

Price: R8,600.00 excl. VAT  
Duration: 5 days  
Code: OOADN

# Object-Oriented Analysis & Design using UML

## Description

The Object-Oriented Analysis and Design course focuses on the analysis, design and documentation of object-oriented systems using established OO methodologies and the UML. Delegates write use cases, develop activity, class and sequence diagrams, and participate in CRC sessions. Object-oriented concepts are explained and illustrated using Java. Design principles and patterns are introduced.

## Objectives

Delegates who complete the OO Analysis and Design course using UML will be able to:

- Understand object-oriented concepts and the object-oriented project lifecycle.
- Write effective use cases.
- Design a system using classes and class relationships.
- Develop high-level class diagrams and be able to apply the MVC paradigm to class design.
- Develop sequence and collaboration diagrams.
- Understand object-oriented design principles and design patterns.

## Intended Audience

The Object-Oriented Analysis and Design course is intended for business analysts, system architects, developers and programmers who need to analyse, design and develop object-oriented systems, and who need to use the Unified Modelling Language (UML).

## Prerequisites

Knowledge of an object-oriented language is recommended, although not essential.

## Course Contents

*The lecturer reserves the right to modify the contents of the course to suit the needs of the delegates.*

**Introduction** • The evolution of the object-oriented paradigm. • OOP compared to other programming paradigms. • Advantages and disadvantages of OOP.

**Object-Oriented Concepts and Terminology** • Classes and objects. • Attributes and behaviours. • Data abstraction and encapsulation. • Polymorphism. • Inheritance and code reuse. • Associations and relationships between classes.

**Unified Modelling Language** • History and evolution of the UML. • UML diagrams: use case, class, object, sequence, collaboration, state, activity, component, package, timing, subsystem, model, deployment diagrams. • Common extension mechanisms. • UML modelling tools.

**Object-Oriented Methodologies** • Traditional Software Development Lifecycle. • Iterative and incremental development. • The need for an OOAD process. • The Rational Unified Approach (RUP). • The Iconix method. • Extreme Programming.

**Object-Oriented Analysis** • Behaviour analysis and use cases. • Activity diagrams. • CRC cards. • Domain modelling. • Class identification and domain classes. • Which UML diagrams to use during analysis.

**Object-Oriented Design** • Responsibility driven design. • Class design and detailed class diagrams. • Robustness diagrams. • Sequence and state diagrams. • Which UML diagrams to use during design.

**Design Principles** • What makes a design feel "right". • Advanced design principles.

**Design Patterns** • Design pattern concepts. • Examples of commonly used patterns.