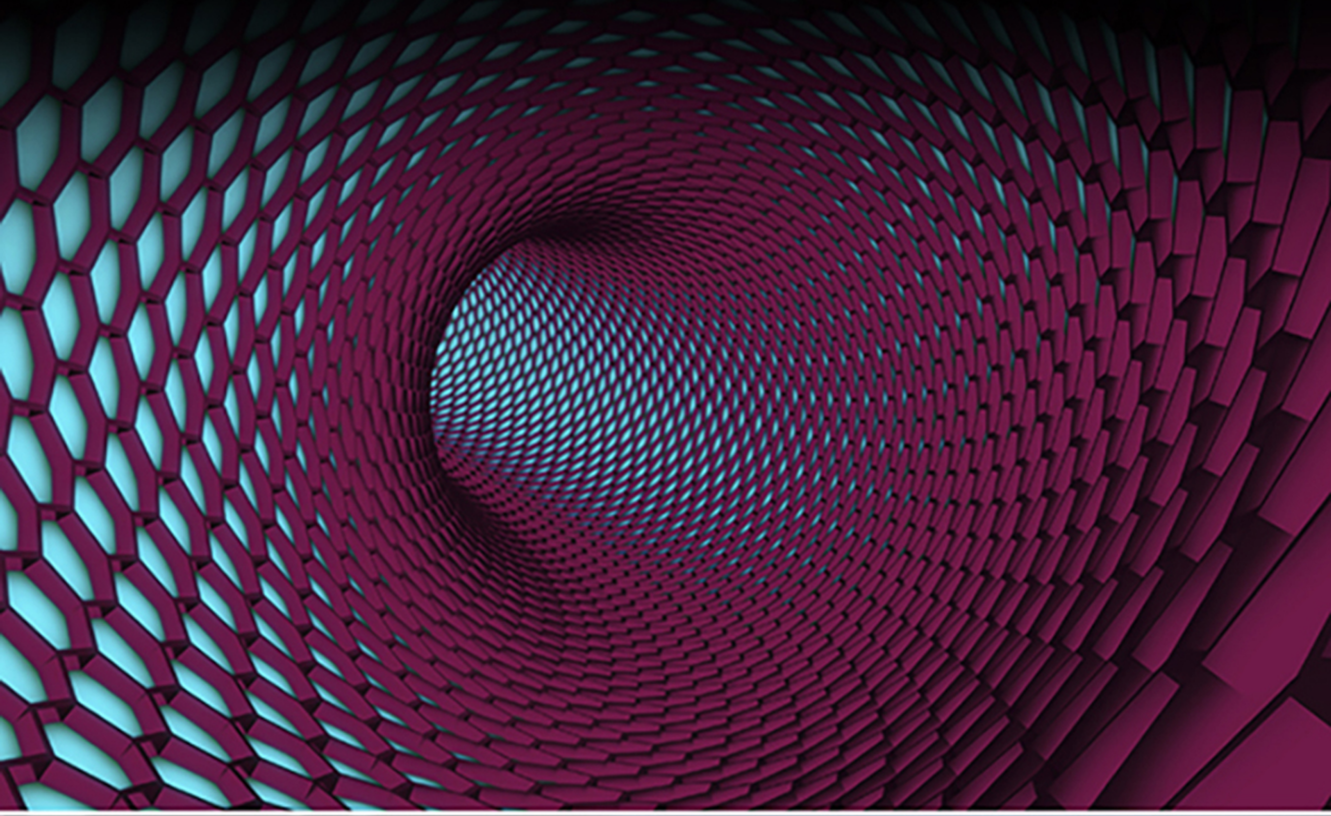
Robert C. Martin Series

# Java Application Architecture

Modularity Patterns with Examples Using OSGi



Kirk Knoernschild

*Forewords by* Robert C. Martin *and* Peter Kriens

## Praise for *Java Application Architecture*

"The fundamentals never go out of style, and in this book Kirk returns us to the fundamentals of architecting economically interesting software-intensive systems of quality. You'll find this work to be well-written, timely, and full of pragmatic ideas."

—*Grady Booch, IBM Fellow*

"Along with GOF's *Design Patterns*, Kirk Knoernschild's *Java Application Architecture* is a must-own for every enterprise developer and architect and on the required reading list for all Paremus engineers."

—*Richard Nicholson, Paremus CEO, President of the OSGi Alliance*

"In writing this book, Kirk has done the software community a great service: He's captured much of the received wisdom about modularity in a form that can be understood by newcomers, taught in computer science courses, and referred to by experienced programmers. I hope this book finds the wide audience it deserves."

—*Glyn Normington, Eclipse Virgo Project Lead*

"Our industry needs to start thinking in terms of modules—it needs this book!"

—*Chris Chedgey, Founder and CEO, Structure 101*

"In this book, Kirk Knoernschild provides us with the design patterns we need to make modular software development work in the real world. While it's true that modularity can help us manage complexity and create more maintainable software, there's no free lunch. If you want to achieve the benefits modularity has to offer, buy this book."

—*Patrick Paulin, Consultant and Trainer, Modular Mind*

"Kirk has expertly documented the best practices for using OSGi and Eclipse runtime technology. A book any senior Java developer needs to read to better understand how to create great software."

—*Mike Milinkovich, Executive Director, Eclipse Foundation*

# Java Application Architecture: Modularity Patterns with Examples Using OSGi

## Table of Contents

- Cover
- Half Title
- Title Page
- Copyright Page
- Contents
- Foreword
- Foreword
- Acknowledgments
- About the Author
- Introduction
    - Object-Oriented Design
    - Logical versus Physical Design
    - Modularity
        - Unit of Modularity: The JAR File
        - OSGi
    - Who This Book Is For
    - How This Book Is Organized
        - Part I: The Case for Modularity
        - Part II: The Patterns
        - Part III: POMA and OSGi

# Table of Contents

Pattern Form

Pattern Name

Pattern Statement

Sketch

Description

Implementation Variations

Consequences

Sample

Wrapping Up

Pattern Catalog

The Code

An Opening Thought on the Modularity Patterns

Reference

PART I: THE CASE FOR MODULARITY

Chapter 1 Module Defined

1.1 Defining a Module

1.2 Succinct Definition of a Software Module

1.3 Conclusion

Chapter 2 The Two Facets of Modularity

2.1 The Runtime Model

2.2 The Development Model

2.3 Modularity Today

2.4 Conclusion

Chapter 3 Architecture and Modularity

3.1 Defining Architecture

3.2 A Software Architecture Story

3.3 The Goal of Architecture

3.4 Modularity: The Missing Ingredient

3.5 Answering Our Questions

Pearson

# Table of Contents

3.6 Conclusion

3.7 References

Chapter 4 Taming the Beast Named Complexity

4.1 Enterprise Complexity

4.2 Technical Debt

4.3 Design Rot

4.4 Cyclic DependenciesThe Death Knell

4.5 Joints, Modules, and SOLID

4.6 Managing Complexity

4.7 Benefits of Modularity

4.8 Conclusion

4.9 References

Chapter 5 Realizing Reuse

5.1 The Use/Reuse Paradox

5.2 The Reuse Disclaimer

5.3 Reuse or Use

5.4 Modular Tension

5.5 Modular Design

5.6 Conclusion

5.7 Reference

Chapter 6 Modularity and SOA

6.1 All the Way Down, Revisited

6.2 GranularityArchitectures Nemesis

6.3 An Alternate View

6.4 Conclusion

Chapter 7 Reference Implementation

7.1 Why No OSGi?

7.2 Background on This Exercise: Building the System

7.3 Version 1

# Table of Contents

7.4 First Refactoring

7.5 Second Refactoring

7.6 Third Refactoring

7.7 Fourth Refactoring

7.8 Fifth Refactoring

7.9 Sixth Refactoring

7.10 Seventh Refactoring

7.11 The Postmortem

7.12 Conclusion

7.13 Reference

PART II: THE PATTERNS

Chapter 8 Base Patterns

Manage Relationships

Module Reuse

Cohesive Modules

Chapter 9 Dependency Patterns

Acyclic Relationships

Levelize Modules

Physical Layers

Container Independence

Independent Deployment

Reference

Chapter 10 Usability Patterns

Published Interface

External Configuration

Default Implementation

Module Facade

Chapter 11 Extensibility Patterns

Abstract Modules

Pearson

# Table of Contents

Implementation Factory

Separate Abstractions

Reference

Chapter 12 Utility Patterns

Colocate Exceptions

Levelize Build

Test Module

PART III: POMA AND OSGI

Chapter 13 Introducing OSGi

13.1 Some History

13.2 Benefits of OSGi

13.3 Digesting OSGi

13.4 OSGi Bundle

13.5 OSGi Runtime Management

13.6 The Two Facets of Modularity, Revisited

13.7 OSGi and the Patterns

Chapter 14 The Loan Sample and OSGi

14.1 Getting Started

14.2 The Manifests

14.3 &#956;Services

14.4 Installation and Execution

14.5 Conclusion

Chapter 15 OSGi and Scala

15.1 Getting Started

15.2 The Scala Code

15.3 Scala Bean Configuration

15.4 Scala &#956;Service Configuration

15.5 Building the Scala Module

15.6 Installation and Execution

# Table of Contents

15.7 Conclusion

Chapter 16 OSGi and Groovy

16.1 Getting Started

16.2 The Groovy Code

16.3 Groovy Bean Configuration

16.4 Groovy Service Configuration

16.5 Building the Groovy Module

16.6 Installation and Execution

16.7 Conclusion

Chapter 17 Future of OSGi

17.1 OSGi as an Enabler

17.2 The Disruption

17.3 The Power of Ecosystems

17.4 The Ecosystem

17.5 Conclusion

Appendix: SOLID Principles of Class Design

Single Responsibility Principle (SRP)

Open Closed Principle (OCP)

Liskov Substitution Principle (LSP)

Dependency Inversion Principle (DIP)

Interface Segregation Principle

Composite Reuse Principle (CRP)

References

Index

Pearson