

# OBJECT TECHNOLOGIES

*Revised 5/29/2008*

***/training/etc***

*The Art of Knowledge.*

**This Page Intentionally Left Blank**

# Table of Contents

---

**NotAssigned**

OOAD Using the Unified Modeling Language (UML).....	1
Implementing Design Patterns.....	2

This Page Intentionally Left Blank

---

**Course Description:** This course uses the industry-standard Unified Modeling Language (UML) as a means of depicting OO software design and providing team members with a common notation and vocabulary for communicating their ideas. Topics include: use case diagrams, class diagrams, interaction diagrams, state diagrams, implementation diagrams, the UML process, and advanced modeling concepts.

**Who Should Attend:** This course is designed for analysts, technical managers, and software developers who need a common, practical technique for describing object-oriented systems.

**Prerequisites:** Experience in analysis, design, or development is desirable.

**Benefits of Attendance:** Upon completion of this course, students will be able to:

- Explain the difference between object and procedure orientation.
- Understand object-oriented concepts such as encapsulation, inheritance, and polymorphism.
- Compare object-oriented analysis with other approaches to systems analysis.
- Explain the origins of the Unified Modeling Language.
- Use common features of UML diagrams.
- Describe and compare features of UML tools.
- Identify actors and use cases.
- Write use case descriptions.
- Create use case diagrams.
- Identify classes, associations, aggregations, and multiplicity.
- Construct class diagrams at various levels of detail.
- Create sequence and collaboration diagrams.
- Identify object states and substates.
- Draw state diagrams and activity diagrams.
- Use component and deployment diagrams.
- Understand general characteristics of UML processes.
- Describe the object-oriented software life cycle.
- Use basic object-oriented design and project metrics.

### Course Outline:

#### Object-Oriented Analysis and Design

What is OOAD?  
Approaches to System Analysis  
Object-Oriented Methodologies  
History of UML  
What is UML?  
Models and Architectural Views  
Common Features of UML Diagrams  
Characteristics of a UML Process

#### Object-Oriented Concepts

What is Object-Oriented?  
What is an Object?  
Encapsulation  
Class vs. Object  
Inheritance  
Multiple Inheritance  
Polymorphism  
Object Orientation vs. Procedure Orientation

#### Using a UML Tool

Introduction  
Selecting a UML Tool

#### Identifying Use Cases

General Steps in a UML-Based Process  
Use Cases  
Actors  
Use Case Diagrams  
Use Case Description  
Use Case Template  
Use Case Relationships  
Use Case Diagram for Elevator System  
Business Modeling

#### Discovering Classes

Class Diagrams  
Class Diagram Details  
Class Stereotypes  
Discovering Classes  
Candidate Classes

#### Associations

Associations  
Association Roles  
Multiplicity  
Aggregation  
Inheritance  
Classes for Elevator System

Class Diagram for Elevator System  
Association Classes  
Building the Static Model

#### Design Patterns

What is a Design Pattern?  
Reasons to Study Design Patterns  
History of Design Patterns  
Cataloging Design Patterns  
Design Pattern "Themes"  
The Singleton Pattern  
The Composite Pattern  
The Iterator Pattern  
The Strategy Pattern

#### Interaction Diagrams

Building the Dynamic Model  
Interaction Diagrams  
Sequence Diagrams  
Messages  
Lifelines  
Activations  
Sequence Diagram - Example  
Sequence Diagram for Elevator System  
Collaboration Diagrams  
Message Labels  
Collaboration Diagram - Example  
Collaboration Diagram for Elevator System

#### State Diagrams

States and Events  
State Diagrams  
Guard Conditions  
State Diagram Details  
Substates  
Concurrent Substates  
Disjoint Substates - Nesting  
Disjoint Substates - Layering  
The History Pseudostate  
State Diagrams for Elevator System  
Digital Clock / Timer

#### Activity Diagrams

Refining the Models  
Activity Diagrams  
Activity Diagram Details  
Activity Diagram - Example  
Activity Diagram with Swimlanes  
Describing a Business Process

#### Behavioral Patterns

The Observer Pattern  
Observer Pattern - Collaboration Diagram  
Observer Pattern - Sequence Diagram  
The Memento Pattern  
Memento Pattern - Sequence Diagram

#### Implementation Diagrams

Moving Towards Implementation  
Logical vs. Physical Architecture  
Hardware and Software Concepts  
Component Diagrams  
Deployment Diagrams  
Allocating Components to Nodes

#### Object-Oriented Software Life Cycle

The Object-Oriented Life Cycle  
The Iterative Process  
Estimating Object-Oriented Projects  
Object-Oriented Design Metrics  
Refactoring  
Steps Toward Reuse

#### UML 2.0

What's New in UML 2.0  
Package Diagram  
Composite Structure Diagram  
Interaction Overview Diagram  
Use Case Multiplicities  
Exception Handling Notation  
Object Constraint Language

#### Appendix A: Analysis and Design

Problems  
Online Survey Application  
Diagram Editor  
Budget Manager  
Intelligent Parcel Scale  
Weather Monitoring Station  
Recycling Machine  
Subway System  
Additional Problems

#### Appendix B: Reference

Abbreviations and Acronyms  
Internet Resources

#### Appendix C: Class Diagrams - Advanced Features

Qualified Associations  
N-ary Associations  
Constraints  
Derived Elements  
Parameterized Classes

#### Appendix D: AntiPatterns

What is an AntiPattern?  
Reasons to Study AntiPatterns  
Functional Decomposition  
The Blob  
Poltergeists

#### Appendix E: EclipseUML

EclipseUML  
EclipseUML Screen Layout  
EclipseUML Menu Bar  
The Eclipse Workspace  
Getting Started  
Use Case Diagrams  
Class Diagrams  
Sequence Diagrams  
Collaboration Diagrams  
State Diagrams  
Activity Diagrams  
Deployment Diagrams

#### Appendix F: ArgoUML

ArgoUML  
ArgoUML Screen Layout  
ArgoUML Menu Bar  
File Formats Used by ArgoUML  
Getting Started  
Use Case Diagrams  
Class Diagrams  
Sequence Diagrams  
Collaboration Diagrams  
State Diagrams  
Activity Diagrams  
Deployment Diagrams

**Course Description:** Design Patterns are proven solutions to recurring problems in object-oriented software systems. This course covers sixteen design patterns and includes detailed programming exercises to allow students to practice implementing selected patterns.

**Who Should Attend:** This course is for system architects, designers and programmers working on or preparing for a software project using an object-oriented design.

**Prerequisites:** Programming experience in Java or C++ and some familiarity with object-oriented concepts is required.

**Benefits of Attendance:** Upon completion of this course, students will be able to:

- Describe the purpose of design patterns.
- Understand the ways that design patterns are documented and classified.
- Use the Singleton Pattern to provide controlled access to the sole instance of a class.
- Use the Composite Pattern to represent whole-part hierarchies of objects.
- Use the Factory Method Pattern to eliminate the need to 'hard-code' specific class names.
- Use the Observer Pattern to minimize coupling between domain and interface objects.
- Use the Template Method Pattern to implement the common parts of an operation.
- Use the Strategy Pattern to configure a class with one of many alternate behaviors.
- Use the Iterator Pattern to separate the traversal mechanism from an aggregate object and to support concurrent traversals on the same object.
- Use other creational patterns to help make systems independent of how its objects are created.

### Course Outline:

#### Introduction

What is a Design Pattern?  
Reasons to study Design Patterns  
History of Design Patterns  
Cataloging Design Patterns  
Patterns covered in this course  
Design Patterns "Themes"

#### The Singleton Pattern

Overview  
C++ Implementation  
Java Implementation  
Java API Example

#### The Composite Pattern

Overview  
Structure  
C++ Implementation  
Java Implementation  
Java API Example  
C++ Source Code Example  
Java Source Code Example

#### The Factory Method Pattern

Overview  
Structure  
C++ Implementation  
Java Implementation  
Java API Example  
C++ Source Code Example  
Java Source Code Example

#### The Observer Pattern

Overview  
Structure  
C++ Implementation  
Java Implementation  
Java API Example  
C++ Source Code Example  
Java Source Code Example

#### The Template Method Pattern

Overview  
Structure  
C++ Implementation  
Java Implementation  
Java API Example  
C++ Source Code Example  
Java Source Code Example

#### The Iterator Pattern

Overview  
Structure  
C++ Implementation  
Java Implementation  
C++ Standard Template Library  
Java API Example  
C++ Source Code Example  
Java Source Code Example

#### The Strategy Pattern

Overview  
Structure  
C++ Implementation  
Java Implementation  
Java API Example  
Java Source Code Example

#### Other Creational Patterns

The Abstract Factory Pattern  
The Prototype Pattern  
The Builder Pattern  
Creational Patterns - Summary

#### Other Structural Patterns

The Adapter Pattern  
The Bridge Pattern  
Decorator Pattern  
Structural Patterns - Summary

#### Other Behavioral Patterns

The Memento Pattern  
The Chain of Responsibility Pattern  
The Visitor Pattern  
Behavioral Patterns - Summary

#### Appendix A: References And Web Resources

#### Appendix B: Exercises - C++

#### Appendix C: Exercises - Java

#### Appendix D: Antipatterns