SE011CPP: *Design Pattern in C++*

DURATION: 5 Days; Instructor-led

WHAT YOU WILL LEARN

Concept of reusability in software development today is not restricted to just code reuse as before. Reuse the software design is getting popular and crucial in deliver high quality and cost-effective software is desired by many organizations. Reusing the good practices and experiences in problem solving formally will gain more advantages in many ways.

The course is to introduce basic concepts in software design patterns in general, and covers entire set of the GoF patterns using latest version of C++ language.

Upon completion of this program, participants should be able to:

- Appreciate what the Software Pattern is about
- Realize the important of basic principles in software engineering
- Understand the important of Software patterns in software development
- Know various types of patterns and use them in the right context
- Apply C++ to implement patterns

AUDIENCE

This is not the language course. It requires students with at least basic programming skill in C++. It provides a starting point for those who are interested in apply formal software design patterns in their software development process.

PREREQUISITES

- At least 3 years' experience in software development and preferably involve in software design before.
- Basic knowledge and skill in C++ programming.

METHODOLOGY

This program will be conducted with interactive lectures, PowerPoint presentation, discussions and practical exercise

COURSE OUTLINES

Module 1 - Introduction to Design Pattern

- Why do we need Design Pattern?
- A bit of the history
- Good software practices and principles
- The Gang of Four (GoF)
- Overview of the Design Patterns
- Essential software engineering disciplines
- How patterns are documented?

Module 2 – Understand Design Pattern Representation in UML

- Class
- Abstract Class
- Aggregation & Composition
- Multiplicity
- Association
- Role
- Attributes
- Operations and Methods

Module 3 – Creational Patterns in brief

- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton

Module 4 - Structural Patterns in brief

- Adapter
- Bridge
- Composite
- Decorator
- Façade
- Flyweight
- Proxy

Module 5 – Behavioral Patterns in brief

- Chain of Responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template Method
- Visitor

Module 6 – Essential C++ features for design patterns

- Inheritance
- Polymorphism
- Abstract Classed and Methods
- Interfaces
- Access Modifiers

Module 7 - The Abstract Factory Pattern

- How a Simple Factory Works?
- Building the Simple Factory
- Using the Factory
- Consequences of Abstract Factory
- Example in C++

Module 8 - The Builder Pattern

- How a Builder Works?
- Consequences of the Builder Pattern
- Example in C++

Module 9 - The Factory Method
- When to Use a Factory Method?
- Example in C++

Module 10 - The Prototype Pattern
- Using the Prototype
- Deep Vs Shallow copy
- Cloning the Class
- Using the Prototype Pattern
- Dissimilar Classes with the Same Interface
- Prototype Managers
- Consequences of the Prototype Pattern
- Example in C++

Module 11 - The Singleton Pattern
- Creating Singleton Using a Static Method
- Exceptions and Instances
- Throwing the Exception
- Creating an Instance of the Class
- Providing a Global Point of Access to a Singleton
- Example in C++

Module 12 - The Adapter Pattern
- Making an Adapter
- The Class Adapter
- Two-Way Adapters
- Pluggable Adapters
- Example in C++

Module 13 - The Bridge Pattern
- Extending the Bridge
- Windows Forms as Bridges
- Consequences of the Bridge Pattern
- Example in C++

Module 14 - The Composite Pattern
- Basic understanding of GC based system
- An Implementation of a Composite
- Doubly Linked Lists
- Consequences of the Composite Pattern
- A Simple Composite
- Example in C++

Module 15 - The Decorator Pattern
- Handling events in a Decorator
- Layout Considerations
- Multiple Decorators
- Decorators, Adapters, and Composites
- Example in C++

Module 16 - The Façade Pattern
- What Constitutes the Façade?
- Consequences of the Façade
- Example in C++

Module 17 - The Flyweight Pattern
- Sharable Objects
- Copy-on-Write Objects
- Intrinsic and extrinsic
- Example in C++

Module 18 - The Proxy Pattern
- Copy-on-Write
- Comparison with Related Patterns
- Example in C++

Module 19 - Chain of Responsibility
- Applicability
- Consequences of the Chain of Responsibility
- Example in C++

Module 20 - The Command Pattern
- Motivation
- Command Objects
- Building Command Objects
- Consequences of the Command Pattern
- Example in C++

Module 21 - The Interpreter Pattern
- Motivation
- Applicability
- State machine
- Objects Used in Parsing
- Regular Expression
- Example in C++

Module 22 - The Iterator Pattern
- Motivation
- Consequences of the Iterator Pattern
- Iterators in C++ collections classes
- Example in C++

Module 23 - The Mediator Pattern
- Interactions Between Controls
- Initialization of the System
- Mediators and Command Objects
- Consequences of the Mediator Pattern
- Single Interface Mediators
- Implementation Issues
- Example in C++

Module 24 - The Memento Pattern
- Motivation
- Implementation
- Providing Undo
- Command Objects in the User Interface
- Example in C++

Module 25 - The Observer Pattern
- "Don't call me, I will call you" model
- Consequences of the Observer Pattern
- Event-Handler model in .NET
- Example in C++

Module 26 - The State Pattern
- Switching Between States
- How the Mediator Interacts with the State Manager?
- Consequences of the State Pattern
- Example in C++

Module 27 - The Strategy Pattern
- Motivation
- The Context
- The Program Commands
- Example in C++

Module 28 - The Template Method Pattern
- Motivation
- Kinds of Methods in a Template Class
- Templates and Callbacks
- Summary and Consequences
- Example in C++

Module 29 - The Visitor Pattern
- Motivation
- When to Use the Visitor Pattern
- Visiting the Classes
- Visiting Several Classes
- Catch-All Operations with Visitors
- Double Dispatching Issue
- Traversing a Series of Classes
- Consequences of the Visitor Pattern
- Example in C++

Module 30 - Relationship between design patterns
- Overall structure of the design pattern catalogue
- Classification of relationships
- Categories of relationships
- Modifying relationships and design patterns