Course 611 – Object-Oriented Analysis and Design Using UML

Duration: 4 days

You Will Learn How To

- Capture user requirements in use cases and transform them into detailed designs
- Exploit the rich object-oriented modeling provided by Unified Modeling Language (UML)
- Adapt to changing requirements with iterative techniques and component-based design
- Design solutions optimized for modern object-oriented languages and platforms
- Apply proven design patterns to refine analysis and design models

Course Benefits

Object-Oriented Analysis and Design (OOAD) is the principal industry-proven method for developing reliable, modular, testable programs and systems. This OOAD training course provides practical skills in the latest OO requirements gathering, analysis, design, and testing methods. Intensive hands-on exercises offer you a working knowledge that turns concepts into practice. Students will not just learn UML diagrams. They will learn how to apply UML in the context of OO software development. From requirements gathering to actually mapping the diagrams to code, concepts in this course are consistently illustrated using a single case study. This course emphasizes on applying different design patterns to solve common OO design issues. Knowing how to create properly designed object-oriented systems is what separates programmers from software architects.

Who Should Attend

Anyone involved in developing systems on modern object-oriented platforms. Project teams benefit greatly by sharing the same methodology with co-developers or with supportive management. Familiarity with an Object-Oriented language is assumed.

Course Content

Introduction to Object-Oriented Analysis and Design

- What is Object-Oriented Analysis and Design (OOAD)?
- Introduction and Overview of Unified Modeling Language (UML)

Iterative Development

- Waterfall vs Iterative Development
- Iterative Time Boxing
- Unified Process and Unified Modeling Language
- Phases of the Unified Process

Use Cases

- Requirements Types
- Use Case Scenarios
- When and How to Write Use Cases
- Kinds of Use Cases
- Use Case Diagrams

Domain Models

- What is a Domain Model?

- Guidelines in Domain Model Creation
- Guidelines in Finding Conceptual Classes
- Adding Associations
- Adding Attributes

System Sequence Diagrams

- What are System Sequence Diagrams?
- Naming System Events and Operations

Operation Contracts

- What are Operation Contracts?
- Sections of a Contract
- Writing Post-Conditions

Logical Architecture

- Using Layers
- Avoiding Lower to Higher Coupling
- UML Package Diagram
- The Facade Design Pattern

Object Design

- Static Modeling
- Dynamic Modeling

Interaction Diagrams

- UML Interaction Diagrams
- Sequence Diagrams
- Communication Diagrams
- Lifeline Boxes and Lifelines
- Messages
- Focus of Control
- Return Values
- "this" calls
- Creation of Instances
- Frames
- Looping
- Conditional Messages
- Nested Frames
- Showing Related Interaction Diagrams
- Invoking Static Methods
- Asynchronous and Synchronous Calls

Class Diagrams

- The Class Box
- Attributes
- Note Symbols
- Operations and Methods
- Generalization, Abstract Classes and Methods
- Dependency
- Interfaces
- Composition vs. Aggregation
- Association Classes
- User-defined compartments
- Active Classes

GRASP: Designing Objects with Responsibilities

- The GRASP Patterns
- Creator
- Expert
- Low Coupling
- Controller

Mapping Designs to Code

- Creating Class Definitions from DCDs
- Implementing Methods from Interaction Diagrams
- Determining Order of Implementation

More GRASP Patterns

- Iteration-2 Requirements
- Polymorphism
- Pure Fabrication

Applying GoF Design Patterns

- Adapter
- Factory
- Singleton
- Strategy

Activity Diagrams and Modeling

- UML Activity Diagrams
- Applications of Activity Diagrams
- Common Symbols

State Machine Diagrams

- UML State Machine Diagram
- Events, States, and Transitions
- Applying State Machine Diagrams
- Modelling
- Transition Actions and Guards
- Nested States

Domain Model Refinement

- Iteration-3 Requirements
- Generalization
- Abstract Classes

More SSDs and Contracts

- New System Sequence Diagrams
- New System Operations
- How to Partition the Domain Model

Logical Architecture Refinement

- Common Layers in an Information System
- Inter-Layer and Inter-Package Coupling
- Inter-Layer and Inter-Package Interaction Scenarios
- System Operations and Layers
- UI Facade

More GoF Patterns

- The Proxy Design Pattern
- Exceptions in a Class, and Interaction Diagrams
- Abstract Factory Pattern

Deployment Diagrams

- Nodes
- EENs as Property Strings
- Communication Paths
- Artifacts
- Node Instances

About ActiveLearning, Inc.

ActiveLearning is the Philippines' leading provider of Information Technology and Project Management education, where thousands of students take courses from Application Development to Project Management to Network Security, and much more. Our courses are taught by expert instructors, and learning is enhanced through a blend of in-depth lectures, workshops, and hands-on exercises.