

Price: R15,800.00 excl. VAT
Duration: 5 days
Delivery: Virtual classroom or
on-site training

Design Patterns

Description

A design pattern is a proven way to solve a common design problem. Design patterns improve code, because they provide a tested solution, and make it easier to maintain the code.

This course shows you how to recognise recurring problems in object-oriented code, choose an appropriate pattern, and apply it without adding unnecessary complexity. You learn when patterns make code easier to understand and maintain — and when they make it worse.

The Design Patterns course is language-neutral and is intended for programmers working in object-oriented environments.

Objectives

After you have completed the Design Patterns course, you will be able to:

- Recognise recurring design problems and select appropriate design patterns to address them.
- Apply common GoF patterns in a way that improves clarity, flexibility, and maintainability.
- Identify misuse and overuse of patterns that introduce unnecessary complexity.
- Explain and compare creational, structural, and behavioural patterns, and understand the trade-offs between them.
- Evaluate existing code and refactor it using appropriate design patterns where beneficial.

Intended Audience

This course is useful if:

- You are an experienced object-oriented programmer and want to improve the structure and maintainability of your code.
- You work on systems where changes are becoming harder and debugging is taking longer.
- You are a systems architect who needs a shared vocabulary for discussing design decisions and trade-offs.
- You need to understand not only common design patterns, but also when patterns introduce unnecessary complexity.
-

Prerequisites

Before you attend the Design Patterns course:

- You should have practical experience programming in an object-oriented language such as Java, C++ or C#.
- You should already understand core object-oriented principles, including classes, inheritance, interfaces, and basic design concepts.
- You should have either attended our Object-Oriented Analysis and Design Course or have equivalent experience in object-oriented analysis and design.

Course Contents

Object-Oriented Foundations (Refresher)

- Responsibilities, boundaries and collaboration.
- Encapsulation and managing coupling.
- Inheritance vs composition.
- Interfaces and abstractions.
- Common design weaknesses that lead to rigid code.

Design Principles

- Design heuristics and trade-offs.
- SOLID principles in practice.
- Package cohesion and coupling.
- MVC and separation of concerns.
- Styles, patterns and idioms — what's the difference?

Understanding Design Patterns

- Patterns as reusable design decisions.
- Recognising recurring problems.
- GoF classifications: creational, structural, behavioural.
- Patterns vs principles.
- When patterns increase complexity.

Anatomy of a Pattern

- Intent and applicability.
- Structure and participants.
- Collaborations and consequences.
- Implementation considerations and trade-offs.

Creational Patterns

- Abstract Factory.
- Builder.
- Factory Method.
- Prototype.
- Singleton (benefits and pitfalls).

Structural Patterns

- Adapter.
- Bridge.
- Composite.
- Decorator.
- Facade.

- Flyweight.
- Proxy.

Behavioural Patterns

- Chain of Responsibility.
- Command.
- Interpreter (overview).
- Iterator.
- Mediator.
- Memento.
- Observer.
- State.
- Strategy.
- Template Method.
- Visitor.

Practical Application

- Refactoring existing code using appropriate patterns.
- Recognising anti-patterns and over-engineering.
- Evaluating pattern trade-offs in real systems.

Additional Patterns

- Simple Factory.
- Null Object.

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