SE012CS - *Defensive Programming in C#*

DURATION: 3 Days; Instructor-led

WHAT YOU WILL LEARN

Defensive programming is sometimes referred to as secure programming by computer scientists who state this approach minimizes bugs. It is a form of defensive design intended to ensure the continuing function of a piece of software under unforeseen circumstances.

In this course audience will learn how to adopt defensive programming techniques to improve software and source code, in terms of:

- General quality - reducing the number of software bugs and problems.
- Making the source code comprehensible - the source code should be readable and understandable so it is approved in a code audit.
- Making the software behave in a predictable manner despite unexpected inputs or user actions.

The course ended with a daylong case study to enforce the audience understanding of the subject matters.

AUDIENCE

This course should be useful for novice and expert C# developers. It is an invaluable training for the developers who are serious about quality in coding.

PREREQUISITES:

- Basic knowledge and skill in C#.
- At least 3 years of experience in software development.

METHODOLOGY

This program will be conducted with interactive lectures, PowerPoint presentations, discussions, practical exercises and Case Study

COURSE OUTLINES

Module 1 - Introduction

- What is Defensive Programming?
- Clean Code
- Testable Code and Unit Tests
- Predictable Code

Module 2 – Defending Your Methods - Part 1

- Clean, Testable, and Predictable Methods
- Example Clean, Testable, and Predictable Methods
- Demo Creating a Class Library Component
- Demo Clean, Testable, and Predictable Methods
- Demo Named Arguments

Module 3 – Defending Your Methods - Part 2

- Validating Method Parameters
- Demo Validating Method Parameters
- Demo Method Overloading

Module 4 – Automated Code Testing

- Why automated testing?
- Code First vs. Test First
- Defining Unit Test Cases
- Creating Unit Tests
- Using Test Explorer
- Generating Unit Tests
- Unit Tests and Exceptions
- Dependencies

Module 5 - Returning Predictable Results

- Method Results
- Demo Returning a Value
- Demo Returning Exceptions
- Demo Returning Multiple Values
- Returning Null

Module 6 – Defending Your Code Constructs

- Local Variable Declarations
- If Statements
- Switch Statements
- Enums
- Casting

Module 7 - Asserts, Errors, and Exceptions

- Demo Preparing the Sample Code
- Asserts
- Anticipated Errors
- Unexpected Exceptions and a Global Exception Handler
- Exception Handling

Module 8 – Design by Contract

- Introduction
- Tolerant Vs Demanding coding styles
- Pre-Conditions
- Post-Conditions
- Class Invariant
- Loop Invariant

Module 9 – Case Study

- 8-Puzzle