Software Architecture Design Patterns in Java
Software Architecture Design Patterns in Java

Partha Kuchana
DEDICATION

To my family
CONTENTS

SECTION I: AN INTRODUCTION TO DESIGN PATTERNS

1  Design Patterns: Origin and History
   Architectural to Software Design Patterns
   What Is a Design Pattern?
   More about Design Patterns
   About This Book
       Source Code
       Source Code Disclaimer

SECTION II: UNIFIED MODELING LANGUAGE (UML)

2  UML: A Quick Reference
   Structure Diagrams
   Behavior Diagrams
   Model Management Diagrams
   Class Diagrams
       Class
       Inner Class
       Access Specifiers
       Static
       Abstract Class/Method
       Exception
       Note
       Generalization
       Interface
       Realization
       Dependency
       Class Association
           Multiplicity
           Navigability
           Composition
           Aggregation
   Sequence Diagrams
       Object
       Message
       Self Call

© 2004 by CRC Press LLC
SECTION III: BASIC PATTERNS

3  **Interface**
   Description
   Example
   Practice Questions

4  **Abstract Parent Class**
   Description
   Example
   Abstract Parent Class versus Interface
   Practice Questions

5  **Private Methods**
   Description
   Example
   Practice Questions

6  **Accessor Methods**
   Description
   Accessor Method Nomenclature
   Example
   Direct Reference versus Accessor Methods
   Practice Questions

7  **Constant Data Manager**
   Description
   Example
   Practice Questions

8  **Immutable Object**
   Description
   Example
   Practice Questions

9  **Monitor**
   Description
   Example
   Practice Questions

SECTION IV: CREATIONAL PATTERNS

10 **Factory Method**
   Description
   Example
   Practice Questions

11 **Singleton**
   Description
   Who Should Be Responsible?
   Example
   Make the Constructor Private
   Static Public Interface to Access an Instance
   Practice Questions
12 Abstract Factory
Description
Abstract Factory versus Factory Method
Example I
Example II
   Logical Flow When the Application Is Run
Practice Questions

13 Prototype
Description
Shallow Copy versus Deep Copy
   Shallow Copy Example
   Deep Copy Example
Example I
   Design Highlights of the HostingPlanKit Class
Example II
   Redesign the UserAccount Class
   Create a Prototype Factory Class
Practice Questions

14 Builder
Description
Example I
   A Side Note
      Back to the Example Application
Example II
Example III
Practice Questions

SECTION V: COLLECTIONAL PATTERNS

15 Composite
Description
Example
Design Approach I
   FileComponent
      getComponentSize()
   DirComponent
      addComponent(FileSystemComponent)
      getComponent(int)
      getComponentSize()
Design Approach II
Practice Questions

16 Iterator
Description
Iterators in Java
Filtered Iterators
Internal versus External Iterators
Example: Internal Iterator
Client/Container Interaction
Example: External Filtered Iterator
Practice Questions
17 Flyweight
   Description
   How to Design a Flyweight in Java
   Design Highlights
   Example
   Design Approach I
   Design Approach II
   Practice Questions

18 Visitor
   Description
      Design Idea 1
      Design Idea 2
   Defining New Operations on the Object Collection
   Adding Objects of a New Type to the Collection
   Example
   Design Approach I
   Design Approach II
   Design Approach III (Composite Pattern)
   Design Approach IV (The Visitor Pattern)
      Application Flow
   Defining a New Operation on the Order Object Collection
   Adding a New Order Type to the Collection
   Practice Questions

SECTION VI: STRUCTURAL PATTERNS

19 Decorator
   Description
   Characteristics of a Decorator
   Example
   Concrete Logger Decorators
      HTMLLogger
      EncryptLogger
   Adding a New Message Logger
   Adding a New Decorator
   Practice Questions

20 Adapter
   Description
   Class Adapters versus Object Adapters
      Class Adapter
      Object Adapter
   Example
   Address Adapter as an Object Adapter
   Practice Questions

21 Chain of Responsibility
   Description
   Example
   Practice Questions
22 **Façade**
Description
Example
Important Notes
Practice Questions

23 **Proxy**
Description
Proxy versus Other Patterns.
   - Proxy versus Decorator
   - Proxy versus Façade
   - Proxy versus Chain of Responsibility
RMI: A Quick Overview
   - RMI Components
   - RMI Communication Mechanism
RMI and Proxy Pattern
Example
Additional Notes
   - Compilation and Deployment Notes
Practice Questions

24 **Bridge**
Description
Example
   - Abstraction Implementation Design
   - Abstraction Interface Design
   - Design Highlights of the Abstraction Interface Classes
Bridge Pattern versus Adapter Pattern
Practice Questions

25 **Virtual Proxy**
Description
   - Advantage
   - Disadvantage
Example
Practice Questions

26 **Counting Proxy**
Description
Example
Practice Questions

27 **Aggregate Enforcer**
Description
Example
   - Design Approach I (On-Demand Initialization)
   - Design Approach II (Early Initialization)
   - Design Approach III (Final Variables)
Practice Questions

28 **Explicit Object Release**
Description
   - The finalize Method
When an Object Goes Out of Scope, It Is Believed to Be Garbage Collected Immediately
The Garbage Collection Process Runs as a Low-Level Background Daemon Thread
The finally Statement
Example
Best Case Scenario
Exception Scenario 1
Exception Scenario 2
Practice Questions

29 Object Cache
Description
Example
Practice Questions

SECTION VII: BEHAVIORAL PATTERNS

30 Command
Description
Example I
Example II
   Application Flow
Practice Questions

31 Mediator
Description
Mediator versus Façade
Example I
   Client Usage of the Mediator
   User Interface Objects: Mediator Interaction
Example II
Practice Questions

32 Memento
Description
Example
   DataConverter (Originator)
      ID
      Memento
      process
      createMemento
      setMemento
   DCClient (Client)
   MementoHandler
Practice Questions

33 Observer
Description
Adding New Observers
Example
   Subject–Observer Association
      Logical Flow
Practice Questions

34 **Interpreter**
Description
Example
Infix-to-Postfix Conversion (Listing 34.8)
Construction of the Tree Structure (Listing 34.9)
Postorder Traversal of the Tree
Additional Notes
Infix-to-Postfix Conversion
Infix Expression
Postfix Expression
Conversion Algorithm
Example
Binary Tree Traversal Techniques
Preorder (Node-Left-Right)
In-Order (Left-Node-Right)
Postorder (Left-Right-Node)
Level-Order
Practice Questions

35 **State**
Description
Stateful Object: An Example
Example
Practice Questions

36 **Strategy**
Description
Strategies versus Other Alternatives
Strategy versus State
Example
SimpleEncryption
CaesarCypher
SubstitutionCypher
CodeBookCypher
Practice Questions

37 **Null Object**
Description
Example
Practice Questions

38 **Template Method**
Description
Abstract Class
Concrete Class
Example
Additional Notes
Mod 10 Check Digit Algorithm
Practice Questions

39 **Object Authenticator**
Description
Example
Practice Questions

40 **Common Attribute Registry**
Description
Example
Practice Questions

**SECTION VIII: CONCURRENcy PATTERNS**

41 **Critical Section**
Description
Example
   Approach I (Critical Section)
   Approach II (Static Early Initialization)
Practice Questions

42 **Consistent Lock Order**
Description
Example
Practice Questions

43 **Guarded Suspension**
Description
Example
   Use of wait() and notify() in the ParkingLot Class Design
Practice Questions

44 **Read-Write Lock**
Description
   Design Highlights of the ReadWriteLock Class
   Lock Statistics
   Lock Methods
   Lock Release
Example
Practice Questions

**SECTION IX: CASE STUDY**

45 **Case Study: A Web Hosting Company**
Objective
KPS Hosting Solutions: A Brief Overview
Requirements
   Functional
   Technical
Business Objects and Their Association
Framework for Application Processing
   Enterprise Service Level
   Generic Interface Contract
   Sample Interface Contract
Task Level
   Generic Task-Handler Mapping
   Sample Task-Handler Mapping
Error Processing
Enterprise Services Design
Address Validation
Credit Card Service
Validation
Search Management
Customer Management
Conclusion

SECTION X: APPENDICES

Appendix A: List of Design Patterns
Appendix B: References
FOREWORD

Partha Kuchana is an experienced enterprise systems architect. He understands that patterns are not about things that are just good ideas, but that patterns are about capturing knowledge bred from experience. This hard-won knowledge is what Partha is sharing with readers of his book. Here are some of the things I really like about what he has to say.

The book presents 42 design patterns, which include the 23 GoF patterns. These patterns are categorized as follows:

- 7 Basic patterns
- 5 Creational patterns
- 4 Collectional patterns
- 11 Structural patterns
- 11 Behavioral patterns
- 4 Concurrency patterns

The discussion of each pattern includes an example implemented in Java. Further, the source code for all examples is found on the following Web site for this book: http://www.crcpress.com/e_products/downloads/download.asp. The source code and the easily understood examples make this format work well.

Partha takes complex material and clearly explains the ideas so they are easy-to-understand, an important consideration for both the novice encountering the material for the first time and the experienced developer who quickly wants to extract the important bits for immediate use. Each pattern discussion also includes Practice Questions for exactly that — your own use to improve your skills or, if this book were to be chosen as a text, to help the time-pressured instructor.

Partha takes the time to compare and contrast the patterns. For example, in the discussion on the Mediator pattern, a table shows similarities and differences between Mediator and Façade. The reader will find that this analysis leads to a clearer understanding than simply trying to focus on each pattern in isolation. The text also includes consideration of relationships between patterns. For example, in the discussion on the Mediator pattern there is a reference to a previous design example for the Command pattern.
Finally, at the end of the book, the reader will be happy to find a case study that pulls some of the patterns together to illustrate how a more complicated problem would be tackled and how the patterns work together. As those who have studied the work of Christopher Alexander realize—patterns are not applied in isolation but collaborate within a specific domain to address large and small problems.

It has been ten years since the GoF book was published. A lot of patterns have been identified and captured in that time. A lot of patterns books have been written. This book is like the GoF book, a catalog; probably not one you will read cover-to-cover in a single setting, but which will find a place on your bookshelf. Keep it handy for all those “How do I do this in Java?” questions where you wish you had an expert in the office next door to provide answers. This book is the next best thing.

**Linda Rising**  
Phoenix, AZ
Partha Kuchana is an experienced enterprise systems architect. He has eleven years of experience in all aspects of project delivery management (onsite/offshore models), enterprise architecture, design, development, mentoring and training. He is a Sun certified enterprise architect.

During the last several years, he has worked on numerous client–server, E-business, Web portal and enterprise application integration (EAI) projects at various client sites in the United Kingdom and the United States, involving iterative design methodologies such as Rational Unified Process (RUP) and extreme programming.

He has extensive experience applying design patterns in application architecture and design. He has successfully architected and designed business-to-business systems and complex heterogeneous systems integration using Web services, middleware and messaging products from various vendors. He has several published software-related publications.

Home page: http://members.ITJobsList.com/partha
E-mail: ParthaKuchana@ITJobsList.com
ACKNOWLEDGMENTS

First and foremost, I would like to thank my wife for her patience and support, for taking some of my workload especially in the areas of UML and Java programming and for her inspirational contributions at the time of frustrating moments. I would like to thank my parents, my sister, my brother and my dear friends whose support and encouragement throughout my life have made it possible for me to build the skill set necessary to succeed.

I would like to thank Venu Kuchana and D.R. Sudhakar for their contributions in terms of writing different Java programs. I would like to thank Balalingam Kuchana for his contributions in the area of UML and for being in charge of creating the formatted version of my draft.

I would like to thank the entire team at Auerbach publications for their contributions in this project and for making this a remarkable experience. In particular, I have a deep sense of gratitude towards my acquisitions editor, John Wyzalek, for sharing my enthusiasm and providing me with great advice and help. I also would like to thank the managing editor, Claire Miller, for her invaluable advice and contribution in arranging the book in a presentable form.

My sincere thanks to Linda Rising for writing the Foreword.

I am truly appreciative and thankful to the following reviewers who have taken the time to read the draft and provide me with feedback.

- Pradyumn Sharma, CEO, Pragati Software Pvt. Ltd.
- Carsten Kuckuk, project lead, Design Patterns Study Group Stuttgart, RIB Software AG
- Tim Kemper, Boulder Design Patterns Group
- Geoffrey Sparks, CEO, Sparx Systems P/L
- Edward L. Howe, software architect, Employease, Inc.
- Christopher R. Gardner, software developer, McKesson Information Solutions
- David Deriso, senior software engineer, Employease, Inc.
- Mike Heinrich, software engineer, Canada
- Rodney Waldoff, director of systems architecture, Encyclopedia Brittanica Inc.
- Thomas SMETS, software engineer, Belgium
- Linda Rising, Ph.D., independent software consultant, Arizona State University
- Ray Tayek, coordinator LAJUG/OCJUG
In particular, thanks to Pradyumn Sharma, Carsten Kuckuk, and Tim Kemper for their insightful recommendations, their thoroughness, and their invaluable suggestions, including questions that an inquisitive reader might have about design patterns.

I would like to thank Mark Grand for his encouragement and advice on various aspects of writing a patterns book. I am sure I have forgotten someone important; please accept my sincere apologies.