Design Patterns

Level: intermediate / advanced

Length: 35 - 40 hours

Course objective: exercise the object oriented design principles by studying several design patterns, how they are expressed in a particular language

What You Will Learn

- Learn the design principles, how they are applied
- Show the place and the role of design patterns in software development
- Study the key general design patterns and learn to apply them
- The optional part is dedicated to design patterns which appear in concurrent context, specific to multi-threading programming.
- Gain a valuable experience by covering diverse practical problems.

Who can attend: programmers who want to take advantage in every day work of the experience and knowledge expressed by the design patterns.

Prerequisites: knowledge of an object oriented programming language at least at a medium level, basics of the object oriented programming principles, and basics of the Unified Modeling Language 2. In addition, for the optional part related to concurrency, the attendees have to know the support for multi-threading programming offered by the language (C++, Java, C#, etc.), or an external library or framework which offers support for multi-threading (for example POSIX threads).

Required facilities: VGA projector, white board, computers, development environment for an object oriented programming language (like C++, Java, C#, etc.).

Bibliography (minimal): Design Patterns – Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Addison-Wesley, ISBN 0-201-63361-2

Related courses: Unified Modeling Language; Applying of OOP, UML and Design Patterns, trainings for object oriented programming languages – C++, Java, C#, Python.

Note: the course will be personalized based on the attendees' technical profile (namely which programming languages they work with, their work experience).

Description

This course approaches a set of design patterns which includes those originally defined by GoF (see the minimal bibliography) by examples and case studies. Besides understanding the theory it is essential how they are used: recognize the context of usage, variants, implementation particularities which depend of a programming language.

The course covers a large number of design patterns, some are quite simple and they are handled briefly, some are quite complex or more interesting and we'll spend more time on them (like state or small language).

The course is highly interactive, it is an excellent opportunity to learn and exercise the principles of object oriented design.

The theoretical aspects are applied by solving practical problems, by particular implementations in Java, Python, C# or C++ (or other OO language).

The last chapter – design patterns related to concurrency – presents the fundamentals problems the programmers face when threads are used, how they are addressed by using the language support, shows several common patterns, how they are used, details of implementation.

Contents

1. Introduction: context of using them, definition, catalogs of design patterns, specific issues

2. **Principles of class design**: single responsibility, open/closed, Liskov substitution, dependency inversion, interface segregation; how are they applied, support of an object oriented programming languages for implementing them

3. **Fundamental patterns**: delegation, interface, abstract superclass, immutable object, proxy. Implementation of transactions by using a proxy.

4. **Creational patterns**: method factory, abstract factory, builder, prototype, singleton, monostate, object pool. Contexts of using them, examples.

5. **Partitioning patterns**: filter, composite object, read-only interface. How to dynamically process information by using filters. How to implement a composite object, how an iterator on a composite might be defined.

6. **Structural patterns**: adapter, iterator, bridge, facade, flyweight, decorator, cache management. Use of bridge for accessing several heterogeneous libraries. How to implement decorators.

7. **Behavioral patterns**: chain of responsibility, command, interpreter, mediator, snapshot, observer, state, strategy, null object, template method, visitor. Use of command to implement undo & redo, how to design & implement an interpreter, use of visitor to implement operations on a data structure.

8. **Concurrency patterns** (optional): critical zone, lock object, guarded suspension, balking, scheduler, read/write lock, producer/consumer, two step termination, double buffering, asynchronous processing, future, thread pool, double check locking, active object, monitor object, thread specific storage, leader/followers