Sela.

TDDC2

Test Driven Development for C++ Developers

college@sela.co.il 03-6176666

Test Driven Development for C++ Developers

TDDC2 - Version: 1

3 days Course

Description:

Bring your development team up to par with some of the most productive and successful software development teams in the world. Companies such as Google, Microsoft and that BBC have already realized the potential that lies in Test Driven Development. It's time you do the same for your team. Use TDD techniques to reduce the amount of bugs in your software, make your developers more productive and your software more stable.

Notes:

1.This course is very technical and includes extensive hands-on practice. However, the real world is always more complex than the course examples, and therefore it is strongly recommended to add two additional consulting days for guided practice on the your own code base, and an ongoing periodic accompaniment.

2.Typically the tools used in this course are Visual Studio, Google Tests, Google Mock and Resharper C++, but we can use any testing framework for C++ of your choice (e.g. CppUnit) and any mocking framework, IDE or productivity Add-on.

Intended audience:

This course is intended for Developers, architects and team leaders

Prerequisites:

At least one year experience developing software using C++

Objectives:

Know how to write unit tests
Know the process of TDD
Understand the need for mock objects and know how to use them
Understand the benefits of TDD
Understand the relationships between clean code and good design principles with TDD
Learn some refactoring techniques and best practices
Understand how TDD fits into the software development lifecycle
Understand the ATDD (Acceptance Test Driven Development) methodology and its benefits

Topics:

Introduction to unit tests

- Types of automated tests
- Advantages and limitations of automated tests
- Advantages and limitations of Unit tests

Testing Framework overview (Google Test)

- Writing a simple test without a testing framework
- Advantages of a unit test framework
- Test initialization and cleanup lifecycle
- Asserts

Designing and structuring a test

- Designing the tests as scientific experiments
- One claim per test
- Arrange-Act-Assert
- Given-When-Then
- Hands-on practice

Test Driven Development

- The Red-Green-Refactor process
- The benefits of writing the tests first
- The benefits of writing the tests first
- Hands-on practice

Breaking dependencies using mock objects

- Why do we need mock objects?
- Creating manual mock objects
- Using a mocking framework (Google Mock)

Clean code and Design for Testability

- Avoiding static variables and Singletons
- The SOLID principle
- The 4 rules of Simple Design

Refactoring techniques and best practices

- Taking small steps
- Create before delete
- Useful refactoring transformations (e.g. Extract method, Extract interface, inline method, etc.)

- Using Resharper C++ for refactoring and code analysis (optional)

ATDD - Acceptance Test Driven Development

- Overview on ATDD
- How ATDD boosts the development lifecycle
- Relationships between TDD and ATDD