

**Acceptance Test-Driven Development, Behavior-Driven Development, and Test-Driven Development Training**

**Course Number:** JAV-303
**Duration:** 3 days

**Overview**

This TDD training course covers Test-Driven Development and Behavior-Driven Development (BDD) concepts of “write your tests first” and “fail first.” Attendees learn how to drive their development process through “test-implement-test pass-refactor” cycles. Students discover the link between Agile Stories and Acceptance Test-Driven Development (ATDD) through the definition of acceptance criteria that can then be implemented via Cucumber-JVM together with JUnit.

Attendees translate scenarios into runnable acceptance tests using Cucumber-JVM. Various testing patterns and approaches will be explored, including utilization of isolated testing via Stubs, Fakes, and Mock objects. Students leverage the evolution of User Stories and Business Flows into code to test web applications via the Selenium API.

**Prerequisites**

All students must have Java experience.

**Materials**

All attendees receive comprehensive courseware covering all topics in the course.

**Software Needed on Each Student PC**

* Windows, macOS, or Linux with at least 8 GB RAM
* A recent JDK version installed
* Related software that Accelebrate will specify in a detailed setup sheet following the purchase of this class

**Objectives**

* Capture the core JUnit syntax
* Use BDD and the test-fail-first approach
* Translate business requirements into user stories with acceptance tests
* Refactor to clean up your code
* Make your code testable
* Use Fakes and Mocks for isolated Unit Tests
* Use Cucumber-JVM to turn User Stores into code
* Leverage the Selenium API

**Outline**

* Introduction
* How Stories Fit into the Agile Process
	+ What is Agile?
	+ The Role of the Product Owner
	+ Criteria for evaluating stories
	+ Requirement analysis
	+ The different types of stories
* JUnit Overview/ Primer
	+ What is Unit Testing?
	+ JUnit, the test lifecycle - @Before, @After, @BeforeClass, @AfterClass, @Test
	+ Writing assert methods – assertEquals, assertSame, assertTrue, assertFalse, assertNull
	+ Hamcrest Matchers and using assertThat
	+ PIT mutation testing
	+ Code coverage
* Test-Driven Development (TDD)
	+ Principals and techniques
	+ TDD Metaphors
	+ Benefits, challenges, and limitations
	+ Handling requirements change
	+ Characteristics of good tests
	+ Revisit anti-patterns
* Testable Designs (Mocks, Fakes and Stubs)
	+ Creating testable code - if you cannot test it what use is it?
	+ Strategies for testable code
	+ Test unfriendly features
	+ Stubs, Fakes, and Mocks
	+ Mocks as Collaborators
	+ Mocks and return values, void methods, frequency calls, and ordering
	+ Using the EasyMock and Mockito testing frameworks
* Behavior-Driven Development (BDD) with Cucumber
	+ Writing User Stories
	+ Acceptance criteria
	+ Acceptance Test-Driven Development (ATDD), specification by example
		- Creating software based on realistic examples,
		- Bridging the communication gap between business stakeholders
	+ Introduction to Cucumber
	+ Writing scenarios with Gherkin
		- Gherkin syntax
		- Using given, when, then
		- Java, Ruby or Groovy Fixtures generated from scenarios
	+ Using JUnit to run your Cucumber stories and scenarios
	+ Writing set-up and tear down code using hooks (@Before and @After)
	+ Using Data Tables, Scenario Outlines
	+ Automate Cucumber steps to drive your application through and below the user interface
	+ Refactor Cucumber step definitions to make them more readable and maintainable
	+ Cucumber Reports
* Testing Websites with Selenium with Cucumber
	+ Using a WebDriver to test web applications and the Selenium API
	+ Interrogating a response page
	+ Simulating links and form submissions
	+ Simulate multiple page navigation
	+ Continuous Integration
* Conclusion