Object-Oriented Analysis and Design (OOA&D) is a process of identifying the needs of a software project and laying out specifications in various models. The Unified Process, in which four iterative phases are used to hone the architecture and build the system, is the framework for many OOA&D endeavors. Each phase can consist of models from different perspectives. These models are documented using the Unified Modeling Language, an industry-standard language for visualizing, specifying, and documenting the architecture of the system.

In this course, students learn how to identify and design objects, classes, and their relationships to each other, which includes links, associations, and inheritance. A strong emphasis is placed on UML diagram notation for use cases, class and object representation, links and associations, and object messages. This course utilizes UML 2.0 notation.

Course Objectives:

- Apply the principals and practices of Object-Oriented Analysis and Design.
- Use modeling in analysis and design, particularly in visual modeling.
- Use the Unified Modeling Language to create visual models of business problems and software solutions.
- Design programs with objects.
- Create more flexible and more maintainable software systems at lower costs.

Audience: Analysts, designers, and programmers responsible for applying OO techniques in their software engineering projects.

Prerequisites: Familiarity with structured techniques such as functional decomposition is helpful.

Number of Days: 5 days

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

2. Introduction to Analysis and Design
- Why is Programming Hard?
- The Tasks of Software Development
- Modules
- Models
- Modeling
- Perspective
- Objects
- Change
- New Paradigms

3. Objects
- Encapsulation
- Abstraction
- Objects
- Classes
- Responsibilities
- Attributes
- Composite Classes
- Operations and Methods
- Visibility
- Inheritance
- Inheritance Example
- Protected and Package Visibility
- Scope
- Class Scope

4. Advanced Objects
- Constructors & Destructors
- Instance Creation
- Abstract Classes
- Polymorphism