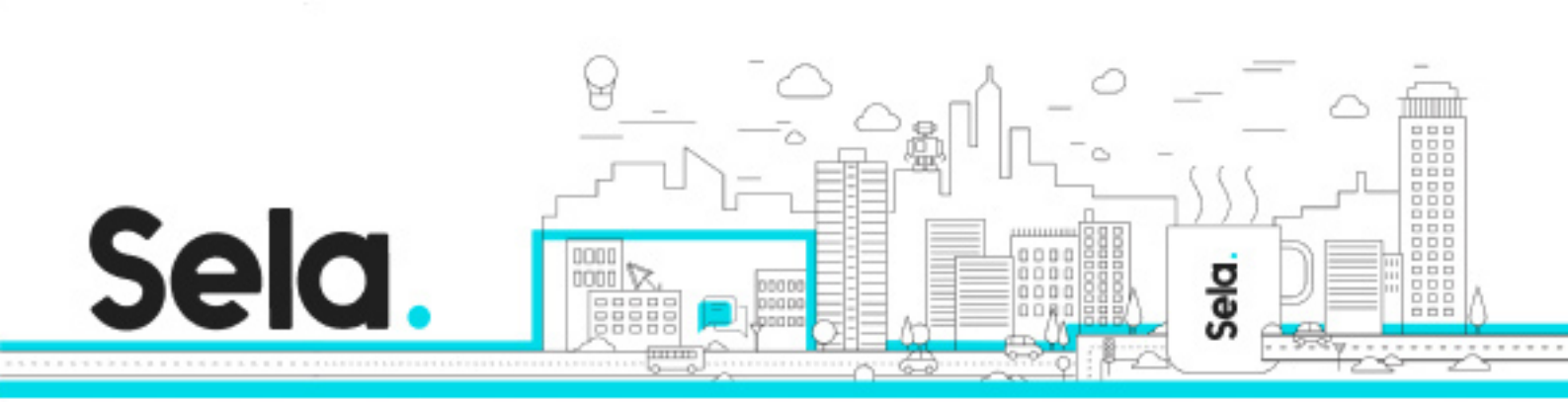
Sela.

C7

Design Patterns in C++

college@sela.co.il 03-6176666



Design Patterns in C++

C7 - Version: 3.1

5 days Course

Description:

The course proceeds beyond C++ programming to investigate advanced aspects of Object Oriented Programming (OOP), including the implementation of many useful, industry-standard design patterns, using C++ code examples and exercises. The course promotes a complete understanding of the object-oriented paradigm, both on requirement level (e.g., dynamic classification, multi-methods) and implementation level (inheritance layout, virtual table). The developer attending the course will be able to identify and implement many common patterns in low-level design, and some patterns that affect the top-level design, i.e., the class diagram and the contents of abstract interfaces. It is pre-requisite for the “OOD/ Object Modeling” course.

Intended audience:

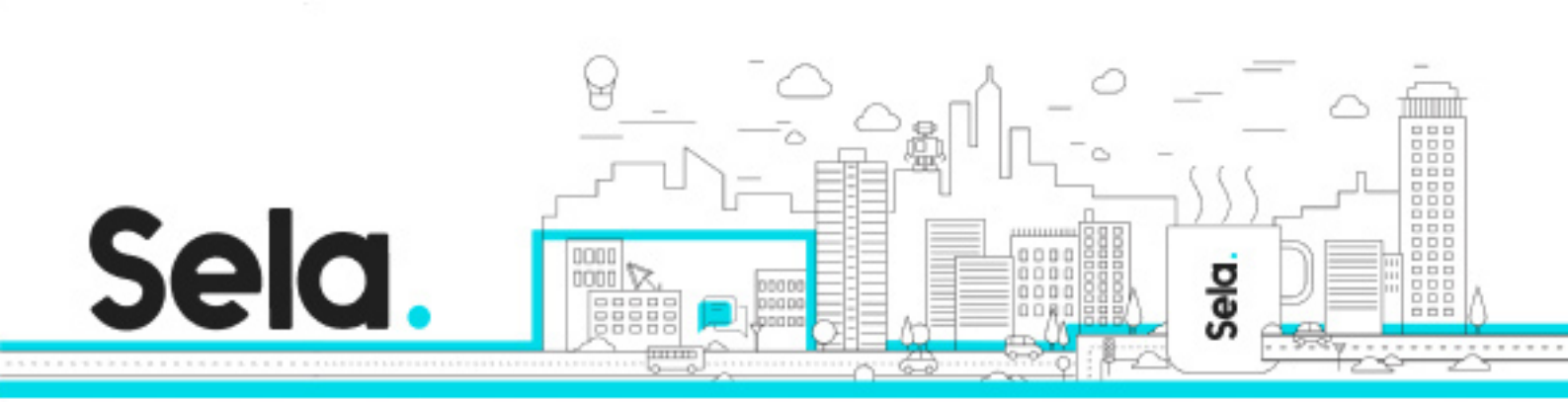
This course is intended for C++ programmers, C++ Project Managers, and C++ Designers.

Prerequisites:

Completed a high level C++ course
At least a year of experience programming in Object-Oriented C++

Objectives:

Know how to write better software using Design Patterns
Know how to utilize Design Patterns in projects.



Topics:

STL – The Standard Templates Library

- General Overview
- Containers
- Iterators
- Algorithms
- The ways to customize STL functionality
- Iterator Pattern

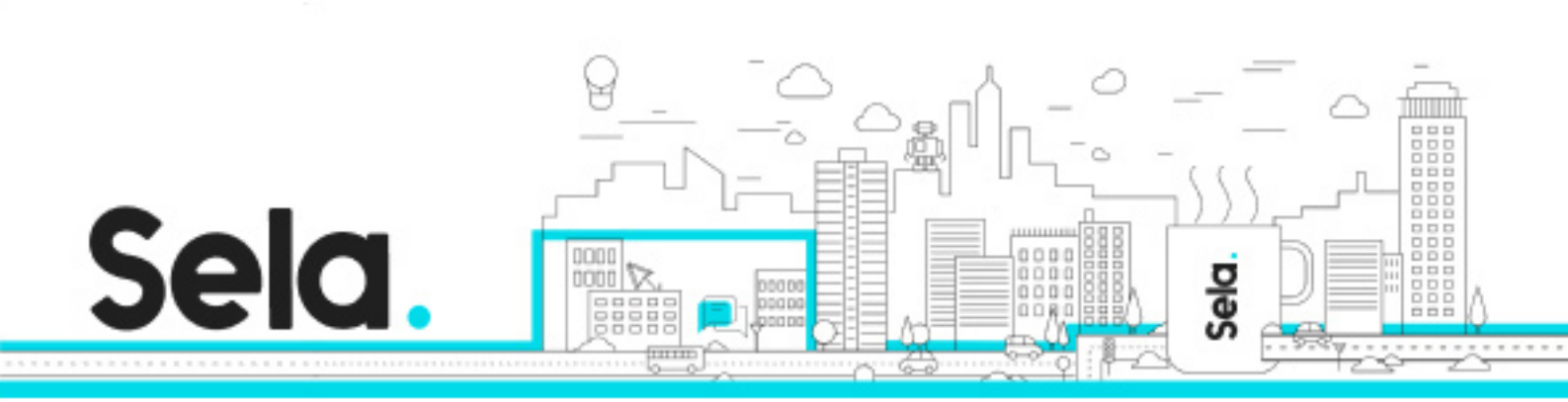
Introduction to OO Design Patterns

- The OO Design Challenge
- The Course Goals
- Application Score of Design Patterns
- Design Patterns - The Inspiration
- Design Pattern Items - GOF Form
- Common Terminology

Controlling Object Access

- UML - "The Class Diagramm"
- Reference Count Idiom
- RC Smart Pointer Pattern
- Flyweight pool Pattern
- Proxy Pattern

Initialization And Registration



- Singleton Pattern
- Handleton Pattern
- Object Initialization Order Pattern
- The "Open/Closed" Principle
- Registration Idiom
- Singleton Destruction Manager
- Command Pattern
- Observer Pattern

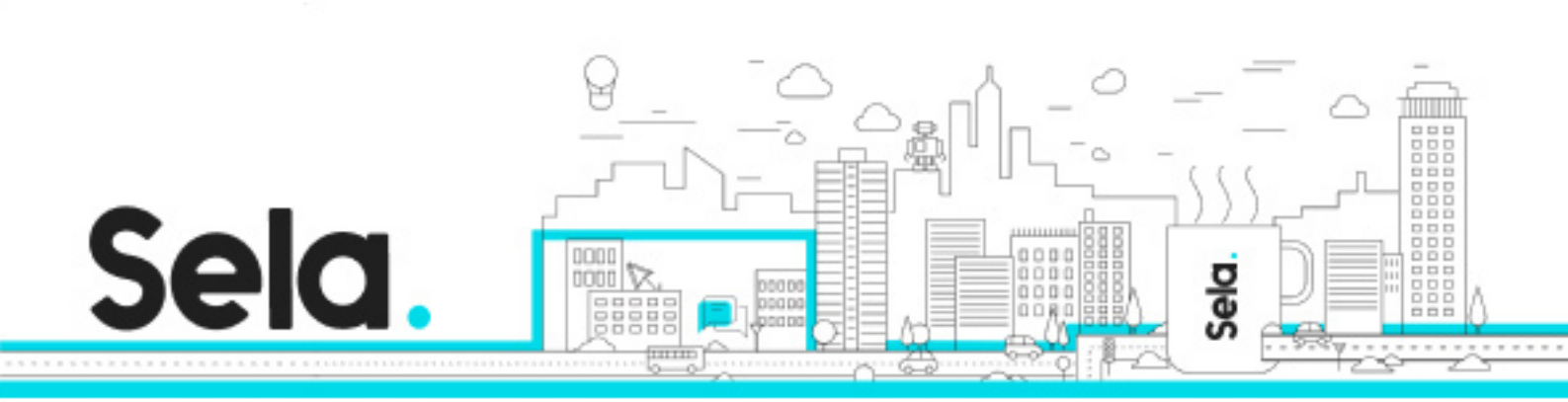
Changing Object Behaviour

- Strategy Pattern
- Liskov's substitutability principle
- State Pattern
- Bridge Pattern
- Decorator Pattern
- Adapter Pattern

Creating Objects Of Any Type

- Template Method Pattern
- Prototype Pattern
- Factory Method Pattern
- Abstract Factory Pattern
- Typelist Idiom
- Dynamic Pluggable Factory Pattern
- Product Trader Pattern
- Virtual Constructor Pattern

Distributing Functionality



- Composite Pattern
- Multi-methods Idiom
- Double Dispatch Idiom
- Visitor Pattern