

This course explores proven real-world techniques to meet the biggest challenge in the software development community - building quality systems which fulfill your requirements, and delivering them on time. The focus of the course is to give you the practical skills that are most critical in building well designed software systems. Written exercises are used throughout the course to enhance your understanding of the principles discussed during the lectures. This course explores the most common object-oriented design patterns (Gang of Four) and how to use these patterns to develop solid, robust, and reusable software development applications.

The course covers the patterns in the three core areas of Creational, Structural, and Behavioral and is hands-on with design projects and programming labs. A basic understanding of Java is beneficial to understanding the code samples presented throughout the course. There are no coding exercises presented in this course.

Course Objectives:

- Explore and understand basic Design Pattern concepts.
- Use Design Patterns effectively to build robust, well designed, reusable systems.
- Gain familiarity with the GOF Patterns.

Audience: Application developers, programmers, system designers, and project manager developers who need to improve the systems development through the use of design patterns.

Prerequisites: Some experience in object-oriented thinking/programming, professional experience with object-oriented technologies and UML diagrams, and a basic knowledge of Java.

Number of Days: 2 days

- | | |
|---|---|
| <p>1. Introduction
 What's our World?
 OK – So Just What is a Design Pattern?
 Design Patterns are not Esoteric
 Why Use Patterns?
 The Adapter Pattern
 Reviewing Interfaces & Abstract Classes
 Interface Types
 Interface Definitions
 Abstract Methods
 Abstract Classes
 Using Abstract Classes
 Important Principal of OO Design</p> | <p>Another Design for Collection Traversal
 Using Our New Collection
 Differences in Traversing Our Collection
 Why is This Important?
 Why is This a Design Pattern?
 We Will Expand on Our Design</p> |
| <p>2. The Iterator Pattern
 Patterns: Traversing a Collection
 A Simple ArrayList
 Using Our ArrayList
 Using Our Simple Collection</p> | <p>3. Design Patterns – Background
 Design Patterns Arise From Architecture
 Christopher Alexander
 The TimelessWay
 A Core Principle of His Books
 Patterns in <i>A Pattern Language</i>
 Sitting Circle (185)
 Different Chairs (251)
 Patterns Evolution in Software
 OOPSLA 88</p> |

- Patterns Evolution in Software
- Patterns Today
- 4. **UML Overview**
 - Unified Modeling Language (UML)
 - Using UML
 - UML Diagrams
 - Class Diagram
 - Class Diagram Notation
 - Association Relationships in Detail
 - Class Diagram Notation
 - Abstract Class Notation
 - Interface Notation
 - Another Class Diagram
- 5. **Gang of Four Design Patterns**
 - Description**
 - What Do We Know Now About Patterns
 - GOF Pattern Description
 - Iterator: Overview
 - Iterator: Motivation
 - Iterator: Applicability
 - Iterator: Structure – Java
 - Iterator: Structure – General
 - Iterator: Participants
 - Iterator: Collaborations and Consequences
 - Iterator: Implementation
 - Implementation: Who Controls the Iteration
 - Implementation: Who Defines the Traversal
 - Implementation: Robustness
 - Iterator: Known Uses and Related Patterns
 - So – What is a Design Pattern?
- 6. **The GOF Patterns Catalog**
 - Organizing the Catalog
 - Creational, Structural, and Behavioral Purpose
 - Class and Object Scope
 - Design Pattern Space
 - The GOF Catalog of Design Patterns
- 7. **Factory Method Pattern**
 - Motivation – Forces and Solution Motivation
 - Factor Method: Iterator Usage
 - Factory Method: General Structure
- 8. **Strategy Pattern**
 - Participants
 - Collaborations and Applicability
 - Applicability
 - Consequences
 - Implementation
 - Known Uses and Related Patterns
- 9. **Decorator Pattern**
 - Motivation – Forces and Solution Structure
 - Alternative to Strategy
 - How Do We Choose Among Alternative?
 - Participants
 - Collaborations and Applicability
 - Consequences
 - Implementation
 - Known Uses and Related Patterns
 - Difference From Factory Method
- 10. **Composite Pattern**
 - Motivation – Forces
 - Motivation – Solution Structure
 - Participants
 - Collaborations
 - Consequences
 - Implementation
 - Known Uses and Related patterns
- 11. **Template Method Pattern**
 - Motivation – Forces and Solution Structure
 - Participants and Collaborations
 - Consequences

- Implementation
- Known Uses and Related Patterns
- 12. **Command Pattern**
- Motivation – Forces and Solution
- Structure
- Participants and Collaborations
- Consequences
- Implementation
- Undo and Redo
- Known Uses
- 13. **Chain of Responsibility Pattern**
- Motivation – Forces
- Motivation – Solution
- Structure
- Participants and Collaborations
- Consequences/Applicability
- Implementation
- Known Uses and Related Patterns
- 14. **Facade Pattern**
- Motivation – Forces and Solution
- Structure
- Participants and Collaborations
- Consequences/Applicability
- Implementation
- Known Uses
- 15. **Patterns for Enterprise Systems**
- Meeting the Challenge – Technologies
- Meeting the Challenge – Best Practices
- Some Patterns for Enterprise Systems
- Business Delegate
- Business Delegate: Solution
- Business Delegate: Structure
- Business Delegate: Consequences
- Value Object
- Value Object: Solution
- Value Object: Structure
- Value Object: Consequences
- Data Access Object (DAO)
- DAO: Solution
- DAO: Structure
- DAO: Consequences
- Lazy Load
- Lazy Load: Solution
- Lazy Load: Consequences
- 16. **Wrap-Up**
- What Have We Done?

So – What Do You Think About
Patterns?
Where Do We Go From Here?
Do We Fit Into Alexander’s
Vision?
Design Patterns Isn’t All You
Need
Have Fun