

Software Design patterns and Refactoring methods

Course 407 – 40 Hours

Overview

A Design Pattern is a catalogued solution that has been applied and tested in multiple situations to produce well-designed, reusable, object-oriented software. Designing for reusability is an art, typically acquired after many years of software development, refining and iterating over designed software modules. In this course, each pattern session will start with theoretical understanding followed by practical use. The design patterns will be described using Intent, Motivation, Sample Code, Applicability, Structure, Consequences and its Known Uses. The students will also test their understanding by completing a practical assignment for several very popular design patterns.

Course Objectives

Upon completion students should be able to:

- Understand the usage of design patterns in the real world
- Understand how to read UML based description of design pattern
- Implement design pattern using popular OOP languages such as C++ and C#
- Use existing design patterns that are implement on standard libraries of C++ and .Net Framework

Who Should Attend

This course is intended for programmers, team leaders, and software project managers.

Course Contents

UML Diagram

- Class Diagrams
- Sequence Diagrams

Design Principles

- The Open-Closed Principle
- The Liskov Substitution Principle & Design by Contract
- Single Responsibility Principle (optional)
- Dependency Inversion Principle (optional)

Design Patterns

- Singleton
- Proxy

- Visitor
- Acyclic Visitor (optional)
- Template Method
- Abstract Factory
- Prototype
- Command
- Composite
- Decorator
- Strategy
- Mediator (optional)
- Observer (optional)

Refactoring methods

- Use refactoring to improve design
- Refactor existing applications to make them more maintainable
- Use tests during refactoring