

**Java Testing with JUnit 5**

**Course Number:** JAV-312
**Duration:** 2 days

**Overview**

JUnit 5 is substantively different than JUnit 4. Although the core testing principles are the same, there are substantial implementation differences, important new features, and a brand new extension model.

Accelebrate's Java Testing with JUnit 5 teaches experienced Java developers the fundamentals of unit testing using JUnit 5 and Mockito libraries. Attendees learn how to test enterprise components in the persistence, service, and web layers. Best practices are emphasized and demonstrated throughout this course.

**Prerequisites**

All students must have a good working knowledge of Java and OO, including the use of interfaces, abstract classes, collections, factories, and generics. Experience with Java 8 lambda expressions is helpful, but not required.

**Materials**

All JUnit 5 training students receive comprehensive courseware.

**Software Needed on Each Student PC**

* A recent version of Windows, macOS, or Linux with at least 8 GB RAM
* JDK 8 or later
* Eclipse or IntelliJ IDEA
* Other free software - please contact us if you have purchased this class

**Objectives**

* Understand the new JUnit 5 library structure, role of each component, and how they interact
* Configure IDE projects to run tests natively, and via Maven Surefire
* Write cohesive and effective tests and design classes for testability
* Understand the full test lifecycle, and employ it to configure test fixtures
* Run tests using all available mechanisms: IDE, Maven, JUnit Console Launcher, Launcher API
* Use test discovery and filtering to define and run test plans, including conditional test execution
* Employ naming conventions at every level - test name, classname, display name
* Organize tests with assertion groups and nested tests
* Use test interfaces to apply good OO principles to testing
* Understand the new JUnit extension model, and how to write and use them
* Understand JUnit 4 compatibility and migration
* Use mock objects with Mockito to support isolated testing
* Explore Mockito's facilities for dependency injection of mocks
* Use argument matchers for more generalized testing with mocks
* Implement partial mocking with spies
* Understand the issues in testing enterprise components
* Understand the two basic approaches: standalone testing with mocks, and in-container testing
* Test database access components, using both fakes and an embedded database
* Understand the additional issues involved in testing

**Outline**

* Introduction
* Unit Testing with JUnit 5
	+ Overview
		- Unit Testing and JUnit Overview
		- New Features in JUnit 5
		- JUnit 5 Library Components
		- Naming Conventions and Organizing Tests
	+ Tests and Assertions
		- Writing Test Methods
		- Assertions
		- Assertion Messages
	+ Test Fixtures and Test Lifecycle
		- Creating and Using Text Fixtures
		- Test Run Lifecycle: @BeforeEach and @AfterEach, @BeforeAll and @AfterAll
		- Controlling Test Instances
* Writing and Running Tests (includes a brief primer on Java 8 new features)
	+ Additional Testing Needs
		- Testing for Exceptions
		- Setting Timeouts
		- Assertion Groups
	+ Running Test
		- IDE Support: Eclipse, IntelliJ IDEA
		- Maven Configuration
		- JUnit Platform Console Launcher
		- Launcher API
		- Test Discovery and Selection
		- Display Names
		- Grouping and Filtering with Tags
		- Configuration Parameters
		- Nested Tests
	+ Advanced Capabilities
		- Custom Composed Annotations
		- Inheritance with Test Classes
		- Extensions
		- Conditional Test Execution
		- Parameterized Tests
	+ JUnit 4 Migration
		- The Do-Nothing Case
		- Using a JUnit 4 Runner
		- API Changes
		- JUnit 4 Runners and Rules
		- JUnit 4 Test Suites
	+ Best Practices
		- Testing Void and Private Methods
		- Test Cohesion and Assertion Scope
		- Characteristics of Good Tests
		- Writing Testable Code
		- Testing Anti-Patterns
* Testing with Mocks
	+ Overview
		- Mock Objects as Collaborators
		- Mockito Overview
	+ Creating and Using Mocks
		- Basic Steps in Mocking
		- The Mockito Class
		- Mock Creation with @Mock
		- JUnit 5 MockitoExtension
		- Stubbing
	+ Additional Capabilities
		- Argument Matchers
		- Partial Mocking with Spies
		- Mocking the Unmockable
		- Dependency Injection of Mocks
* Testing Enterprise Components
	+ Overview
		- Unit Testing vs. Integration Testing
		- Testing with Mocks vs. In-Container Testing
		- Mocks vs. Fakes
	+ Testing the Persistence Layer
		- Database Options: Installed, Embedded, Embedded-in-Memory
		- Standalone vs. In-Container Testing
		- Test Independence and Transaction Rollback
		- In-Container Testing with Arquillian [Overview]
	+ Testing Services
		- Similar Issues, Different Layer
		- Working with External Resources
		- The Argument for In-Container Testing
	+ Testing Web Components
		- Interfacing with External Clients
		- Difficulties in Standalone Testing with Mocks
		- Manual vs. Automated Testing
		- Automated Testing with Selenium [Overview]
* Conclusion