AMBIENT-IT

+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated 05/08/2025

Sign up

JUnit training : Efficient Java testing

2 days (14 hours)

Presentation

Master Java unit testing with JUnit 5 thanks to this comprehensive, operational training course. Designed for developers and quality teams, it will provide you with the essential skills to write, structure and maintain efficient, reliable and automatable tests in your Java projects.

Starting with the fundamentals of testing, test types and JUnit 5 architecture, you'll learn how to write clear, robust tests with key annotations, advanced assertions and execution in different environments.

The course will introduce you to parameterized tests, nested tests, the use of Mockito to simulate dependencies, and best practices for organizing your test code for a maintainable project.

You'll integrate your tests into a CI/CD chain and discover the principles of TDD (Test-Driven Development) with practical workshops on real business cases that can be tested via Spring Boot.

As with all our training courses, this one will be presented with the latest JUnit updates.

Objectives

- Understand the RapidMiner ecosystem and the CRISP-DM lifecycle
- Configure the RapidMiner Studio environment and import multi-source data
- Prepare, transform and enrich datasets for analysis
- Design, train and evaluate predictive and unsupervised models
- Automate analytical workflows with Auto Model and parameter optimization
- Integrate RapidMiner into a reliable, industrialized analytical production process

Target audience

- Java developers
- QA engineers

Prerequisites

- Basic knowledge of the Java language

JUnit : Efficient Java testing

Introduction to Java testing

- Early detection of bugs
- Living documentation
- Regression and secure refactoring
- Unit testing vs. integration testing
- Functional testing and E2E
- Test pyramid and good balancing practices
- Introduction to JUnit
- Alternatives: TestNG, Spock, Mockito, AssertJ

Getting started with JUnit 5

- Jupiter
- Platform
- Vintage
- Creating Maven/Gradle projects with JUnit dependencies
- Simple test example with @Test
- Execution with IDE and command line
- @Test, @BeforeEach, @AfterEach, @BeforeAll, @AfterAll
- @Disabled, @DisplayName
- Using assertions

Writing effective unit tests

- Input data, action, verification

- Readability and clarity
- @ParameterizedTest
- Sources : @ValueSource, @CsvSource, @MethodSource, @EnumSource
- @Nested to structure tests
- Group similar test cases together
- assertAll, assertThrows, assertTimeout
- Comparison with AssertJ for more readable assertions

Duplicate tests

- Why use mocks?
- Concepts: stub vs. mock vs. spy
- Mockito integration with JUnit 5 (@ExtendWith(MockitoExtension.class))
- @Mock, @InjectMocks, when(), verify()
- Test a service dependent on a DAO or web service
- Simulate specific exceptions or behaviors

Test organization and management

- Naming conventions
- Organization by package/module
- Test case isolation
- Help methods
- Fixtures and Builders
- @Tag, conditional execution
- Test groups: unit, integration, slow, fast

Good practices and pitfalls to avoid

- Fast, reliable, readable, isolated, reproducible
- Overly complex testing
- Environment-dependent testing
- Multiple irrelevant assertions
- Tools: JaCoCo, SonarQube
- Measuring = guaranteeing quality

Integration into the development cycle

- Integration with GitHub Actions, GitLab CI, Jenkins

- Failsafe vs Surefire for integration testing
- Introduction to Test-Driven Development
- Cycle: Red? Green? Refactor
- Practical TDD workshop
- Spring Boot : @SpringBootTest, @WebMvcTest
- Repository testing with @DataJpaTest

Final hands-on workshop

- Simple Java application
- Structuring test code
- Nominal and error case coverage
- Service testing with mocking
- REST controller testing with Spring

Companies concerned

This course is aimed at both individuals and companies, large or small, wishing to train their teams in a new advanced IT technology, or to acquire specific business knowledge or modern methods.

Positioning on entry to training

Positioning at the start of training complies with Qualiopi quality criteria. As soon as registration is finalized, the learner receives a self-assessment questionnaire which enables us to assess his or her estimated level of proficiency in different types of technology, as well as his or her expectations and personal objectives for the forthcoming training course, within the limits imposed by the selected format. This questionnaire also enables us to anticipate any connection or security difficulties within the company (intra-company or virtual classroom) which could be problematic for the follow-up and smooth running of the training session.

Teaching methods

Practical course: 60% Practical, 40% Theory. Training material distributed in digital format to all participants.

Organization

The course alternates theoretical input from the trainer, supported by examples, with brainstorming sessions and group work.

Validation

At the end of the session, a multiple-choice questionnaire verifies the correct acquisition of skills.

Certification

A certificate will be awarded to each trainee who completes the training course.