

## **Test Driven Development**

### **What is Test-Driven Development?**

- What is it?
- Why do I want to adopt it?
- What are the benefits of test-driven development?
- What are the costs?
- What outcome should I expect when I adopt it?

### **Testing Concepts**

- Test Plans, Test Cases, and Test Suites
- Unit Testing
- Integration Testing
- Continuous Testing
- Robustness Testing
- Acceptance Testing

### **Test-Driven Development Concepts**

- Test-driven development
- Where to begin
- Test-driven development patterns
- Test-driven development best-practices

### **Testing Tools**

- Unit and Integration Testing with JUnit and Mock Objects
- Automated testing with ANT and Maven
- Continuous Testing with CruiseControl, ANT, and Maven
- Robustness Testing with DBUnit, HTTPUnit, JMeter
- Code coverage analysis

### **Setting up the Testing Infrastructure**

- Defining the test strategy
- Picking the tools
- Configuring the IDE
- Configuring the build system
- Reporting

### **JUnit**

- What is JUnit
- Creating unit tests
- Creating integration tests
- Assertions

## Mock Objects

- What they are
- When and how to use them effectively
- Creating mocks with jMock and EasyMock
- Replacing Mock Objects with Live Objects

## Refactoring

- What is refactoring?
- Refactoring concepts and best-practices
- Identifying and implementing potential refactorings
- Testing refactored code

## Testing Legacy Code

- What do to with legacy code
- Common approaches for legacy code testing
- Exploring dependencies, creating loose couplings