Automated Testing with JUnit and Mockito

Course 5465 – 16 Hours

Overview

It doesn't matter how much time you put into design and how careful you are when programming; mistakes are inevitable. Without automated testing, it is time consuming and difficult to ensure that changes will not break existing code.
Sometimes, you need to simulate the underlying system in order to test it – in this case you might think of mocking, which is replacing subsystem objects with hardcoded objects to ease and simplify testing.
Fortunately, for Java programmers, JUnit makes such testing easy. JUnit makes it fast and easy to set up unit tests for your programs.
JUnit is a regression testing framework. It's an open source software used by the developer who implements unit tests in Java.
Mockito is another open source solution for mocking – so you have it all!
This course introduces testing in general and focuses on how to implement unit testing via JUnit and Mockito.

On Completion, Delegates will be able to

- Understand testing concept
- Understand unit testing
- Use Junit for testing applications
- Understand the roles of JUnit when performing tests
- Understand the need in mocking
- Implement testing mocks via Mockito

Who Should Attend

- Java Developers
- QA testers learning Automation Testing

Prerequisites

- Experience in Java programming

Course Contents

Introduction to Testing

- Goals
- Testing methodologies
- Introduction to TDD and how it is used in Agile
- White-box & black-box testing
- Introduction to unit testing
- Benefits of unit testing
- Testing suites
- Coding & designing for future tests and mocks

Introduction to JUnit

- Usage
- Goals
- The XUnit family
- Why test? Why JUnit?
- Continuous Integration Concepts
- Testing Philosophy

Building Tests and Test Suites via JUnit

- JUnit Mechanics
- The TestCase Super Class
- Overriding the setUp(), tearDown(), testXXX, suite(), main() methods
- Simple Testcase Examples

Methodology

- What should and shouldn't be tested?
- Test independence rule
- Unguaranteed Test execution order
- Using Test Suites
- Standard and Custom Test Runners
- Designing for Testing: Factories, Strategy, Mock Objects
- Test then Refactor

Mocking

- What is mocking?
- Test independence rule
- Unguaranteed Test execution order
- Using Mockito API to define mocks
- Bad practices, do's and don'ts