

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

Object Oriented Implementation

Description

The Object-Oriented Implementation Course covers the entire implementation of a small software system, from inception, through use case modelling, class and interaction diagram development to coding and testing. Emphasis is placed on the implementation aspects of the system, including design pattern usage and process issues.

Objectives

Delegates who complete the Object-Oriented Implementation course will be able to:

- Become familiar with the entire OOAD software development process.
- Identify and use design patterns correctly.
- Identify process and implementation issues.
- Implement unit and functional testing.

Intended Audience

The Object-Oriented Implementation course is ideal for experienced object-oriented programmers who require more advanced knowledge of design patterns and process implementation issues.

Prerequisites

Knowledge of an OO language such as Java, C++, C# or VB.NET is essential, as well as either our Object-Oriented Analysis and Design Course or experience in OO analysis and design.

Course Contents

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

Revision of OO Concepts and Methodologies. • Classes and objects. • Attributes and behaviours. • Data encapsulation. • Polymorphism. • Overloading and overriding. • Inheritance and interfacing. • Composition and aggregation. • OO methodologies such as RUP, Iconix and Extreme Programming. • Iterative, incremental development. • Inception, Elaboration, Construction, Transition phases. • Robustness analysis and the MVC paradigm.

UML Revision • Use case diagrams and use case text. • Activity diagram. • Class and object diagrams. • Sequence and collaboration diagrams. • State and timing diagrams. • Component and deployment diagrams. • Package, model and subsystem diagrams. • Visual modelling tools.

Inception Phase • Analysis of proposed system using domain modelling, use cases, class diagrams, CRC sessions, activity diagrams.

Elaboration Phase • Preliminary and detailed design of proposed system using class and object diagrams, sequence diagrams, state diagrams. • Use of use case templates.

Construction Phase • Detailed design and implementation. • Code development from class, interaction and state diagrams. • Design pattern usage. • Unit and functional testing.

Concurrency and Persistence • Threading and synchronization. • Locks, semaphores and mutexes. • Persistence using serialization, flat files and databases. • Relational vs object oriented databases. • Mapping objects to tables. • Transactions. • Scalability.

Implementation Issues • Exception handling. • Memory management. • Debugging. • Coding styles. • Frameworks. • Reusable components: ActiveX, JavaBeans, EJBs. • Distributed components: COM/DCOM, CORBA, RMI, Web Services.