

OO-60  
2-Days

## Introduction to OO Design and Development

*This course teaches the practicalities of best practice OO design and development techniques. It is aimed at software designers and developers having experience in non-OO techniques and who need to convert quickly to OO environments. The focus of the course is on the identification of Objects from the business perspective and how these Objects are designed and implemented using OO languages such as Java and C#.*

*The course uses an introductory level of UML to describe OO design. Tools such as Enterprise Architect are demonstrated. Key topics such as testing within an OO environment, Robustness Analysis and OO interfacing to relational databases are also covered.*

*At the end of the course delegates will be able to understand and create OO designs, read and produce basic UML diagrams, understand Design Patterns, understand the terminology of OO and contribute to OO design and development.*

*The course is supported by many practical exercises and complete solutions*

### Course Objectives

- Explain the origins of OO
- Show how OO fits into system scope definitions
- Show how Objects relate to the business world
- Show how Objects are necessary for modern application implementations
- Explain how OO implementations simplify the maintenance and support tasks
- Present the basic techniques of OO design
- Show how Robustness Analysis uncovers hidden classes
- Explain how CRC cards are used to capture OO concepts.
- Explain the use of Use Cases in developing OO models
- Explain how OO modelling is related to ER modelling
- Show how Requirements Capture is used with Use Cases to identify candidate Objects
- Explain the features of OO languages in relation to procedural languages

### Course Objectives

- Show how to develop and use Class diagrams
- Present the "code behind the scenes" of OO applications
- Show how Objects are derived from Project Requirements
- Present the end-to-end OO Development lifecycle
- Explain how (some) Objects are accessed / stored into Relational databases
- Explain the practicalities of OO Development environments
- Show how OO implementation affects the testing process.
- Present Interfaces as a design technique

Extensive time is set aside for discussions, review of solutions and demonstrations.

### Audience

- Development or Support Staff currently working in non-OO environment
- OO Developers wishing to firm up their understanding of OO fundamentals
- Technically-oriented Management wanting to understand OO techniques

### Prerequisites

Some background in software development

### Timetable

Register at 09:00 for 09:30 start.  
Finish at 17:00.

### Presentation Style

Lectures, demonstrations, exercises, hands-on use of OO tools and group discussions.

### Dates and Venues

Refer to *Course Schedules*.

The course covers:

### Introduction

- 📁 Objects and Programming
- 📁 Relation of Objects to conventional programming
- 📁 Techniques and issues in dealing with Objects

### Modelling

- 📁 What is Modelling?
- 📁 Why Model?
- 📁 Business Modelling
- 📁 Business Models and Objects
- 📁 The Static models
- 📁 The Dynamic Models
- 📁 How to use CRC
- 📁 Capturing the results
- 📁 Business Modelling and State Diagrams / Workflow

### What is a Use Case

- 📁 Business Use Cases
- 📁 Business Process Diagrams
- 📁 Extracting and identifying Use Cases from Business Use Cases
- 📁 (System) Use Cases
- 📁 Supplementary Specifications
- 📁 Actor-identification through the Life Cycle
- 📁 The Use Case Module
- 📁 Identifying Business Rules

### Writing Powerful Use Cases

- 📁 Use case elements
- 📁 Actor inheritance
- 📁 The Use Case audience
- 📁 The Use Case Glossary
- 📁 Use Case packaging
- 📁 The use of Pre and Post-requisite conditions
- 📁 Inheritance, <<extends>> and <<include>>
- 📁 Scenarios
- 📁 Success / Failure categories
- 📁 Use Case sampling
- 📁 Linking Use Cases to Class diagrams

### Object Capture

- 📁 Deriving Objects from Use Cases
- 📁 Types of Object, based on purpose and attributes
- 📁 Developing an Object description from multiple Use Cases
- 📁 Verifying Object interfaces by Role Play
- 📁 Show how Robustness Analysis helps model the architecture and identify hidden classes

### OO Languages

- 📁 OO Languages - some history
- 📁 Current languages C#, Java
- 📁 Development Environments
- 📁 Relationship between OO Design and OO languages
- 📁 Code skeletons
- 📁 Forward Engineering
- 📁 Reverse Engineering
- 📁 Tools and Support

### Object Testing

- 📁 OO and Testing - advantages and disadvantages
- 📁 Planning Tests for Objects
- 📁 Test-Driven Development techniques
- 📁 Managing the results
- 📁 Test Patterns - JUnit and NUnit
- 📁 How this works in Your Environment

### Databases and Multi-Tier Applications

- 📁 Architecture Overview
- 📁 Designing for Relational Databases
- 📁 Solution Choices
- 📁 Abstract Factory Patterns

### Summary

- 📁 Main Points
- 📁 Next Steps