Java Best Practices And Design Patterns

Course: **00005**
Filter: **Beginner**
Duration: **4 days**
Category:: **Java**
Price: **3895,00 €**

About Course

Solve real-world software development problems, and deliver responsive applications that are fast and reliable. In this training course, you learn how to leverage Java best practices, avoid pitfalls, perform industry-standard software development techniques, use design patterns to implement proven solutions to reoccurring problems, and apply idioms and patterns to improve your Java code.

What you'll learn

- Employ best practices to build reliable and scalable Java applications
- Effectively apply test-driven development to enhance program maintainability
- Solve architectural problems with proven design patterns
- Employ advanced Java APIs for multi-threaded programming

Pre-requisites

- Knowledge at the level of Java Programming Introduction
- Three to six months of Java programming experience
- Understand Java classes, the inheritance model, polymorphism, and encapsulation
- Use fundamental standard edition Java APIs
- Apply object-oriented analysis and design, including defining classes and creating objects

Curriculum

Module 1: Effective Programming in Java

- Clarifying the goals of best practices
- Identifying the key characteristics of high-quality software
- Organizing classes, packages and subsystems into layers
- Designing to the principles of SOLID

Module 2: Exploiting a testing framework

- Composing and maintaining JUnit tests
- Taking advantage of advanced JUnit features
- Testing in the presence of exceptions

Module 3: Monitoring software health using logging libraries

- Configuring logging with log4j and SLF4J
- Minimizing the impact of logging on performance

Module 4: Creating matchers and mock objects

- Writing custom Ham crest matchers
- Testing with fake objects and mocks

Module 5: Employing common design patterns

- Observer
- Iterator
- Template method
- Strategy
- State
- Data Accessor Object
- Data Transfer Object
- Composite
- Service Locator
- Proxy
- Factory

Module 6: Refactoring legacy code

- Identifying reasons to change software
- Clarifying the mechanics of change
- Writing tests for legacy classes and methods

Module 7: Improving type safety with generics and enum types

- Creating generic classes and methods
- Navigating generic class hierarchies
- Implementing enum types for fixed sets of constants

Module 8: Adding metadata by writing annotations

- Leveraging the built-in and custom annotations
- Annotating with meta-annotations

Module 9: Modifying runtime behavior with reflection

- Retrieving class and method data dynamically
- Flagging methods with naming conventions
- Adding information to code with annotations
- Assessing disadvantages of reflection

Module 10: Measuring and improving performance

- Assessing response time
- Conducting load and stress tests