ICON
AGILITY SERVICES

www.IconAgility.com
(636) 530-7776

Test Driven Development (TDD) Foundations

Test-Driven Development (TDD) helps developers craft and sustain a high-quality code base. TDD complements agile processes such as Scrum particularly, but it can also provide you with numerous benefits if you're not at all agile.

This course is the best place to start for most development teams. This class covers TDD fundamentals, sustainable tests, refactoring/continuous design, test doubles/mocks, and – what every developer wants to learn – how to start rescuing your legacy code.

Learning Objectives:

In this course, you'll obtain a solid foundation for doing test-driven development (TDD). You'll learn:

- How continual attention to design simplifies your work and extends the life of your software
- How TDD encourages continuous design improvement
- How to craft tests that help you understand and control a higher-quality system
- How to deal with troublesome dependencies that can make unit testing tough
- How to tackle the tough challenges of a legacy code base (pretty much every developer's reality)
- Specific concepts and themes for approaching the high risk of a legacy codebase
- How to manage a larger refactoring in a legacy codebase using the Mikado method

Audience:

This course is primarily intended for hands-on developers who have experience in the language being used. While relatively little prior experience in TDD or writing unit tests is assumed, participants should be equipped and prepared to engage in hands-on coding during the course.

Pre-requisites:

Participants should be proficient with the JavaScript programming language and its commonly used libraries and APIs, and at least somewhat familiar with the Eclipse IDE.

System Requirements:

JDK (Java Development Kit version 1.5 or later, Eclipse IDE version 4.0 or later, JUnit version 4.X and JUnit plug-in for Eclipse, Mockito framework version 1.9 or later.

Duration:

3 days

ICON AGILITY SERVICES

www.IconAgility.com
(636) 530-7776

Outline:

1. **Overview of Test-Driven Development**
2. **Basic TDD Technique**
   - Exercise
3. **Your Unit Testing Tool**
4. **TDD and Design**
   - What's the next test?
   - Group exercise
5. **Test Smells**
   - Exercise
6. **Code Katas**
   - Exercise
7. **Refactoring and Design**
8. **Refactoring Drivers**
   - Code smells
   - Simple design
   - Classic design principles
9. **Basic Refactoring**
   - Exercise
10. **Additional Catalog Refactorings**
    - Exercise
11. **Macro Refactoring**
    - Backing into tests
    - Optional exercise
12. **Basic "Mock" Technique**
    - Test double variants
    - Implementing Test Doubles
    - Mock injection techniques
    - Exercise
13. **Additional Test Double Topics**
    - Using mock tools
    - Exercise
    - Challenges/best practices
14. **Legacy Code Challenges**
    - Legacy themes
    - Dependency-breaking techniques
    - 2-3 exercises
    - The Mikado Method
    - Optional exercise
15. **Acceptance Tests (ATs) and TDD**
16. **Sustaining and Succeeding with TDD**