Java Design Patterns

A Hands-On Experience with Real-World Examples

Second Edition

Vaskaran Sarcar

Foreword by Sunil Sati

Apress®

Java Design Patterns

A Hands-On Experience with Real-World Examples

Second Edition

Vaskaran Sarcar
Foreword by Sunil Sati

Apress®

Java Design Patterns: A Hands-On Experience with Real-World Examples

Vaskaran Sarcar
Bangalore, Karnataka, India

ISBN-13 (pbk): 978-1-4842-4077-9 ISBN-13 (electronic): 978-1-4842-4078-6
https://doi.org/10.1007/978-1-4842-4078-6

Library of Congress Control Number: 2018964945

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: Laura Berendson
Coordinating Editor: Aditee Mirashi

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-4077-9. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

This book is dedicated to
Almighty God, my family, and the Gang of Four.
You are my inspiration.

Table of Contents

About the Author

Vaskaran Sarcar obtained his Master of Engineering degree from Jadavpur University, Kolkata. Currently, he is senior software engineer and team lead in the R&D Hub at HP Inc. India. He was a national Gate Scholar and has more than 12 years of experience in education and the IT industry. He is an alumnus of prestigious institutions in India, such as Jadavpur University, Vidyasagar University, and Presidency University (formerly Presidency College). Reading and learning new things are his passions. You can connect with him at `vaskaran@rediffmail.com` or find him on LinkedIn at `www.linkedin.com/in/vaskaransarcar`.

Other books by Vaskaran include the following:

- *Design Patterns in C#* (Apress, 2018)
- *Interactive C#* (Apress, 2017)
- *Interactive Object-Oriented Programming in Java* (Apress, 2016)
- *Java Design Patterns (First Edition)* (Apress, 2016)
- *C# Basics: Test Your Skill* (CreateSpace, 2015)
- *Operating System: Computer Science Interview Series* (CreateSpace, 2014)

About the Technical Reviewers

Shekhar Kumar Maravi is a system software engineer whose main interests are programming languages, algorithms, and data structures. He obtained his master's degree in computer science and engineering from the Indian Institute of Technology, Bombay. After graduation, he joined Hewlett-Packard's R&D Hub in India to work on printer firmware. Currently, he is a technical lead for automated lab diagnostic device firmware and software at Siemens Healthcare India. He can be reached by email at `shekhar.maravi@gmail.com` or via LinkedIn at `www.linkedin.com/in/shekharmaravi`.

Ritesh Jha is passionate about large-scale distributed systems. Currently, he is working as a senior development engineer for the Supply Chain Technology Group at Walmart Labs. Before Walmart, he worked at eBay and Hewlett-Packard. He has a BE in computer science from Jadavpur University, Kolkata. When he is not exploring new technologies, he can be found exploring new places on his bike. He can be reached by email at `ritesh.jha@hotmail.com` or via LinkedIn at `www.linkedin.com/in/riteshjha9/`.

Ankit Khare is a senior software engineer with expertise in software architecture and designing, programming languages, algorithms, and data structure. After obtaining a BE in computer science, he joined Hewlett-Packard's R&D Center in India in 2010, where he worked with various laser-jet firmware teams. He is currently involved in future machine vision development for print image diagnostic tools involving ink-jet, large-format, and laser-jet printers. He can be reached by email at `akikhare@gmail.com` or via LinkedIn `https://www.linkedin.com/in/khareankit/`.

Acknowledgments

At first, I thank the Almighty. I sincerely believe that with His blessings only, I could complete this book. I extend my deepest gratitude and thanks to

Ratanlal Sarkar and Manikuntala Sarkar. My dear parents, with your blessings only, I could complete the work.

Indrani, my wife, and Ambika, my daughter. Sweethearts, once again, without your love, I could not proceed at all. I know that we needed to limit many social gatherings and invitations to complete this work on time and each time I promise you that I'll take a long break and spend more time with you.

Sambaran, my brother. Thank you for your constant encouragement toward me.

Shekhar, Ritesh, and Ankit. You are my friends and technical advisors. I know that whenever I was in need, your supports were there. Thank you one more time.

Anupam. My friend and another technical advisor. Though this time, you were not involve but still I acknowledge your support and help toward me in the development of Java Design Patterns first edition.

Sunil Sati. My ex-colleague cum senior. A special thanks to you for investing your time to write a foreword for my book. From the moment when experts like you agreed to write for me, I got some additional motivation to enhance the quality of my work.

Celestin, thanks for giving me another opportunity to work with you and Apress.

Laura, Amrita, Nagarajan, Sivachandran, Pradapsankar and Vinoth thank you for your exceptional support to beautify my work.

Lastly, I extend my deepest gratitude to my publisher, the editorial board members, and everyone who directly or indirectly supported this book.

Foreword

"A problem well stated is a problem half solved."

—Charles Kettering, inventor and engineer

To build on this concept, I must say that thinking about all possible scenarios and coming out with a best possible option is the key to a robust and lasting solution. This new second edition of *Java Design Patterns* will serve as a mentor and guide to engineers and designers who are regularly challenged to come up with the best possible solution in resource-constrained environments. This book explains in very clear terms the design patterns, the alternatives, and the concept of antipatterns. The complete section on antipatterns is very thought-provoking and helps us appreciate the utility of design in the first place.

As in his previous books, Vaskaran has provided hands-on experience in implementing design patterns. His innate way of getting engineers to think of an alternative solution is very insightful. This book will serve as mentor and task master as one traverses the chapters.

When I reflect on my interactions with Vaskaran while facing some complex engineering issues in HPI, I find him to be a keen listener, deep thinker, and a person who doesn't rush for a solution. After analyzing all possible alternatives, he picks the best one among the lot. In summary, this is what this book all about.

Sunil Sati

Senior Project Manager, BU Automotive Division, NXP Semiconductors

About Sunil Sati

Sunil Sati is an engineer with a major in electronics and communication from NIIT Surathkal and EGMP from IIM-Bangalore. He has 23 years of experience in various roles and capacities in process automation and semiconductor industries. Currently, he is working as senior manager with NXP Semiconductors in the automotive division. He was the brand ambassador in HPI for the Print Renaissance program. He loves to work and build teams across geographic locations.

Introduction

Welcome to your journey through *Design Patterns in C#*.

This is an introductory guide to the design patterns that you want to use in Java. You probably know that the concept of design patterns became extremely popular with the Gang of Four's famous book *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley, 1994). Most important, these concepts still apply in today's programming world. The book came out at the end of 1994, and it primarily focused on C++.

But Sun Microsystems released its first public implementation Java 1.0 in 1995. So, in 1995, Java was totally new to the programming world. But it grew rapidly, becoming rich with features. It has now secured its rank in world's top programming languages. In today's programming world, it is always in high demand. On the other hand, the concepts of design patterns are universal. So, when you exercise these fundamental concepts of design patterns with Java, you will be a better programmer and you'll open new opportunities for yourself.

In 2015, I wrote *Design Patterns in C#: Computer Science Interview Series*, and in 2016, I wrote *Java Design Pattern : A tour with 23 Gang of Four Design Patterns in Java*. They are basically the companions to this book.

In those books, my core intention was to implement each of the 23 Gang of Four (GoF) design patterns with C# and Java implementations. I wanted to present each pattern with simple examples. One thing was always on my mind when writing *Java Design Patterns* (First Edition): I wanted to use the most basic constructs of Java, so that the code would be compatible with both the upcoming version and the legacy version of Java. I have found this method helpful in the world of programming.

In the last two years, I got a lot of constructive feedback from my readers. This fully revised and updated version is created keeping those feedback in mind. I also took the opportunity to update the formatting and correct some typos in the previous version of the book and add new content to this new edition.

This time, I wanted to focus on another important area; I call it the "doubt-clearing sessions." I knew that if I could add some more information such as alternative ways to write these implementations, the pros and cons of these patterns, when to choose one

approach over another, and so on, readers would find this book even more helpful. So, in this enhanced version of the original, I have added a "Q&A Session" section to each chapter that can help you learn about each pattern in more depth.

In the world of programming, there is no shortage of patterns, and each has its own significance. So, in addition to the 23 GoF design patterns covered in Part I, I discuss three design patterns that are equally important in today's world of programming in Part II. Finally, in Part III, I discuss the criticism of design patterns and give you an overview of antipatterns, which are also important when you implement the concepts of design patterns in your applications.

Before jumping into these topics, I want to highlight few more points.

- You are an intelligent person. You have chosen a subject that can assist you throughout your career. If you are a developer/programmer, you need these concepts. If you are an architect at a software organization, you need these concepts. If you are a college student, you need these concepts, not only to score high on exams but to enter the corporate world. Even if you are a tester who needs to take care of white-box testing or needs to know about the code paths of a product, these concepts will help you a lot.
- I already mentioned that this book was written using the most basic features of Java so that you do not need to be familiar with advanced Java topics. These examples are simple and straightforward. I believe that these examples are written in such a way that even if you are familiar with another popular language, such as C#, C++, and so on, you can still easily grasp the concepts in this book.
- There are many books about design patterns and related topics. You may be wondering why I would want to write a new one about the same topics. The simple answer is that I have found other reference material to be scattered. Second, in most cases, many of those examples are unnecessarily large and complex. I like simple examples. I believe that anyone can grasp a new idea with simple examples, and if the core concept is clear, you can easily move into more advanced areas. I believe that this book scores high in this context. The illustrated examples are simple. I wanted to keep this book concise so that it motivates you to continue your journey of learning.

- Each chapter is divided into six parts: a definition (which is basically called *intent* in *Design Patterns: Elements of Reusable Object-Oriented Software*), a core concept, a real-world example, a computer/coding–world example, a sample program with various output, and the Q&A Session section. These Q&A Session sections help you learn about each pattern in more depth.
- Please remember that you have just started on this journey. As you learn about these concepts, try to write your own code; only then will you master the area.
- You will be able to download all the source code in the book from the publisher's website. I plan to maintain the "Errata," and if required, I can also make update/announcements there. So, I suggest that you visit those pages to receive any corrections or updates.

Guidelines for Using This Book

Here are some suggestions for you to use the book more effectively.

- You should have a basic understanding of Java and you should know how to create classes, interfaces, and so forth. It is helpful to be familiar with common terms in object-oriented programming; for example, encapsulation, abstraction, polymorphism, and inheritance.
- I assume that you have some idea about the GoF design patterns. If you are absolutely new to design patterns, I suggest you quickly go through Appendix A. This appendix will help you become familiar with the basic concepts of design patterns.
- If you are confident about the content in Appendix A, you can start with any part of the book. But I suggest you go through the chapters sequentially. The reason is that some fundamental design techniques may be discussed in the "Q&A Session" section of a previous chapter, and I did not repeat those techniques in the later chapters.

- There is only one exception to the previous suggestion. There are three factory patterns: simple factory, factory method, and abstract factory. These three patterns are closely related, but the simple factory pattern does not directly fall into the GoF design catalog, so it appears in Part II of the book. So, I suggest that when you start learning about these three factory patterns, you begin with the simple factory pattern.
- These programs are tested with Java 8 (update 172). I used the Eclipse editor in a Windows 10 environment. So, in the Eclipse Package Explorer view, you may notice the string jdk1.8.0_172. At the time of this writing, Photon is the latest edition of Eclipse (released in June 27, 2018), Java 8 is the long-term support (LTS) version, and Java 10 is the rapid release version. Java 11 is the next LTS version after Java 8 and planned for September 2018. But all of this version information should not matter much because I used the most basic constructs of Java. So, I believe that the code should execute smoothly in the upcoming versions of Java/Eclipse as well.
- One of my reviewers tested the code in a Linux environment. So, I believe that the results should not vary in other environments, but you know the nature of software—it is naughty. So, I recommend that if you want to see the same output, it is best if you can mimic the same environment.
- To draw class diagrams, ObjectAid Uml Explorer is used in the Eclipse editor. It is a lightweight tool for Eclipse. At the time of this writing, it is free if you want to draw the class diagrams, but to draw the sequence diagrams, you need to purchase a license. The website at `www.objectaid.com/home` gives more information about licenses, terms, and conditions.

Lastly, I hope that this enhanced edition provides more help to you.

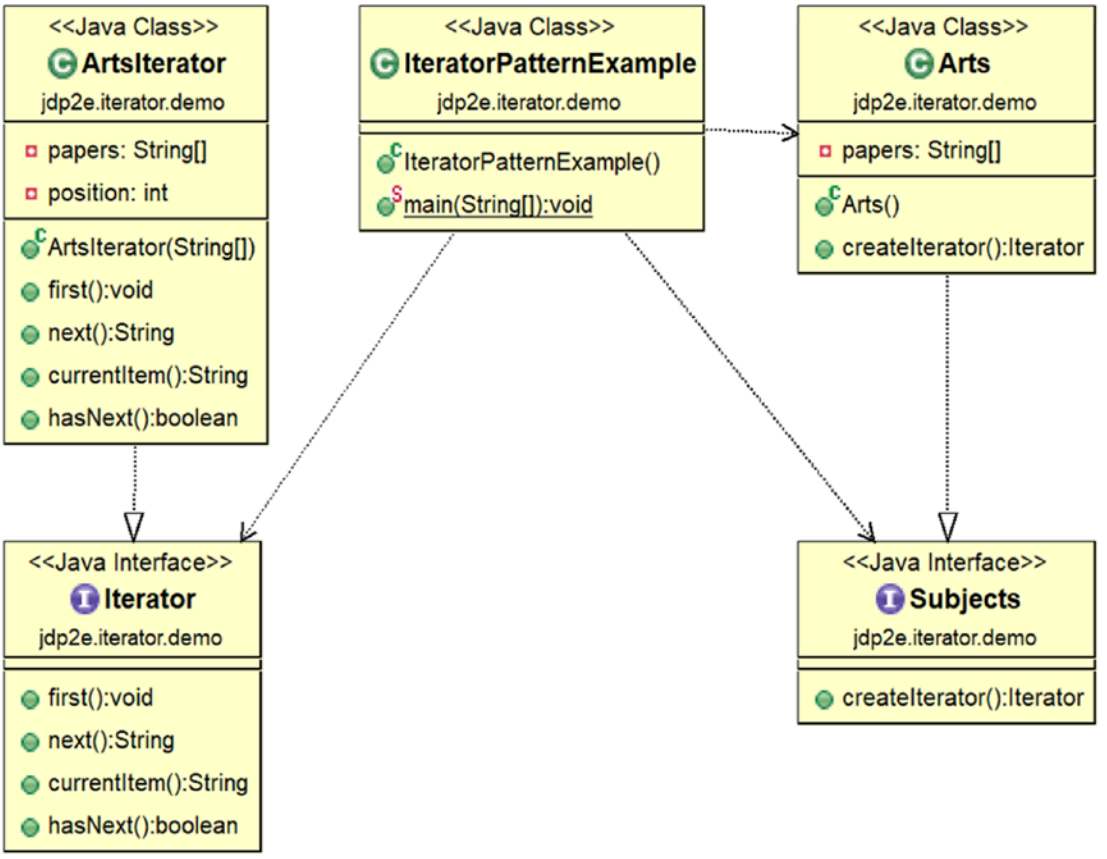
Conventions Used in This Book

- All the output and code in the book follow the same font and structure. To draw your attention, in some places, I made it bold, like the following.

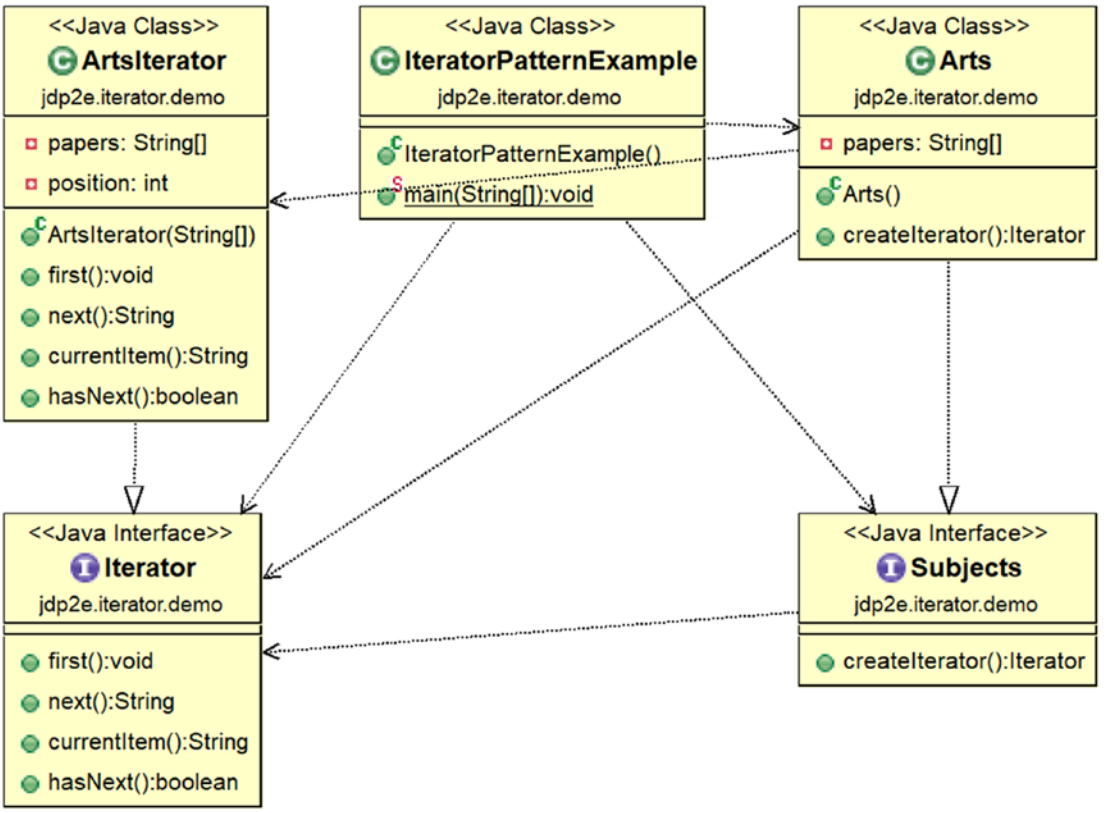
```
***Mediator Pattern Demo***
At present, registered employees are:
Amit
Sohel
Raghu
Communication starts among participants...
Amit posts: Hi Sohel, can we discuss the mediator pattern? Last
message posted at 2018-09-09T17:41:21.868
Sohel posts: Hi Amit, yup, we can discuss now. Last message posted
at 2018-09-09T17:41:23.369
Raghu posts: Please get back to work quickly. Last message posted
at 2018-09-09T17:41:24.870
```

An outsider named Jack trying to send some messages.

- In some cases, to present a cleaner class diagram and focus on the important parts, the less important dependencies are not shown. For example, consider the diagram presented in Chapter 18.


<<Java Class>>
ArtsIterator
jdp2e.iterator.demo
papers: String[]
position: int
ArtsIterator(String[])
first():void
next():String
currentItem():String
hasNext():boolean
<<Java Class>>
IteratorPatternExample
jdp2e.iterator.demo
IteratorPatternExample()
main(String[]):void
<<Java Class>>
Arts
jdp2e.iterator.demo
papers: String[]
Arts()
createIterator():Iterator
<<Java Interface>>
Iterator
jdp2e.iterator.demo
first():void
next():String
currentItem():String
hasNext():boolean
<<Java Interface>>
Subjects
jdp2e.iterator.demo
createIterator():Iterator

But if I need to show all the dependencies, it may look like the following.



You can see that the later one is much more complex and difficult to understand. For the ObjectAid class diagrams in Eclipse, you can always show these dependencies by selecting an element in the diagram, right-clicking and selecting Add ➤ Dependencies.

- I like to put curly braces on a new line. and I love to see a method body like the following.

```
public void myFunction()
{
//Some code
}
Instead of the following:
public void myFunction(){
//Some code
}
```

I used the same format for most of the methods, except main() methods.