Training: Capstone Courseware
192 Design Patterns in Java Software

TRAINING GOALS:

Version 8.0

This course seeks to develop, for the experienced Java programmer, a strong, shared vocabulary of design patterns and best practices. The course begins with a discussion of how to recognize and apply design patterns - that is, how to incorporate pattern awareness into one's own analysis, design, and implementation practices. The main body of the course focuses on the Gang of Four design patterns, with a chapter each on creational, behavioral, and structural patterns. The course includes both pencil-and-paper design exercises and traditional coding labs to reinforce finer points of important patterns.

This is not a patterns catalog: it is as much a study of how to "think in patterns" as it is an introduction to several of the most important patterns. Students will be challenged to bring their own previous development experience to the discussion, to see the patterns in everyday design and coding solutions. The course puts more emphasis on some patterns than others. We believe that students will be better served by going into several patterns in depth -- and with lively discussions of several others -- than by through every GoF pattern in rote form.

The course also includes an optional "Chapter Zero" on some more basic practices in object-oriented concepts and OO factoring and re-factoring. Though not appropriate for all students, it may be helpful for some audiences with less real-world Java experience.

Learning Objectives

- Start to think in terms of design patterns.
- Recognize and apply patterns to specific software development problems.
- Use known patterns as a shared vocabulary in designing and discussing solutions.
- Use Factories and Singletons to control object creation, for a variety of reasons.
- Use Observers, Observables, and Model/View/Controller systems to decouple application behavior and preserve code scalability.
- Understand the full motivation for the Command pattern and take advantage of Command frameworks in JFC.
- Implement Adapters, rather than building redundant classes or creating intermediate data structures for consumption by existing code.
- Understand and apply a range of other J2SE and J2EE patterns to improve code quality and scalability, and to produce high-quality solutions right off the bat.

Compendium Education Center Ltd.
ul. Tatarska 5, 30-103 Kraków, tel.: (12) 298 47 77
ul. Postępu 18B, 02-676 Warszawa, tel.: (22) 417 41 70
BNP Paribas Bank Polska S.A.
EUR: PL 79 1600 1462 1853 6451 3000 0006
USD: PL 52 1600 1462 1853 6451 3000 0007

CONSPECT:

- Object-Oriented Refactoring
  - A Place for Everything ...
  - Warning Signs
  - Magic Numbers and String Literals
  - Effective Use of Enumerated Types
  - Externalizing Volatile Information
  - Over-Encapsulation
  - Separation of Concerns
  - Making Classes Observable
  - Delegation Instead of Inheritance
  - Factories and Dependency Injection
- Recognizing and Applying Patterns
  - Design Patterns
  - Defining a Pattern
  - Unified Modeling Language
  - Seeing Patterns
  - Warning Signs and Pitfalls
  - Functional Programming and Its Impact on Patterns
- Creational Patterns
  - Factory Patterns
  - The Singleton Pattern
  - Singleton vs. Class Utility
  - APIs and Providers
  - Cascading Factories
  - Factories vs. Dependency Injection
- Behavioral Patterns
  - Un-Tangling Your Code
  - Warning Sign: Letting Subclasses Dictate
  - The Strategy Pattern
  - The Template Method Pattern
  - The Observer Pattern
  - Functional Interfaces as Observers
  - The Model/View/Controller Pattern
  - The Command Pattern

Compendium Education Center Ltd.
ul. Tatarska 5, 30-103 Kraków, tel.: (12) 298 47 77
ul. Postępu 18B, 02-676 Warszawa, tel.: (22) 417 41 70
BNP Paribas Bank Polska S.A.
EUR: PL 79 1600 1462 1853 6451 3000 0006
USD: PL 52 1600 1462 1853 6451 3000 0007

- The Chain of Responsibility Pattern
- Structural Patterns
  - The Composite Pattern
  - The Adapter Pattern
  - Adapters for Performance
  - The Decorator Pattern
  - The Façade Pattern
  - The Flyweight Pattern
  - Fixed vs. Open Flyweights
- J2EE Patterns
  - Model/View/Controller, Redux
  - The Intercepting Filter Pattern
  - The Front and Application Controller Patterns
  - The Business Delegate Pattern
  - The Service Locator Pattern
  - The Transfer Object Pattern
  - The Composite Entity Pattern
  - The Data Access Object Pattern

REQUIREMENTS:

Solid **Java programming** experience is essential - especially object-oriented use of the language. Language features and techniques that are integral to some lab exercises include interfaces and abstract classes, threading, generics and collections, and recursive methods. Course 103 is excellent preparation.

Previous experience with **UML** (**Unified Modeling Language**) will be helpful, but is not critical. The course uses UML class diagrams extensively but keeps notation fairly simple, and also includes a quick-reference appendix.

Difficulty level

CERTIFICATE:

The participants will obtain certificates signed by Capstone Courseware.

Compendium Education Center Ltd.
ul. Tatarska 5, 30-103 Kraków, tel.: (12) 298 47 77
ul. Postępu 18B, 02-676 Warszawa, tel.: (22) 417 41 70
BNP Paribas Bank Polska S.A.
EUR: PL 79 1600 1462 1853 6451 3000 0006
USD: PL 52 1600 1462 1853 6451 3000 0007

TRAINER:

Authorized Capstone Courseware Trainer.

Compendium Education Center Ltd.
ul. Tatarska 5, 30-103 Kraków, tel.: (12) 298 47 77
ul. Postępu 18B, 02-676 Warszawa, tel.: (22) 417 41 70

BNP Paribas Bank Polska S.A.
EUR: PL 79 1600 1462 1853 6451 3000 0006
USD: PL 52 1600 1462 1853 6451 3000 0007