# ADE400 : Design Patterns

| | | | | | |
|---|---|---|---|---|---|
| **Code :** | ADE400 | **Duration :** | 3 days | **Category :** | Analysis and Design |

**Audience :**

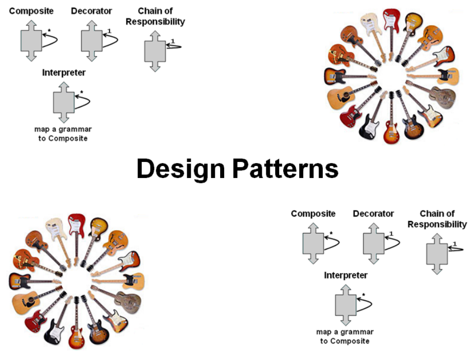
Experienced developers and software architects with knowledge of object oriented programming and systems analysis who want to apply Design Patterns when designing these systems.

**Prerequisites :**

Knowledge of an object-oriented programming language like C++, C#, or Java and experience with object oriented analysis and design with UML is required.

**Realization :**

The concepts are treated according to presentation slides. The theory is illustrated with demos of patterns in C++, C# and Java. There are exercises in design problems where patterns are applied. The course material is in English.



**Contents :**

In this course you will learn how design patterns can be applied to the object oriented design of systems. After an introduction about the role that design patterns play and how they can be used to realize the non-functional requirements of systems, attention is paid to how design patterns are described and cataloged. Also the role of design patterns in the architecture of applications is discussed and the various categories of design patterns that are distinguished. In the module Creational Patterns the Factory patterns and the Builder, Prototype and Singleton pattern are discussed. You learn out of which classes, relationships, responsibilities and cooperations a typical design pattern solution can consist. Next in the module the Structural Patterns the Adapter, Composite, Bridge, Decorator, Proxy and Flyweight pattern are discussed. You will learn the consequences of applying the patterns, the benefits and possible disadvantages in terms of time and space considerations and how to decide on the use of a particular pattern. Next in the module Behavioral Patterns the Chain of Responsibility, Interpreter, Iterator, Mediator, State and Strategy patterns are discussed. Finally the module Architectural Patterns considers certain patterns that are involved in the architectural structure of software including Operating Systems and Frameworks. This module focuses on the Layer pattern, the Micro Kernel pattern and the Model View Controller (MVC) pattern.

**Module 1 : Intro Design Patterns**

What is a design pattern?
Describing design patterns
Reuse through design patterns
Structure of patterns
Classification of patterns
Catalog of Design Patterns
Creational Patterns
Structural Patterns
Behavioral Patterns
Sample design patterns
Selecting Design Patterns
Solving problems with design patterns

**Module 2 : Creational Patterns**

Factory Patterns
Factory Method Pattern
Provide hooks for subclasses
Connect parallel class hierarchies
Abstract Factory Pattern
Encapsulating Implementation Dependencies
Concrete Class Isolation
Exchanging Product Families
Promoting Consistency
Builder Pattern
Vary Internal Representation
Isolate construction code
Controlling the build process
Prototype
Adding products at run-time
Reduced Subclassing
Dynamic configuration
Singleton Pattern
Controlled access
Reduced name space

**Module 3 : Structural Patterns**

Adapter Pattern
Pluggable Adapters
Two-way adapters
Bridge Adapters
Decoupling interface and implementation
Improved Extensibility
Hiding Implementation Details
Composite Pattern
Explicit parent references
Sharing Components
Decorator Pattern
Improved flexibility
Lots of little objects
Faade Pattern
Reducing client-subsystem coupling
Public subsystem classes
Private subsystem classes
Flyweight Pattern
Reducing number of instances
Proxy Pattern
Remote Proxies
Virtual Proxies
Copy-on-write

**Module 4 : Behavioral Patterns**

Chain of responsibility
Reduced Coupling
No guaranteed receipt
Command Pattern
Decoupling Objects
Use of callback functions
Supporting undo and redo
Avoiding error accumulation
Interpreter Pattern
Implementing Grammars
Grammar Changes
Abstract syntax tree
Iterator Pattern
Controlling iteration
Traversal algorithm
Mediator Pattern
Limiting subclasses
Simplification of object protocols
Memento Pattern
Preserving Encapsulation Boundaries
Storing incremental changes
Observer Pattern
Subject and receiver
Broadcast Communication
State Pattern
State transitions
Sharing State Objects
Strategy Pattern
Context interfaces
Template parameters
Template Pattern

**Module 5 : Architectural Patterns**

Architectural patterns versus design patterns
Patterns for real-time software
Layers
Pipes and Filters
Blackboard
Broker
Model-View-Controller
Presentation-Abstraction-Control
Microkernel
Reflection