Test Driven Development (TDD)

Description

Test-Driven Development (TDD) is a design engineering process that relies on a very short development cycle. A TDD approach to software development requires a thorough review of the requirements or design before any functional code is written. The development process is started by writing the test case, then the code to pass the test and then refactoring until completion. Benefits of a TDD approach to software engineering include faster feedback, higher acceptance, reduced scope creep and over-engineering, customer-centric and iterative processes, and modular, flexible, maintainable code.

This 3-day instructor-led course is a deep dive into TDD that incorporates the steps that are necessary for effective implementation. Participants will cover Unit Tests, User Stories, Design, Refactoring, Frameworks, and how to apply them to existing solutions. In addition, this course explores the implications of code dependencies, fluid requirements, and early detection of issues. This is an interactive class with hands-on labs. To get the most out of this course, students are encouraged to fully participate. This course demonstrates the skills developers and teams need for building quality applications sustainably, with quality, for the life of the code base.

Delegates will learn

- Provide knowledge and understanding of Unit Testing principles and practices
- Understand the role of Unit Testing in software development and testing
- Write effective Unit Testing
- Properties of effective unit tests
- How to use mock objects to isolate the “system under test”
- Effective refactoring of the code base
- Benefits of the test-first and Test-Driven Development
- Techniques and practices to aid in the successful adoption of Test-Driven

Development
- How to use Acceptance Testing and Behavior-Driven Development to further advance Test-Driven Development

---

Outline

TDD Basics

- What is TDD?
- Agile Manifesto
- Basic Philosophy of TDD
- TDD Principles
- Purpose of Unit Test
- Test Methods and Types

Principles of Object Oriented Programming (Summary)

- Unit Testing and Tools
- Hello Unit Testing
- Hello Test First Development

Dependency Injection

- What is IoC? What It Does
- Basic DI Principles
- The Importance of DI in Test Code

Mock Concept

- Writing Test Code with Fake Object
- State and Interaction Based Verification
- Stub and Mock Concepts
- Working with Dynamic Mock Libraries
- Testing of external libraries (Db, FileSystem, etc.)

TDD and Refactoring

- Basic Refactoring Techniques
- An existing code becomes testable
- Basic OOP Principles and Impact on Test Code
- Testable Software Design

Quality of Test Code

- Structure of a Good Unit Test
- What Should Be in Test Fixture
- ObjectBuilder and ObjectMother Patterns
- Test Smells
- Unit Test Patterns/Antipatterns

Behavior-Driven Development (BDD)

- Differences with TDD
- Acceptance Test Driven Development
- Specification by Example

Use of Test Tools

- Code Coverage Concept
- Handling Pass and Fail Tests
- Organization of the Test Code
- Test Code Reuse