Comprehensive Design Patterns

Object-Oriented Analysis and Design (OOAD) is a process of identifying the needs of a software project and then laying out those specifications in a readable model. Design Patterns are an extension of these skills that create more maintainable and robust code. Using well-known, proven patterns that either stand-alone or build from one to the next, designers are able to further define the specifications of the project, optimizing quality and time spent on developing the project by programmers.

This advanced Object-Oriented course provides software architects and designers with skills to create high quality object-oriented designs exhibiting improved flexibility, reduced maintenance costs, and with increased understanding of the resulting code. Participants learn more than 30 object-oriented patterns, including the 23 micro-architectures in *"Design Patterns: Elements of Reusable Object-Oriented Software"*, by Gamma, Helm, Johnson, and Vlissides (the gang-of-four, or GoF book). Application examples and code snippets are provided to illustrate the patterns and the rationale for using that pattern in a given situation.

Course Objectives:

- Improve Software Architecture.
- Build Design Pattern Vocabulary.
- Be able to discuss trade-offs in applying various design patterns.
- Gain concepts and tools for writing better object-oriented code.
- Gain concepts for better documenting object-oriented code.
- Review relevant UML notation.

Audience: Software architects and designers requiring advanced design skills.

Prerequisites: *Object-Oriented Analysis & Design with the Unified Modeling Language* or equivalent experience. At least 6 months experience programming with an object-oriented programming language.

Number of Days: 4 days

1. Course Introduction
- Course Objectives
- Overview
- Suggested References

2. Design Pattern Overview
- Objectives in Software Design/Module Design
- Overview of Patterns
- Qualities of a Pattern
- Pattern Systems
- Heuristics vs. Patterns

3. Principles of Object-Oriented Design
- Overview of Principles
- Single-Responsibility Principle (SRP)
- Open-Closed Principle (OCP)
- Tell vs. Ask
- Command/Query Separation (CQS)
- Composed Method
- Combined Method
- Liskov Substitution Principle (LSP)
- Dependency Inversion Principle (DIP)
- Interface Segregation Principle (ISP)
- Law of Demeter

4. Principles of Package Architecture
- Package Cohesion Principles
- Package Coupling Principles
- Martin Package Metrics

5. Basic Object-Oriented Design Patterns
- Delegation vs. Inheritance

© Batky-Howell, LLC