDD) and Unit Testing Essentials | Mocking, JUnit, Refactoring, Best Practices & More (TT3503)?

Nexus Human is recognized as one of the leading training providers. Our trainers have won multiple awards including:

- Small Firms Best Trainer Award
- National Training Partner of the Year (Ireland) - Multiple Years
- Global Top 30 Instructor Awards (2012, 2019, 2021)
- Tech Excellence Award Nominations
- Learning Performance Institute (LPI) External Training Provider Sponsor 2024

Q: Are there any discount codes available?

Yes! Use discount code **PENPAL5** when booking your Test Driven Development (TDD) and Unit Testing Essentials | Mocking, JUnit, Refactoring, Best Practices & More (TT3503) training. Please note that only one discount code can be used per booking and cannot be combined with other special offers.

Nexus Human

Professional Training & Development

Email: info@nexushuman.com

Website: www.nexushuman.com

Phone: +353 1 889 8700 (Ireland) | +44 20 8142 8961 (UK)

© 2026 Nexus Human. All rights reserved. This brochure was generated on 13/11/2026.