

# Preface

---

*“Today, we’re at the cusp of a major shift in computing. The intersection of AI and accelerated computing is set to redefine the future.”*—Jensen Huang, NVIDIA CEO<sup>1</sup>

Welcome to *Java for Programmers with Generative AI, Fifth Edition*. In this Preface, we present the “soul of the book.” This edition’s key theme is **leverage**. You’ll learn Java programming faster, broader and more deeply with a trio of the latest complementary pedagogic methodologies. You’ll do more powerful things than with previous learning approaches and be better prepared to meet the programming and software development challenges you’ll face in your professional career:

- **Powerful: We use code to teach code.** You’ll learn with our traditional **live-code approach**. You’ll **leverage your learning experience**, mastering Java’s basics by immersing yourself in hundreds of complete working code examples with meaningful outputs.
- **More powerful: We use objects to teach objects.** New in this edition, you’ll use our **objects-natural approach** to learn Java object-oriented programming. You’ll **leverage your ability to write more significant programs** by using richly functional, preexisting classes from Java’s extensive libraries and other open-source libraries on repository sites like `github.com`.
- **Most powerful: We use generative AI to teach generative AI.** Also new in this edition, you’ll master **genAI**—the key set of artificial-intelligence technologies that **leverage your creative capabilities**. In the examples and genAI exercises, you’ll use prompts and build programs that produce original text, Java code, images, audio, speech, music and even video!

## A Personal Anecdote from Co-Author Harvey Deitel

In the 1970s, a friend suggested I teach an intro-to-computing course at a local college. In the segment on the hierarchy of programming-language types from **low-level machine languages** to **intermediate-level assembly languages** to **high-level languages**, each of which I programmed in, I explained that programming becomes more productive as you move up this hierarchy. You get further from the raw computer-hardware details and closer to the application domain, where you think in terms of the real-world problems you’re trying to solve. Individual statements in these languages become more powerful up the hierarchy, **leveraging your ability to solve more challenging problems faster while producing better-engineered solutions**.

---

1. Jensen Huang, NVIDIA COMPUTEX 2024 Keynote, June 2, 2024. Accessed March 2, 2025. <https://www.youtube.com/watch?v=pKXDVsWZmUU>.

The students were curious, asking where all this was headed. I said, “It seems reasonable to predict that even higher-level programming technologies will appear, which will help people solve ever more challenging real-world problems faster and better. As the programming statements you write become more powerful, someday you’ll be programming so close to the application domain that you’ll simply **tell the computer *what* you need rather than providing the increasingly complex details of *how* to build it—and the computer will produce the software for you!**”

That was in the 1970s. Then, in the 1980s and 1990s, object-oriented programming—Java’s key programming model—exploded onto the scene. Objects come from “blueprints” called classes. Almost any noun or concept can be represented by a class. There are, of course, vast numbers of those in the real world. Programmers began building extensive class libraries, **further leveraging the software-development process**. When the early versions of Java appeared in the mid-1990s, notably, they were free and accompanied by hundreds of powerful pre-built classes, making it easier for programmers to quickly build significant applications in key domains like graphics, GUI, concurrency, Internet networking, data structures, database and much more. Many of these classes had hundreds or thousands of lines of code, but to create an object of a class, a programmer typically had to write only one straightforward line of code, then use a few brief statements to make that object “strut its stuff.” This represented a **very-high-level programming capability** even closer to the problem domain.

All the while, the field of AI was evolving quickly from its roots in the 1950s. However, it wasn’t until Google’s 2017 research paper “Attention is All You Need”<sup>2</sup> defined the transformer model that the current AI boom took off. This is the root of today’s large language models, which power the genAI chatbots and tools you’ll use throughout this book.

In November 2022, OpenAI released ChatGPT, reaching a million users in five days and 100 million in two months. **No other application had ever become so widely used that quickly while profoundly enhancing users’ productivity and leveraging their ability to make computers do really interesting and powerful things.**

We have integrated genAI throughout this book. Most importantly for Java developers, genAIs remarkably can write, document, explain, debug, correct, critique, tune and enhance the performance of Java code. GenAIs are not perfect—they make mistakes and sometimes “hallucinate”—so you must monitor them carefully. We’ll demonstrate these coding capabilities in the approximately 600 genAI prompting exercises and examples throughout the book. GenAI has created an **ultra-high-level programming capability** that will leverage even further your Java learning experience and ability to produce robust, top-quality Java software quickly, conveniently and economically.

## An Extraordinary Range of Real-World Java Applications

In this book, we provide a friendly, contemporary, code-intensive, case-study-oriented introduction to Java, one of the world’s most popular programming languages.<sup>3,4</sup> Java is popular for a wide range of applications because it’s platform-independent, robust and has

2. Ashish Vaswani et al., “Attention Is All You Need.” arXiv preprint, arXiv:1706.03762, 2017. Accessed January 31, 2025. <https://arxiv.org/abs/1706.03762>.
3. PYPL PopularitY of Programming Language Index for January 2025. Accessed January 18, 2025. <https://pyp1.github.io/PYPL.html>.
4. Tiobe Index for January 2025. Accessed January 18, 2025. <https://www.tiobe.com/tiobe-index/>.

extensive standard and third-party class library support. It's used to build software for everything from the smallest Internet of Things (IoT) devices (like sensors connected to the Internet worldwide) to the largest cloud-based enterprise computing platforms. Some interesting uses of Java include:<sup>5</sup>

- Minecraft (one of the all-time most popular video games),<sup>6</sup>
- financial trading platforms (e.g., the LSEG Real-Time SDK<sup>7</sup>)
- Federal Aviation Administration (FAA) APIs for flight and airport information,<sup>8</sup>
- Android mobile app development<sup>9</sup>—Android is the leading mobile device operating system, with a 73.5% market share worldwide,<sup>10</sup>
- NASA Mars rover software<sup>11</sup> and
- tools for analyzing genetic DNA and RNA sequencing data.<sup>12</sup>

Other Java uses listed by the popular genAI chatbots ChatGPT,<sup>13</sup> Gemini,<sup>14</sup> Claude<sup>15</sup> and Perplexity that you'll use throughout this book<sup>16</sup> include:

- desktop GUI applications (as we do beginning with JavaFX in Chapters 15–17),
- cross-platform 2D and 3D game-development libraries (such as FXGL, which we will cover in a blog post),
- embedded systems (such as smart cards, Blu-ray players, cable set-top boxes, printers and automotive “infotainment” systems),
- scientific and educational software,
- web-based applications (such as the Spring and Spring Boot frameworks),
- high-performance server-side software,

- 
5. *ChatGPT* response to “List interesting things Java has been used for, like the Mars rover.” January 15, 2025. <https://chatgpt.com>. We edited the response.
  6. “Minecraft,” Wikipedia. Last modified January 20, 2025. Accessed January 20, 2025. <https://en.wikipedia.org/wiki/Minecraft>.
  7. LSEG Developers. “Real-Time SDK for Java.” Accessed January 20, 2025. <https://developers.lseg.com/en/api-catalog/real-time-opnsrc/rt-sdk-java>.
  8. Federal Aviation Administration, GitHub Repository. Accessed January 20, 2025, <https://github.com/Federal-Aviation-Administration>.
  9. “Welcome to Android Developers.” Accessed January 15, 2025. <https://developer.android.com/>.
  10. “Mobile Operating System Market Share Worldwide.” StatCounter. Accessed January 20, 2025, <https://gs.statcounter.com/os-market-share/mobile/worldwide>.
  11. Ross, Philip. “Java Runs Remote-Controlled Mars Rover.” CNET. Java 16, 2004. Accessed January 20, 2025. <https://www.cnet.com/tech/tech-industry/java-runs-remote-controlled-mars-rover/>.
  12. Broad Institute. “Getting Started with GATK4.” GATK Documentation. Updated July 20, 2024. Accessed January 20, 2025. <https://gatk.broadinstitute.org/hc/en-us/articles/360036194592-Getting-started-with-GATK4>.
  13. *ChatGPT* response to “Give me 100 words on what kinds of apps Java is popular for.” January 16, 2025. <https://chatgpt.com>. We edited the response.
  14. *Gemini* response to “Give me 100 words on what kinds of apps Java is popular for.” January 16, 2025. <https://gemini.google.com/app>. We edited the response.
  15. *Claude* response to “Give me 100 words on what kinds of apps Java is popular for.” January 16, 2025. <https://claude.ai>. We edited the response.
  16. *Perplexity* response to “Give me 100 words on what kinds of apps Java is popular for.” January 16, 2025. <https://perplexity.ai>. We edited the response.

- enterprise solutions for banking and finance,
- large-scale distributed systems,
- robotics, and
- big data technologies (like Hadoop that store and process vast amounts of data).

## Target Audiences

The book’s modular architecture (see the “**High-Level Overview**” below) makes it appropriate for several audiences:

- Introductory and intermediate professional Java programming training courses based on the curriculum recommendations of the ACM and IEEE professional societies.<sup>17</sup>
- Programmers experienced in other languages who must learn Java for use in upcoming projects.
- Programmers who know some Java and need to learn the latest Java features and idioms to prepare for upcoming projects.

It will also help you prepare for most topics covered by the **Oracle Java SE (Standard Edition) Developer Professional certification exam**.<sup>18,19</sup>

## Live-Code Approach

At the heart of the book is the Deitel signature **live-code approach**. Rather than code snippets, you’ll learn Java hands-on from 200+ fully coded, real-world examples and case studies with live outputs drawn from computer science, data science, AI and other fields. Read the Before You Begin section that follows this Preface to learn how to download the code examples and set up your Windows or macOS computer to run them.

Chapter 1’s Test-Drive (Section 1.7) shows how to compile and run the code examples with the free, open-source OpenJDK version of the Java Development Kit. **Executing each program in parallel with reading the text will make your Java learning experience “come alive.”** We also provide test-drives of the following popular Java developer tools at <https://deitel.com/javafp5>:

- JetBrains IntelliJ IDEA Community Edition
- The Eclipse IDE for Java Developers
- Microsoft Visual Studio Code

---

17. ACM/IEEE (Assoc. Comput. Mach./Inst. Electr. Electron. Eng.). 2023. *Computer Science Curricula 2023* (New York: ACM). Accessed January 18, 2025.

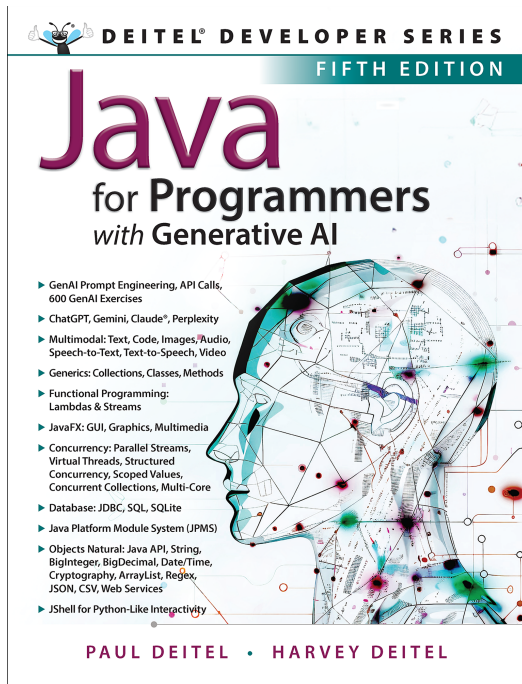
<https://csed.acm.org/wp-content/uploads/2023/03/Version-Beta-v2.pdf>.

18. Jeanne Boyarsky and Scott Selikoff, *OCP Oracle Certified Professional Java SE 21 Developer Study Guide* (New York: Wiley, 2024). [Jeanne Boyarsky was a technical reviewer on *Java for Programmers with Generative AI*, 5/e.]

19. “Exam: Java SE 21 Developer Professional (1Z0-830).” Accessed January 18, 2025.  
<https://mylearn.oracle.com/ou/exam/java-se-21-developer-professional-1z0-830/40805/139080/220555>.

# Java for Programmers with Generative AI, 5/e

## High-Level Overview



### Programming Paradigms

**Procedural** (Chapters 2–7), **object-oriented** (Chapters 8–10), **generic** (Chapters 12 and 13), **functional** (Chapter 14), **concurrent** (Chapter 18) and **genAI-enhanced** (Chapter 19).

### Part I: Introduction

**Chapter 1, Intro, Test-Driving a Java Application, and Generative AI**—Internet and Web. “Cloud.” Internet of Things (IoT). Metaverse. Intro to AI/genAI. Test-driving the Java Development Kit (JDK). **27 genAI prompt exercises.**

**Chapter 2, Intro to Java Programming**—Java fundamentals. Input/output. Primitive types. Arithmetic. Decision making. **Objects-natural (ON) case study: String class.** **12 genAI prompt exercises.**

### Part 2: Additional Programming Fundamentals

**Chapter 3, Control Statements: Part I**—if, if/else, while statements. Counter-controlled/sentinel-controlled iteration. Nested control statements. **ON case study: BigInteger for supersized integers.** **16 genAI prompt exercises.**

**Chapter 4, Control Statements: Part 2**—for, do/while, switch, break, continue statements. Logical operators. **ON case study: BigDecimal for precise monetary calculations.** **13 genAI prompts.**

**Chapter 5, Methods**—Custom methods. Random numbers. Simulation. **ON case study: Date/Time API.** **16 genAI prompt exercises.**

**Chapter 6, Arrays and ArrayLists**—Data structures. Built-in arrays. **ON case study: ArrayList for dynamically resizable arrays.** **28 genAI prompt exercises.**

**Chapter 7, Strings, NLP and Regex: Generative AI Foundations**—String, Character, StringBuilder classes. Intro to natural language processing (NLP). **ON case studies: 1. Regular expressions for pattern matching in text. 2. Securing data with AES private-key cryptography.** **22 genAI prompt exercises.**

### Part 3: Object-Oriented Programming

**Chapter 8, Real-World Modeling with Custom Classes**—Crafting valuable classes. Account class case study. Card-shuffling-and-dealing simulation. Time class case study. Controlling class-member access. Constructors. Set/get methods. Data validation. Throwing exceptions. static and final class members. record classes. **33 genAI prompt exercises.**

**Chapter 9, Real-World Modeling with Inheritance, Polymorphism & Interfaces**—Inheritance hierarchies. Runtime polymorphism. Inheritance *is-a* relationship vs. composition *has-a* relationship. Interfaces. Programming to an interface, not an implementation. Dependency injection. sealed classes and interfaces. **31 genAI prompt exercises.**

**Chapter 10, Exception Handling: A Deeper Look**—try statement to catch and handle exceptions. Throwable hierarchy. Checked vs. unchecked exceptions. try-with-resources statement. **30 genAI prompt exercises.**

**Chapter 11, Files, I/O Streams, JSON Serialization & CSV Files**—Data persistence. Text vs. binary files. NIO classes. Retrieving file/directory info. Formatter class. JSON (JavaScript Object Notation) serialization. Invoke an OpenWeatherMap web service with `java.net.http` features. CSV (comma-separated values) file format. Titanic disaster CSV dataset and basic data analytics. **ON case study: Securing data and protecting user privacy with RSA public-key cryptography.** **38 genAI prompt exercises.**

## Part 4: Data Structures, Generic Collections, Lambdas and Streams

**Chapter 12, Generic Collections**—Prepackaged data structures from the Java collections framework. Pre-built generic data structures. **30 genAI prompt exercises.**

**Chapter 13, Generic Classes and Methods: A Deeper Look**—Implement a custom generic method and a custom generic class. Compile-time type safety. **24 genAI prompt exercises.**

**Chapter 14, Functional Programming with Lambdas & Streams**—Use lambdas and stream pipelines to write certain kinds of programs faster, simpler, more concisely and hopefully with fewer bugs than previous techniques. Focus on immutability. **Chapter 18, Concurrency: Platform Threads to Virtual Threads**, demonstrates parallelizing stream pipelines to enhance performance on multi-core architectures. **37 genAI prompt exercises.** [Chapter 22 can be covered here.]

## Part 5: JavaFX Graphical User Interfaces, Graphics and Multimedia

**Chapter 15, JavaFX Graphical User Interfaces: Part I**—Scene Builder for simple drag-and-drop GUI design. Layouts. Controls. Event handling. Welcome app. Tip Calculator app. **13 genAI prompt exercises.**

**Chapter 16, JavaFX GUI: Part 2**—Additional layouts and controls. Mouse, `RadioButton` and property-change events. Data binding. Customizing a control's appearance. `FileChooser` and `DirectoryChooser` dialogs. **21 genAI prompt exercises.**

**Chapter 17, JavaFX Graphics and Multimedia**—Cascading Style Sheets (CSS) for customizing JavaFX nodes' appearance and text fonts. 2D and 3D shapes. Move, rotate and scale node transformations. Display, play and pause video. Incrementally change properties with `Transition` and `Timeline` animations. Frame-by-frame animations with `AnimationTimer`. CSS transitions for simplified animation effects. Canvas for drawing with the mouse. **31 genAI prompt exercises.**

## Part 6: Advanced Topics

**Chapter 18, Concurrency: Platform Threads to Virtual Threads**—Multi-core programming. Create and manage multiple tasks. Performance case studies. Profile sequential vs. parallel sorting with Date/Time APIs. Prepackaged parallel algorithms. Classic producer-consumer relationship. Easier-to-use, less error-prone, higher-level concurrency features. Project Loom: Lightweight virtual threads, structured concurrency, scoped values. **51 genAI prompt exercises.**

**Chapter 19, Building API-Based Java Generative AI Applications**—Multimodal genAIs that understand text, code, image, speech and video inputs and generate text,

code, images, speech and video. Work with OpenAI APIs to create apps that summarize text, analyze text for sentiment, create accessible descriptions of images, detect a text's language and translate text among languages, generate Java code, perform named entity recognition on text and obtain structured JSON outputs, transcribe speech to text, synthesize speech from text, generate closed captions for a video and more. **21 genAI prompt exercises and 73 API-based programming projects.**

**Chapter 20, Accessing Databases with JDBC and SQLite**—Storing persistent data in databases. SQLite database management system. Structured Query Language (SQL) CRUD (create, read, update, delete) operations. JDBC API. Connecting to a database. Retrieve data from a database. JavaFX `TableView`. JDBC `PreparedStatement`s. **19 genAI prompt exercises.**

**Chapter 21, Java Platform Module System**—Create custom packages and modules. Declare module dependencies. Specify which packages a module exports. Define the services a module offers or consumes. Reflection and enabling reflective access. Bundling resources with modules. **31 genAI prompt exercises.**

**Chapter 22, Recursion and Big O**—Recursive and iterative factorial calculations. Recursive Fibonacci calculations. JavaFX app that recursively creates a “feather fractal.” Text-based algorithm visualizations. Rich selection of recursion exercises. **11 genAI prompt exercises.** [Can be covered at the end of Part 4: Data Structures.]

## Appendices

**Appendix A, Introduction to JShell for Interactive Java**—Java's friendly, command-line REPL (read-evaluate-print-loop) for exploration, discovery and experimentation. Like having Python's interactivity in Java. Many sections can be read in conjunction with the book's early chapters.

**Appendix B, Formatted Output**—formatting features for output and for `Strings` in memory. Summarizes the formatting features we discuss throughout the book and introduces additional capabilities.

**Appendix C, Number Systems**—Introduces binary (base 2), octal (base 8), decimal (base 10) and hexadecimal (base 16) number systems.

## Possible Topics for the Deitel.com Blog

Sequenced collections. Flexible constructor bodies. Bit manipulation. Labeled `break` and `continue` statements. JavaFX `Subscription` API. Intro to basic JavaFX game development with FXGL (FX Game Library), animation, collision detection, particle effects and more. We'll link our Java-related blog posts to the book's webpage at <https://deitel.com/javafp5>



## “Objects-Natural” Approach

Object-oriented programming books have traditionally used a “late objects” or an “early objects” teaching approach. What’s really “late” or “early” in these books is not “objects.” It’s teaching how to develop custom classes—the “blueprints” from which objects are built. We’ve written books using both approaches in popular object-oriented programming languages.

### What Is “Objects Natural”?

As we wrote our textbook *Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and the Cloud*,<sup>20</sup> we noticed that although our presentation fits the “late objects” model, it is actually something more—and that something is special. We call it the “**objects-natural approach**,” and we’re now using it in each object-oriented programming language textbook and professional book we write.

Like “late objects,” our objects-natural approach begins with programming fundamentals such as data types, variables, operators, control statements, methods, arrays and strings in the early chapters—all before you develop your own *custom* classes. However, with the objects-natural approach, you’ll get lots of practice using powerful *existing* classes that do significant things, quickly creating objects of those classes (typically with one line of code) and telling them to “strut their stuff” with a minimal number of simple Java statements. This is one of the most compelling aspects of working with a mature object-oriented language like Java. And you’ll do this long before you create and use custom Java classes in Chapter 8 and inheritance, polymorphism and interfaces in Chapter 9.

### An Abundance of Free Classes

We emphasize using the massive number of valuable free classes in the Java ecosystem. These typically come from:

- the **Java Application Programming Interface (API)**—Java’s standard library—and
- free third-party Java libraries, often created by the open-source community.

We encourage you to view lots of Java code on sites like GitHub. Reading other programmers’ code is a great way to learn.

## Generative AI (GenAI) Approach<sup>21</sup>

Leveraging genAI was integral in developing *Java for Programmers with Generative AI, 5/e*. It will also be essential to how you interact with the material. Here, we outline our approach to integrating genAI throughout the book, its potential to profoundly enhance

20. Deitel, Paul, and Harvey Deitel. *Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and the Cloud*. Upper Saddle River, NJ: Pearson, 2020.

21. *ChatGPT* (<https://chatgpt.com>), *Gemini* (<https://gemini.google.com>), *Claude* (<https://claude.ai>) and *Perplexity* (<https://perplexity.ai>) responses to the prompt, “The following is a list of miscellaneous notes on how we used genAIs when working on our new Java programming textbook and how our readers will use genAI while reading the book. Turn these notes into useful flowing text, reorganizing the points as necessary and removing redundancy.” January 1, 2025. To demonstrate the power of genAIs, we followed the preceding prompt with a lengthy (and sometimes repetitive) list of our raw author notes for this section, asking the genAIs to organize our notes into smooth, flowing paragraphs. We then edited the final presentation.

Java education, and **how to leverage these tools effectively while maintaining a cautious and critical mindset.**

While writing the book, we integrated genAI not as a replacement for our work as authors but as a powerful enhancement tool. Similarly, you should use genAI to augment your learning and software-development skills, not to replace them. **Our experience with genAI has been both enlightening and entertaining—it’s like having a diverse team of experts available to assist you 24/7, though one that requires careful supervision.**

### Two Approaches to Using GenAI

We integrate genAI using two approaches:

- **Prompt engineering:** Outside Chapter 19, we focus on prompt engineering to optimize the interaction with genAI tools. Think of this as learning a new and different type of programming language. With prompt engineering, you craft precise instructions or queries to guide genAIs in producing the required responses. You can also iteratively tune your prompts, refining and adjusting them to help the genAIs produce more accurate and relevant results—**the better your prompts, the better the results.** It takes lots of experience to learn how to create great prompts—you’ll get that experience in this book.
- **Java code using AI APIs plus prompt engineering:** Chapter 19 combines prompt engineering and API-based code development, emphasizing how genAI can help you build and tune complete Java applications. We use the OpenAI APIs.

Chapter 19’s examples and exercises dive deeper into using **genAI for code generation and analysis** as we build API-based Java genAI applications. You’ll learn to craft effective prompts to produce code snippets, debug and tune your programs, and create complete Java applications.

### GenAI Tools We Used

Our work relied on many genAIs, including four primary genAI chatbots:

- OpenAI ChatGPT
- Google Gemini
- Anthropic Claude
- Perplexity

We used them all for each task because they produced some overlapping and some different results—each added value. In fact, the same genAI often yielded different results when we gave it the same prompt, even if only moments later. Each has unique strengths and weaknesses. GenAIs constantly evolve, so capabilities and limitations will change over time, most likely improving dramatically.

We also used **OpenAI’s Dall-E for image generation** (Dall-E is now integrated into ChatGPT) and **Sora for video generation.** Many genAIs offer free trials or tiers, which can give you good results but might be less capable than their paid counterparts.

### Role of GenAIs in Education

GenAIs will play a crucial role in professional and college education in general and especially in programming-language education. They serve as an always-available team of virtual experts to assist you, accessible at little or no cost.



Organizations and educational institutions remain divided on whether to allow genAI usage. Some embrace it as we do in this book. Some have banned it. Others permit it, with safeguards. We encourage you to become familiar with these tools, which are likely to be indispensable throughout your career. However, comply carefully with your organization's or institution's policies.

### GenAIs for Enhancing Learning Experiences

GenAIs can enhance learning experiences by:

- Providing multiple perspectives on complex concepts.
- Offering interactive debugging assistance.
- Helping identify areas for review when concepts are unclear.
- Supporting various learning styles through different explanatory approaches.
- Offering personalized feedback on the code you write.

We encourage you to do the work, using genAIs to enhance your learning experience. Our Generative AI exercise sections help you:

- Learn prompt engineering—crafting and refining prompts to achieve optimal results.
- Learn what genAIs can do by trying lots of prompts.
- Learn the level of detail with which each genAI responds to prompts.
- Discover new ways to solve problems.
- Compare the strengths and weaknesses of different genAIs.
- Understand that genAIs make mistakes and sometimes hallucinate, and how to detect and address those errors.
- Be aware that genAIs can produce inaccurate results.
- Get clarification on points raised in the book.
- Enhance your knowledge when you want more detail.

### How We Used GenAIs

We integrated genAI into our authoring efforts—not to write the manuscript (except by design in cited instances) or code examples but to enhance research, development, tuning the writing and proofreading. We used genAIs to:

- Refine our genAI prompting skills to get better results.
- Brainstorm.
- Extract key points from this book's content and use them to draft PowerPoint slides for instructors to prepare and deliver their classes.
- Verify our content.
- Check that we used current Java programming idioms and clean code<sup>22</sup> guidelines appropriate for a book at this level.

---

22. Martin, Robert C. *Clean Code: A Handbook of Agile Software Craftsmanship*. Upper Saddle River, NJ: Addison-Wesley, 2009.

- Check our grammar and writing smoothness and clarity.
- Generate multiple-choice question drafts from our content.
- Draft exercise solutions using only the techniques we presented to a given point in the book.
- Draft marketing text directly from our content.
- Draft blog posts directly from our content.
- Refactor our code to conform to our coding conventions, such as code indentation and comment relevance and density.
- Refactor code (e.g., renaming variables, methods and classes, extracting repetitive code into separate functions, etc.).

Effective prompt engineering was key in most cases, and revisiting an ongoing chat with a carefully tuned prompt often led to dramatically improved results.

### Guidelines for GenAI Users

We encourage you to use genAIs as a learning tool rather than just an answer generator. Just as programming mastery requires understanding language and library capabilities, effective genAI use requires developing skills with different prompts. You should:

- Keep track of your prompts and the quality of their results. This is crucial for learning what works and what doesn't. This will also help you identify the nuances of different genAI platforms. You'll build a repertoire of valuable prompts and anticipated results that will leverage future programming efforts.
- Experiment with multiple genAIs—including new ones as they appear—and compare the results to understand their diverse capabilities.
- Experiment with different prompting strategies to see which ones help you achieve your desired outcomes. Different programming challenges can require different prompting strategies.
- **If genAI results are unsatisfactory, tune your prompt and try again.** For example, you'll see that some genAIs emphasize producing smooth, professional quality writing, while others dig deeper into the subject matter.
- Use genAIs for **discovery**—finding new ideas, insights and patterns that might not initially be apparent. GenAIs offer wide-ranging perspectives, analyzing vast amounts of data and synthesizing information into actionable outputs that can enhance creativity or inform decisions. GenAI discovery can enhance your creativity and uncover possibilities beyond your expertise, helping you grow your knowledge and wisdom.<sup>23</sup>

### Hallucinations and Errors

GenAIs have limitations. They've been known to "hallucinate," confidently providing fabricated answers. When you ask them to perform tasks, sometimes the results contain errors, such as Java code that does not work. We've included successful and unsuccessful

---

23. *ChatGPT* response to "What does it mean to use generative AIs for discovery?" January 18, 2025. <https://chatgpt.com>. We edited the response.

genAI interactions in the book. For instance, while genAIs helped us enhance our Number Systems Appendix with innovative applications of hexadecimal, we also encountered situations where they struggled with seemingly straightforward tasks. As early as Chapter 2, we encountered genAI failures when asking them to write simple Java statements. We deliberately included some of these failure cases as valuable teaching moments, demonstrating that genAIs are fallible, so you must use them cautiously. **This underscores the importance of maintaining human oversight and verification, especially in crucial business-critical and mission-critical situations.**

### Why We Use Some Paid Services

While free or free-tier genAIs are valuable, paid services often offer more robust capabilities, availability and reliability. Many genAIs are paid services due to the sheer cost of training the large language models that provide the genAIs' powerful capabilities. For example, training OpenAI's GPT-4 and Google's Gemini cost \$78 million and \$191 million, respectively—not including the salaries of the people developing the large language models and other overhead costs.<sup>24</sup> We encourage you to use free or low-cost tools when available. You can even ask genAIs to suggest the best free versions. However, professional developers will often choose paid tools to get the best results.

### What We're Trying to Accomplish

We aim to prepare you for a future in which genAIs will be indispensable in your personal life and professional career. By integrating genAI examples and exercises throughout the book, we aim to help you become comfortable using these tools and understand their advantages and pitfalls. Developing a healthy sense of caution while exploring genAI capabilities will empower you to use them effectively.

### Chapter 19's API-Based, Complete, Live-Code Java GenAI App Case Studies

Chapter 19 presents the following fully implemented code examples that programmatically interact with OpenAI's APIs:

- **Text Summarization** (Section 19.4.1): Asks the OpenAI chat API for a summary abstract paragraph of a Deitel video transcript and a list of the transcript's key points. Text responses like these are known as **chat completions**.
- **Sentiment Analysis** (Section 19.4.2): Asks the OpenAI chat API for the sentiment of a transcript and an explanation of how the genAI came to its conclusion.
- **Accessible Image Descriptions** (Section 19.4.3): This multimodal application uploads images and asks the OpenAI chat API for detailed descriptions of the for-loop UML diagram in Section 4.3 and a scenic Aruba beach photo.
- **Language Detection and Translation** (Section 19.4.4): Asks the OpenAI chat API to translate text to a specified target language—it **autodetects the source language**. We translate text from English to Spanish and Japanese, then from Spanish and Japanese back to English. Many of today's genAIs can translate among scores of languages.

---

24. Stanford Institute for Human-Centered Artificial Intelligence (HAI), AI Index Report 2024, May 2024. Accessed March 8, 2025.  
[https://hai-production.s3.amazonaws.com/files/hai\\_ai-index-report-2024-smaller2.pdf](https://hai-production.s3.amazonaws.com/files/hai_ai-index-report-2024-smaller2.pdf).

- **Java Code Generation** (Section 19.4.5): Asks the OpenAI chat API to generate Java code for a simulation summarizing millions of six-sided die rolls and displaying the results.
- **Named-Entity Recognition and Structured Outputs** (Section 19.4.6): Demonstrates producing a structured JSON response in your specified format. This app performs a natural language processing (NLP) task called named entity recognition (NER), which attempts to locate and categorize text mentions like dates, times, quantities, places, people, things, organizations, etc. To make the results easier to process, we provide the API with the exact JSON response format we wish to receive, then use the **Jackson open-source library** (Chapter 11) to process the results.
- **Speech-to-Text** (Section 19.5.1): This multimodal application (speech audio and text) uploads a speech audio file to the **OpenAI Whisper model**, which transcribes it to text and returns the text transcription.
- **Text-to-Speech** (Section 19.5.2): This multimodal application (text and speech audio) uploads text to the **OpenAI TTS-1 model** to synthesize speech from the text. This example converts English, Spanish and Japanese text samples into speech using the same voice.
- **Image Generation** (Section 19.6): Demonstrates code that asks the **Dall-E APIs** to generate images of a Havanese dog in the styles of a Japanese anime character in neon colors against a black background, Vincent Van Gogh and Leonardo DaVinci.
- **Creating Closed Captions for a Video** (Section 19.7.1): This multimodal accessibility application (speech audio, text and video playback) uploads an audio track from a video and transcribes it into a JSON format containing timestamps and corresponding transcribed text. The app then requests a chat completion that generates **closed captions in VTT format**.<sup>25</sup> We show VTT-formatted text and screen captures displaying captions over the corresponding video.
- **Moderation** (Section 19.8): Asks the **OpenAI moderation API** to evaluate text prompts for harmful or inappropriate text, such as harassment, hate speech, violence, self-harm and sexually explicit content.

### GenAI Prompt Exercises

We fed the complete list of all the book’s approximately 600 genAI exercises (a 100+ page PDF) to ChatGPT, Gemini, Claude and Perplexity, asking them to categorize the kinds of things we do in those exercises. Next, we fed all their categorized lists to the four genAIs, asking them to summarize the summaries, and we chose the best one—Claude in this case:

- **Code Generation and Implementation**—Writing new Java programs from specifications. Implementing specific features, algorithms and APIs. Creating test programs and practical applications. Generating solutions for both basic and advanced programming tasks.

---

25. “WebVTT.” Wikipedia. Last modified December 31, 2024. Accessed March 1, 2025.  
<https://en.wikipedia.org/wiki/WebVTT>.

- **Code Refactoring and Enhancement**—Modernizing code with current Java features (streams, enhanced loops). Improving code structure, readability, and maintainability. Converting between different approaches while maintaining functionality. Improving performance.
- **Educational Content**—Creating tutorials, exercises, and learning materials. Explaining complex concepts (polymorphism, immutability, etc.). Developing beginner-friendly programming exercises. Writing comprehensive documentation and guides.
- **Technical Analysis**—Analyzing code behavior and feature implementations. Comparing different approaches, tools, and frameworks. Evaluating trade-offs in design decisions. Breaking down complex technical concepts.
- **Best Practices and Standards**—Implementing coding standards and design patterns. Addressing security considerations. Optimizing performance. Following Java development best practices.
- **Technology Evaluation**—Comparing libraries, tools, and frameworks. Assessing the pros and cons of different approaches. Making informed technology choices. Exploring new features and updates.
- **Debugging and Error Handling**—Finding and fixing syntax and logical errors. Implementing exception handling. Improving fault tolerance. Preventing common pitfalls.
- **API and Library Integration**—Working with Java APIs and external libraries. Understanding API features and capabilities. Implementing integration techniques. Creating API documentation and tutorials.
- **Real-world Applications**—Developing practical use cases and industry applications. Creating interactive applications (GUIs, games, multimedia). Implementing real-world scenarios. Building sample projects.
- **Performance Optimization**—Analyzing and improving performance. Optimizing resource usage. Conducting benchmarks. Implementing efficiency improvements.
- **Creative Development**—Building multimodal applications. Creating visualizations. Generating test scenarios and sample data. Developing unique use cases.

### GenAI API-Based Java Programming Exercises

Chapter 19, *Building API-Based Java Generative AI Applications*, suggests challenging project exercises like creating genAI multimedia apps that can debate one another and using genAI to build and solve crossword puzzles. We fed Chapter 19's 90+ exercises into the genAIs, asking for a categorized summary of them, then summarized the summaries. Here's what they produced:

- **Multimodal Applications**—Combining text, image, audio, and video capabilities. Creating integrated experiences like interactive books. Developing multimedia educational content. Building comprehensive tools that leverage multiple AI modalities.
- **Text-Based Applications**—Document processing (indexing, summarization, exploration). Creative writing (story generation, poetry, debates). Language tools

(translation, tone rewriting). Professional document creation (resumes, presentations). Structured outputs.

- **Image Processing Applications**—Generative art and design (logos, fashion, floor plans). Technical visualization (UML diagrams). Image analysis and recognition.
- **Audio and Music Applications**—Speech processing (transcription, voice cloning). Music generation (MIDI, Magenta AI). Multilingual audio applications. Podcast and audio content analysis.
- **Educational Tools**—Programming tutors (Java, coding exercises). Subject-specific learning aids (math, arithmetic). Course content creation. Interactive educational experiences.
- **Gaming and Puzzle Applications**—Puzzle generators and solvers (Crosswords, Word Search). Interactive game development.
- **Video**—Investigating and experimenting with generative AI video creation tools.
- **Chatbot Development**—Character-based chat experiences. Specialized domain experts.
- **Research and Analysis Tools**—Medical applications (drug discovery, personalized medicine). AI capability exploration. Text detection and analysis. Educational research.
- **Creative Applications**—Children’s book creation. Interactive storytelling. Artistic content generation. Creative writing tools.
- **Practical Tools and Utilities**—Document generators. Translation services. Content summarizers. Professional tools (resume filters, presentation creators).

### Key Takeaways

- **Try multiple genAIs:** Compare their strengths and weaknesses.
- **Master prompt engineering:** Craft effective prompts for optimal results. Build a catalog of your best prompts.
- **Be cautious of hallucinations:** Always review genAI output critically.
- **Embrace the power of genAI:** Use it to enhance your Java learning experience, not to replace it.
- **Adhere to your organization’s genAI policies.**

## New and Updated Features

In the following sections, we discuss the key new features and updates we’ve made for *Java for Programmers with Generative AI, 5/e*, including:

1. **Objects-Natural Case Studies.** Chapter 1 presents a friendly introduction to object technology’s basic concepts and terminology. In the early chapters, you’ll create and use powerful objects of preexisting Java API classes to do significant things without knowing how to write Java classes in general or how those particular classes are implemented—and you’ll do this long before you create and use objects of your own custom classes in Chapter 8. We’ve added objects-natural case studies



on class `String` (Chapter 2), class `BigInteger` for supersized integers (Chapter 3), class `BigDecimal` for the precise monetary calculations required in business applications (Chapter 4), the Date/Time API (Chapter 5), class `ArrayList` (Chapter 6) for dynamically resizable arrays, regular expressions for locating patterns in text (Chapter 7), secret-key AES cryptography (Chapter 7) and public-key RSA cryptography (Chapter 11). Throughout the rest of the book, many other case studies use objects of Java API and open-source library classes extensively.

2. **Approximately 600 integrated generative AI (genAI) prompt exercises.** Most sections end with genAI prompt exercises (approximately 450) in which you'll interact with genAIs using prompt engineering. In addition, many chapters include end-of-chapter genAI prompt exercises (approximately 150). You'll feel like you have lots of Java experts at your side 24/7 (some receiving modest pay and some free) to answer your questions about the book's Java content, help you probe more deeply into topics of interest, and even help you write and debug Java code. Better yet, they're competing among themselves and a steady flow of new entries to offer you the best results while each rapidly improves.
3. **Chapter 19, Building API-Based Java Generative AI Applications,** demonstrates programmatically interacting with OpenAI's APIs for text summarization, sentiment analysis, describing images for accessibility, translating text among languages, generating Java code, named-entity recognition, transcribing speech to text, synthesizing speech from text, creating closed captions for video, image generation and much more in the chapter's 94 exercises.
4. **Covers the latest Java language features, library features and programming idioms.** To keep the book up to date as new versions of Java are released, we'll place new code examples and explanations on **our blog** at <https://deitel.com/blog> and link them to the book's webpage at <https://deitel.com/javafp5>.
5. **Streamlined contemporary treatment of object-oriented programming,** including record classes, sealed classes and interfaces, and a new case study on programming to an interface, not an implementation, focusing on composition and dependency injection.
6. **Enhanced file-processing coverage,** including using the **Jackson open-source library** to input and output data in the JSON and CSV formats that are so popular in today's data-science applications.
7. **Invoking popular web services** (such as **OpenWeatherMap**) with web networking capabilities provided by the **java.net.http** package.
8. **Special feature introducing data analytics with the CSV-format Titanic disaster dataset.**
9. **Special feature on cryptography,** including **secret-key AES cryptography** and **public-key RSA cryptography**. Cryptography is crucial for the rapidly growing interest in **computer privacy and security**.
10. **Special feature on simple data wrangling** steps used to prepare text for training natural language processing (NLP) and genAI models. Data wrangling is a key **data science** technology.

11. New Chapter 1 sections briefly discuss the **metaverse** and its related technologies—virtual reality (VR), augmented reality (AR), mixed reality, blockchain, cryptocurrency, nonfungible tokens (NFTs) and Web3—and **software development technologies, big data, and AI’s intersection with computer science and data science**.
12. Enhanced treatment of **JavaFX GUI, graphics, animation and video**—technologies that will be core to the emerging metaverse. The presentation includes an intro to JavaFX’s new CSS transitions for simplified animation effects.
13. A brief intro to **JavaFX game development** with the **open-source FXGL (FX Game Library)** in Chapter 17.
14. Flexible coverage of **JShell**, Java’s friendly, command-line REPL (read-evaluate-print-loop) environment for quickly exploring, discovering and experimenting with Java’s language features and libraries—it’s like having Python’s interactivity in Java.
15. **Concurrency, Parallelism and Multi-Core Performance** updates for key **Project Loom** technologies like **virtual threads** and the emerging **structured concurrency** and **scoped values**.
16. **Enhanced database treatment** that interactively introduces **Structured Query Language (SQL)** using the popular **SQLite database management system’s command-line tools**. The examples now use **SQLite databases**. We replaced the older Swing **JTable** with **JavaFX’s TableView**, which offers a nicer look-and-feel, better performance and more customization than **JTable**.
17. Many new and updated **contemporary examples**.
18. We now use **RandomGenerator** (from Java 17), which provides enhanced random-number generation capabilities and is preferred for simulations. We use **SecureRandom** in security scenarios, such as our cryptography examples.
19. Our **hundreds of programming tips** gleaned from our combined ten decades of programming, teaching and industry experience have been fully integrated into the regular text flow in this edition.
20. We added various application programming case studies.
21. We enhanced existing case studies and added new ones focusing on AI and data science, including simulations with random-number generation, natural language processing (NLP) and artificial intelligence via generative AI and heuristic programming.

## Pedagogy Features

### Syntax Coloring and Syntax Highlighting

In our e-books, we syntax color the Java code, similar to how most Java integrated development environments (IDEs) and code editors syntax color code. For the print version of the book, we use the following syntax highlighting:

```

comments appear like this
keywords appear like this
constants and literal values appear like this
all other code appears in this thin black font

```

### Using Fonts for Emphasis

We emphasize on-screen components in bold (e.g., the File menu) and Java program text in a fixed-width font (e.g., `int x = 5;`).

### Objectives

The chapter Objectives sections list the chapter’s goals.

### Illustrations/Figures

Abundant tables, line drawings, UML diagrams, programs and program outputs are included.

### Java as It’s Intended to Be Used

We use the terminology from the latest Java specification document (<https://docs.oracle.com/javase/specs/>) in preference to general programming terms. We also show Java as it’s intended to be used with a rich collection of application programming case studies, focusing on computer science, artificial intelligence, data science and many other fields.

### Keep It Topical

To “take the pulse” of modern Java, we read, browsed or watched thousands of current articles, research papers, white papers, books, documentation pieces, blog posts, forum posts, webinars and videos. We placed Google Alerts on hundreds of important Java-related, general computing, AI and data science topics.

### Hundreds of Contemporary Examples and GenAI Exercises

Our code examples and genAI exercises present current topics of interest in computing.

### 50+ Hours of Videos Available to O’Reilly Online Paid Subscribers

In our *Java Fundamentals with Paul Deitel, 3/e*, videos:

<https://learning.oreilly.com/course/java-fundamentals-with/9780135353134/>

co-author Paul Deitel discusses the material in the Before You Begin section, and explains and provides additional insights on most of the book’s programs. You can get a free 10-day subscription at <https://www.oreilly.com/start-trial/>.

### Performance

We focus on techniques and strategies for meeting the extraordinary performance needs of today’s applications.

### Data Experiences

In Chapter 11, you’ll work with real-world text data. You’ll read and analyze the Titanic Disaster dataset,<sup>26</sup> which is popular for introducing the field of **data analytics**. This dataset is stored in a **CSV (comma-separated values) file**, a format we introduce in Chapter 11. Also, see the “Data Science Overlaps with Computer Science” section later in this Preface.

---

26. “TitanicSurvival” dataset on <https://vincentarelbundock.github.io/Rdatasets>. Dataset authors: Frank E. Harrell, Jr., and Thomas Cason.

### Secure Java Programming

It's challenging to build industrial-strength systems that stand up to attacks from viruses and other forms of "malware." Today, via the Internet, such attacks can be instantaneous and global in scope. Building security into software from the beginning of the development cycle can significantly reduce vulnerabilities. We audited our book against the Oracle Secure Coding Guidelines for Java SE

<https://www.oracle.com/java/technologies/javase/seccodeguide.html>

and adhere to secure coding practices appropriate for a book at this level.

### Privacy

The ACM/IEEE's curricula recommendations<sup>27</sup> for Computer Science, Information Technology and Cybersecurity mention privacy hundreds of times. Every programmer must consider privacy issues and concerns. We discuss secret-key and public-key cryptography in Chapters 7 and 11, respectively.

### Ethics

The ACM/IEEE's curricula recommendations<sup>28</sup> for Computer Science, Information Technology and Cybersecurity mention ethics more than 100 times. In several Chapter 1 and Chapter 19 generative AI exercises, you'll focus on ethics issues via Internet search and generative AI research. You'll investigate privacy and ethical issues surrounding intelligent assistants, like Amazon Alexa, Apple Siri and generative AIs' voice capabilities. We'll also examine the excitement and controversies surrounding genAIs such as OpenAI's ChatGPT,<sup>29</sup> Dall-E<sup>30</sup> and many more.

### Programming Wisdom

Hundreds of programming tips are fully integrated into the regular text flow in this edition:

- **Good programming practices** call attention to techniques that help you produce clearer, more understandable and more maintainable programs.
- We point out **common programming errors** to reduce the likelihood you'll make them.
- **Performance tips** highlight opportunities to make your programs run faster or minimize memory use.
- **Software engineering observations** highlight architectural and design issues that affect the construction of software systems, especially large-scale systems.
- **Look-and-feel observations** (Chapters 15–17) highlight GUI conventions. These observations help you design attractive, user-friendly GUIs that conform to industry norms.

---

27. "Curricula Recommendations." Accessed January 18, 2025.

<https://www.acm.org/education/curricula-recommendations>.

28. "Curricula Recommendations." Accessed January 18, 2025.

<https://www.acm.org/education/curricula-recommendations>.

29. "Introducing ChatGPT." Accessed January 18, 2025. <https://openai.com/blog/chatgpt>.

30. "Dall-E 2." Accessed January 18, 2025. <https://openai.com/product/dall-e-2>.

## Tour of the Book

The “High-Level Overview” early in this Preface shows the book’s modular architecture. We recommend that you refer to that overview as you read this section.

The early chapters establish a solid foundation in Java fundamentals. The middle and later chapters introduce intermediate and advanced Java programming topics. We discuss six programming paradigms:

- procedural programming (Chapters 2–7)—**enhanced with our objects-natural approach**,
- object-oriented programming (8–10),
- generic programming (Chapters 12 and 13),
- functional programming (Chapter 14),
- concurrent programming (Chapter 18),
- and new in this edition—**genAI-enhanced programming** (Chapter 19)

Whether you’re a student getting a sense of the book you’ll be using, an instructor planning your course syllabus, or a professional software developer deciding which chapters to read as you prepare for a project, this Tour of the Book will help you make the best decisions.

### Part I: Introduction

Parts 1 and 2 (Chapters 1–7) provide a friendly, example-driven treatment of traditional introductory programming topics.

**Chapter 1, Intro, Test-Driving a Java Application, and Generative AI**, discusses technology trends and Internet concepts. It lays the groundwork for the Java programming discussions in Chapters 2–22 and the many case-study examples and exercises.

We discuss the technologies you’ll likely use as you develop software. We introduce the Java API—existing, reusable, top-quality, high-performance capabilities that help you avoid “reinventing the wheel.”

We discuss the Internet, the web, the cloud, the Internet of Things (IoT), the emerging metaverse, and exciting new developments in generative AI. We briefly mention many other topics of current interest including open-source software, virtualization, simulation, web services, multi-core hardware architecture, multithreading, natural language processing, data science, robust secure programming, cryptography, GitHub, StackOverflow, forums, blockchain, NFTs (nonfungible tokens), cryptocurrencies (like Bitcoin and Ethereum), artificial general intelligence (AGI) and more, laying the groundwork for modern Java application development.

This chapter’s test-drive demonstrates compiling and executing Java code with the OpenJDK version of the Java Development Kit. We consider just how big “big data” is and how quickly it’s getting even bigger. The chapter closes with an introduction to artificial intelligence (AI)—a key overlap between computer science and data science. **Includes 27 genAI prompt exercises.**

**Chapter 2, Intro to Java Programming**, presents Java fundamentals and illustrates key language features, including input, output, primitive data types, arithmetic operators and their precedence, and decision-making. As part of our objects-natural approach, Section 2.8’s

objects-natural case study demonstrates creating and using objects of the Java API's class **String**—without you having to know how to develop custom classes in general or how class **String** is implemented in particular). **Includes 12 genAI prompt exercises.**

## Part 2: Additional Programming Fundamentals

**Chapter 3, Control Statements: Part 1**, introduces Java's control statements. You'll use the **if** and **if...else** selection statements, the **while** iteration statement for counter-controlled and sentinel-controlled iteration, and the increment, decrement and assignment operators. Section 3.12's **objects-natural case study** demonstrates using the Java API's **BigInteger** class to create supersized integers much larger than computer hardware can natively represent. **Includes 15 genAI prompt exercises.**

**Chapter 4, Control Statements: Part 2**, presents additional Java control statements—for, do...while, **switch**, **break** and **continue**—and the logical operators. Section 4.11's **objects-natural case study** demonstrates using the Java API's **BigDecimal** class for precise monetary calculations. **Includes 13 genAI prompt exercises.**

**Chapter 5, Methods**, introduces custom methods. We discuss random-number generation and simulation techniques and use them in our first of several **random-number simulation case studies** throughout the book to **implement a popular casino dice game**. We introduce the Java API's random-number generation capabilities. In Chapters 7 and 11, we use the Java API's features for producing “nondeterministic” random numbers that cannot be predicted—desirable in security applications, such as cryptography. We also discuss how the method-call stack and activation records support the method call/return mechanism.

Section 5.12's **objects-natural case study** introduces features of the Java **Date/Time** API, which provides robust capabilities for conveniently managing date and time information, performing date and time calculations and presenting dates and times using locale-specific formatting that considers the user's spoken (or written) language, country and time zone (that is, the user's locale). We'll build a program that inputs a user's birth date and time (or approximations if they wish), then determines the day of the week the user was born and calculates how long the user has been alive. **Includes 16 genAI prompt exercises.**

**Chapter 6, Arrays and ArrayLists**, begins our early coverage of data structures. We present built-in arrays that store lists and tables of values. You'll define and initialize arrays and access their elements. We discuss passing arrays to methods, sorting and searching arrays and manipulating multidimensional arrays. We introduce class **Arrays**, which contains methods for performing common array manipulations. Section 6.20's **objects-natural case study** discusses creating and manipulating objects of the **Java Collection Framework's ArrayList<E>** collection class. The Java API provides many predefined collections (that is, data structures) that store groups of related objects in memory. These classes offer efficient, proven methods that organize, store and retrieve data portably without requiring knowledge of how the data is being stored. This reduces program development time and helps create more robust applications. **Includes 28 genAI prompt exercises.**

**Chapter 7, Strings, NLP and Regex: Generative AI Foundations**, presents many of the **String**, **Character** and **StringBuilder** classes' features. We present an intro to natural language processing (NLP), which is at the root of powerful generative AIs like OpenAI's



ChatGPT, Google’s Gemini, Anthropic’s Claude, Perplexity and many others. This chapter presents two **objects-natural case studies**:

- Section 7.8 introduces **regular-expression pattern matching and text replacement** using built-in features of class `String` and features from the `java.util.regex` package. After demonstrating regular expression fundamentals, we use them to perform simple data-wrangling steps to prepare text for training NLP and generative AI models.
- Section 7.9’s title—`pMa5tfEKwk59dTvC04Ft1IFQz9mEXnkfYXZwxk4ujGE=`—looks like gibberish. This is not a mistake! This case study continues our emphasis on security and privacy by introducing **cryptography**, which is critically important in today’s connected world. Every day, cryptography is used behind the scenes to ensure your Internet-based communications and stored data are private and secure. This case study introduces **private-key cryptography** with the AES (Advanced Encryption Standard) algorithm. In Chapter 11, our final objects-natural case study presents **public-key cryptography with the enormously popular RSA algorithm**.

Includes 22 genAI prompt exercises.

### Part 3: Object-Oriented Programming

**Chapter 8, Real-World Modeling with Custom Classes**, begins our contemporary object-oriented programming treatment. Java is extensible—each class you create becomes a new type you can use to create objects. In Chapters 8 and 9, you’ll learn Java’s features for crafting valuable classes and manipulating objects of those classes, starting with a case study on creating and using a simple bank account class. In the context of a **Random-Number Simulation case study**, you’ll use collections of `Strings`, random-number generation and simulation techniques to implement a **text-based, card-shuffling-and-dealing program**. You can rework this case study using JavaFX (Chapters 15–17) and attractive free Wikimedia Commons card images from

```
https://commons.wikimedia.org/wiki/
Category:SVG_English_pattern_playing_cards
```

Using a `Time` class case study and several additional classes, we continue with a deeper look at building classes, controlling access to class members and creating constructors to initialize class objects. We validate data and throw exceptions to indicate that problems have occurred. We design classes with *set* and *get* methods for changing and retrieving an object’s instance variable values and discuss **composition** in which a class has references to objects of other classes as members. We discuss `enum`, `static` and `final` in more depth. We also discuss how Java reclaims unused objects (a process called garbage collection) and show a special relationship among classes in the same package. We introduce **record classes** for creating immutable objects that conveniently store related data items then use them to introduce pattern matching with `switch` expressions. **Includes 33 genAI prompt exercises.**

**Chapter 9, Real-World Modeling with Inheritance, Polymorphism & Interfaces**, focuses on the relationships among classes in an inheritance hierarchy and the powerful runtime polymorphic processing capabilities (for conveniently “programming in the general”) that these relationships enable. In this chapter’s polymorphism case study, you’ll implement an `Employee` class hierarchy in an application that performs polymorphic payroll calculations.

We distinguish between the **inheritance *is-a* relationship** and **composition *has-a* relationship**. Inheritance tends to create tightly coupled classes, which can make it difficult to maintain a system. We introduce interfaces, which are particularly useful for assigning common functionality to possibly unrelated classes, enabling objects of these classes to be processed polymorphically. We use a `Payable` interface implemented in disparate `Employee` and `Invoice` classes to demonstrate that objects of classes implementing the same interface can respond polymorphically to that interface's method calls.

We also explain current idioms, such as “**programming to an interface, not an implementation**” and “**preferring composition to inheritance**.” We reimplement the `Employee` hierarchy using **composition and dependency-injection techniques** to create loosely coupled classes, which makes systems easier to maintain. We also introduce `sealed` classes and interfaces, which enable a class's or interface's designer to control which classes can extend a superclass or implement an interface. **Includes 31 genAI prompt exercises.**

**Chapter 10, Exception Handling: A Deeper Look**, continues our exception-handling discussion that began in Chapter 6. Exception handling is important for building **business-critical** applications in which a failure could disrupt company operations and cause financial losses, like

- E-commerce platforms (e.g., online stores), such as Amazon, eBay, Etsy and Shopify
- Customer Relationship Management (CRM) software, such as Salesforce, HubSpot and Zoho

and for building **mission-critical** applications in which failure could lead to significant financial loss, injury or even loss of life, like

- air traffic control systems,
- hospital life-support systems,
- emergency response systems used to dispatch police, firefighters and emergency medical technicians (EMTs), and
- military systems.

To use a Java component, you need to know not only how that component behaves when “things go well” but also what exceptions that component “throws” when “things go poorly.”

We discuss when to use exceptions and demonstrate Java's `try` statement for catching and handling exceptions. We introduce Java's `Throwable` hierarchy and checked vs. unchecked exceptions, chained exceptions, creating custom exceptions, preconditions, postconditions, assertions and catching multiple exceptions with one catch handler. We also discuss the **try-with-resources statement** for preventing “resource leaks” by automatically releasing resources like files, network connections and database connections when a try-with-resources statement's `try` block terminates. This version of the `try` statement is used extensively in subsequent chapters to ensure resources like files and database connections are promptly returned to the system when no longer needed. **Includes 30 genAI prompt exercises.**

**Chapter 11, Files, I/O Streams, JSON Serialization & CSV Files**, demonstrates using files for data persistence and is a significant update in this edition. We begin with Java's archi-

ture for handling files programmatically and discuss the differences between text and binary files. We demonstrate using NIO classes and interfaces to retrieve information about files and directories. Then, we create and manipulate sequential text files using `Formatter` objects to write to and `Scanner` objects to read from them.

Next, we present several case studies that demonstrate transforming objects to and from popular data formats. The first introduces **JavaScript Object Notation (JSON)**—a human-and-computer-readable text format. We use the popular **Jackson open-source library** to write objects’ JSON representations to a file (a process known as serialization), then read the JSON and recreate (deserialize) the objects. The second JSON case study uses Java API networking features from the **`java.net.http` package** to invoke an `OpenWeatherMap.org` web service that returns a city’s weather report in JSON format then uses Jackson to process the response and display the weather report.

Next, we introduce the **CSV (comma-separated values) file format** and how to write and read CSV files using the Jackson open-source library. CSV is popular for datasets used in **big data**, **data analytics**, and **data science**, as well as **artificial intelligence applications** like **natural language processing**, **machine learning**, and **deep learning**—all key technologies that enable genAI. We use the Jackson library to read the **Titanic disaster dataset**, which lists information about the passengers and whether they survived when the ship struck an iceberg and sank during its maiden voyage in 1912. Then, we view some of the data and use it to introduce **basic data analytics**—a core **data science** technology.

In our final **objects-natural case study**, we continue emphasizing security by enhancing our private-key cryptography example from Section 7.9, using **RSA public-key cryptography** to encrypt and decrypt the AES secret key. This enables secure transmission of the private key to an encrypted message’s recipient. Such techniques are critical for securing data and protecting user privacy. **Includes 38 genAI prompt exercises.**

#### Part 4: Data Structures, Generic Collections, Lambdas and Streams

**Chapter 12, Generic Collections**, presents our broader and deeper treatment of the Java generic collections framework that began with the **generic `ArrayList` collection** in Chapter 6. The Java collections framework contains many other **pre-built generic data structures**. We discuss the interfaces that declare the capabilities of each collection type, various classes that implement these interfaces, methods that process collection objects, and iterators that traverse collections. You’ll see that **the Java API provides commonly used data structures, so you do not need to create your own**—the vast majority of your data structure needs can be fulfilled by reusing these Java API capabilities.

We introduce convenience factory methods that help you create small immutable collections. After reading **Chapter 14, Functional Programming with Lambdas & Streams**, you’ll be able to reimplement many of this chapter’s examples more concisely and elegantly, and in a way that makes them easier to parallelize to improve performance on multi-core systems. In Chapter 18, **Concurrency: Platform Threads to Virtual Threads**, you’ll learn how to improve multi-core performance using Java’s **concurrent collections** and **parallel stream operations**. **Includes 30 genAI prompt exercises.**

**Chapter 13, Generic Classes and Methods: A Deeper Look**, shows how to write custom generic classes and methods similar to those presented in **Chapter 12, Generic Collections**. We discuss how generics enable the compiler to detect type mismatches at compile time—

known as **compile-time type safety**. We implement a generic method and a generic class. **Includes 24 genAI prompt exercises.**

**Chapter 14, Functional Programming with Lambdas & Streams**, introduces functional programming. We use lambdas and streams to write certain kinds of programs faster, simpler, more concisely and hopefully with fewer bugs than previous techniques. We architected the chapter as a series of easy-to-include-or-omit sections keyed to the book’s earlier sections and chapters. We integrate lambdas and streams into several key examples after Chapter 14 because their capabilities are so convenient and compelling. Many of this chapter’s sections are written so they can be covered earlier in the book. We suggest students begin with Sections 14.1–14.7 after Chapter 6 and professionals begin with Sections 14.1–14.5 after Chapter 4. We demonstrate improved ways to implement tasks you programmed in earlier chapters. After reading Chapter 14, you’ll be able to cleverly reimplement many examples throughout the book. In **Chapter 18, Concurrency: Platform Threads to Virtual Threads**, you’ll parallelize (i.e., perform multiple operations simultaneously) a stream operation so it can take advantage of multi-core architectures to enhance performance—a key goal of lambdas and streams. In Chapter 18, you’ll also learn that it’s hard to create parallel tasks that operate correctly if those tasks modify a program’s state (that is, its variables’ values). So, the techniques you’ll learn in Chapter 14 focus on immutability—not modifying the data source being processed or any other program state. **Includes 37 genAI prompt exercises.**

(Optional) **Chapter 22, Recursion and Big O**, can be covered here.

## Part 5: JavaFX Graphical User Interfaces, Graphics and Multimedia

**Chapter 15, JavaFX Graphical User Interfaces: Part 1**, begins our multiple-chapter feature on JavaFX graphical user interfaces (GUIs), graphics, multimedia and game-playing. Chapter 15 focuses on developing GUIs, which provide a user-friendly mechanism for interacting with applications. A GUI gives an application a distinctive “look-and-feel.” Providing apps with consistent, intuitive user-interface components gives users a sense of familiarity with a new application so they can learn it faster and use it more productively. This chapter introduces JavaFX GUI basics, including layout, controls and event handlers, which we’ll implement using lambdas. You’ll use the free Scene Builder tool to quickly create two GUIs with simple drag-and-drop design techniques. **Includes 13 genAI prompt exercises.**

**Chapter 16, JavaFX GUI: Part 2**, continues our JavaFX presentation with additional GUI layouts and controls. We show how to handle mouse and `RadioButton` events, set up event handlers that respond to property changes on controls (such as the value of a `Slider`), display shapes, bind variables to controls so the controls update automatically when the variables’ data changes, and customize a GUI component’s appearance. You’ll also use `FileChooser` and `DirectoryChooser` dialogs to conveniently select a file or directory. Chapter 18 shows how to perform long-running tasks in a manner that allows a JavaFX GUI to remain responsive. **Includes 21 genAI prompt exercises.**

**Chapter 17, JavaFX Graphics and Multimedia**, presents key JavaFX graphics and multimedia capabilities, including:

- Cascading Style Sheets (CSS) for customizing JavaFX nodes' appearance and text fonts—this is based on the same technology used for decades in web development to precisely and elegantly control webpage styling.
- Two-dimensional shapes (lines, rectangles, circles, ellipses, arcs, polylines, polygons and custom paths).
- Transforming JavaFX nodes in various ways, such as translating (moving), rotating and scaling (resizing).
- Displaying videos and controlling their playback (e.g., play and pause).
- Transition and Timeline animations that dynamically change nodes' property values (such as rotation, size and location) over time.
- Frame-by-frame animations with an AnimationTimer.
- CSS transitions for simplifying certain animation effects.
- Canvas for drawing with the mouse.
- Three-dimensional shapes (boxes, cylinders and spheres).

We also introduce **FXGL for JavaFX game development** and are planning a deeper treatment for our blog. **Includes 31 genAI prompt exercises.**

## Part 6: Advanced Topics

**Chapter 18, Concurrency: Platform Threads to Virtual Threads**, focuses on improving application performance and responsiveness with concurrency and multi-core programming. Chapter 18 is one of the most important chapters in the book, presenting Java's features for building applications that create and manage multiple tasks. We updated the chapter to the latest Java technologies and idioms, especially those of the game-changing Project Loom.

This chapter presents several **multithreading and multi-core systems performance case studies**. In the **Profiling Sequential and Parallel Sorting Algorithms** example, we show how to use prepackaged parallel algorithms to create multithreaded programs that will run faster (often much faster) on today's multi-core computer architectures. For example, we sort 100 million values using a sequential sort then a parallel sort. We use timing operations from Java's Date/Time API to profile the performance improvement we get on today's popular multi-core systems as we employ more cores. We show that the parallel sort runs 700% faster than the sequential sort on our computer with a 12-CPU-core Apple M2 Max processor. We include a **parallel vs. sequential stream processing** example, again using the Date/Time API to show performance improvements. The **CompletableFuture** example demonstrates sequential and parallel execution of long-running calculations.

In the **Producer/Consumer Relationship with ArrayBlockingQueue** case study, we discuss the classic producer–consumer relationship and demonstrate how to implement it using the predefined ArrayBlockingQueue class. We emphasize that **concurrent programming is difficult to get right, so you should prefer the easier-to-use, less error-prone, higher-level concurrency features**.

We've added coverage of **Project Loom**, which enhances Java's threading model with simpler, lightweight **virtual threads** (finalized in Java 21). These require significantly less memory and processor overhead than Java's traditional platform threads, enabling applications to launch enormous numbers of concurrent virtual threads and helping meet the

demands of today’s massively parallel applications. Project Loom reduces the complexity of writing concurrent programs, making concurrency easier to use. We also introduce emerging Project Loom features—structured concurrency and scoped values.<sup>31</sup> **Structured concurrency** enables programmers to treat multiple threads as a single unit of work (a scope), making it easier to manage those threads, handle errors and reclaim resources no longer needed. **Scoped values** simplify data sharing among virtual threads in the same scope, reducing the need for complex thread-synchronization techniques. **Includes 51 genAI prompt exercises.**

**Chapter 19, Building API-Based Java Generative AI Applications**, includes many cool, fully coded API-based Java generative AI applications—see the Integrated Generative AI section’s “Our Chapter 19 API-Based Complete, Live-Code Java GenAI Apps” subsection earlier in this Preface for the complete list. In the preceding chapters, we focused on manually prompting genAIs via their web interfaces and asking you to study, compare and use their results. We did not show the results of our suggested genAI prompts—you observed those in action.

When genAIs were first introduced, they typically received a text prompt and generated a specific type of output, such as text or an image. Many genAIs are now multimodal, enabling them to understand text, images, audio and even video inputs and use them to generate outputs combining text, images, audio and video. For example, you can prompt genAIs with a speech audio file and instructions asking them to transcribe the speech to text, display the transcription, analyze its sentiment (positive, negative or neutral), translate it to another spoken language and synthesize speech in that other language—and even create images and videos from the generated text! These capabilities were out of reach to most developers just a few years ago.

This chapter uses the **Simple-OpenAI open-source library** to enable Java programmers to conveniently interact with OpenAI’s APIs to build powerful multimodal Java generative AI applications. You’ll observe that prompt engineering is also an essential aspect of programming with genAI APIs. In the chapter examples and exercises, you’ll work with genAIs to build text, speech, code, image and video apps. **Includes 94 genAI prompt and API-based coding exercises.**

**Chapter 20, Accessing Databases with JDBC and SQLite**, discusses database management systems (DBMSs) for conveniently storing and organizing persistent data. After an overview of database concepts, you’ll use the popular open-source **SQLite** database management system’s command-line tools to learn **Structured Query Language (SQL)** fundamentals. You’ll execute SQL queries live to perform basic CRUD (create, read, update, delete) operations on a SQLite database and immediately see their results. Next, we use **Java’s JDBC APIs** to interact programmatically with SQLite databases. JDBC is portable—so the same code we show can manipulate databases in many popular open-source and proprietary database management systems. We demonstrate connecting to databases, executing queries to update databases and retrieve data from them, and displaying query results in JavaFX **TableViews**. The chapter features a JavaFX database-driven address book application demonstrating **prepared statements** for reusable, more secure, parameterized queries. You’ll enhance this app with update and delete options in the exer-

---

31. At the time of this writing, structured concurrency and scoped values were preview features undergoing rapid change. Once they stabilize, we’ll present revised live-code examples on our blog at <https://deitel.com/blog>.



cises. Just like genAIs can write Java code for you, they can also write SQL code, as you’ll do in this chapter’s genAI prompt exercises. **Includes 19 genAI prompt exercises.**

**Chapter 21, Java Platform Module System (JPMS)**, introduces modules, which provide a higher level of aggregation above packages and enable precise organization of your code. By default, modules encapsulate implementation details while allowing you to explicitly control what is exposed to client code. Modules help developers be more productive as they build, maintain and evolve large software systems. We discuss JPMS’s goals then demonstrate creating custom packages and modules, declaring module dependencies, specifying which packages a module explicitly makes available to other modules, and defining the services a module offers or consumes. We also discuss how to control reflective access between modules. Reflection allows programs to inspect and manipulate classes and their members at runtime. For example, JavaFX leverages reflection to bind a program’s GUI controls to the Java code that handles user interactions with the controls. **Includes 31 genAI prompt exercises.**

**Chapter 22, Recursion and Big O**, discusses some key classic computer science topics. First, we use factorial and Fibonacci calculations to introduce recursive methods that call themselves directly or indirectly. We develop a JavaFX application that creates a custom “feather fractal”—a geometric figure that can be generated from a pattern repeated recursively. We present a friendly introduction to **Big O notation**, which indicates how hard an algorithm may have to work to solve a problem based on the number of items it must process. **Includes 11 genAI prompt exercises.** [Chapter 22 can be covered after Chapter 14.]

## Appendices

**Appendix A, Introduction to JShell for Interactive Java**, provides optional flexible coverage of JShell, Java’s friendly, command-line REPL (read-evaluate-print-loop) environment that enables you to quickly explore, discover and experiment with Java’s language features and libraries—it’s like having Python’s interactivity in Java. Many of this appendix’s sections can be covered with the book’s early chapters for instructors who’d like to incorporate this capability in their courses from the start.

JShell replaces the edit-compile-execute cycle with its **read-evaluate-print-loop (REPL)**. Rather than complete programs, you write JShell commands and Java code snippets. When you enter a snippet, JShell immediately

- **reads** it,
- **evaluates** it,
- **prints** (displays) messages that help you see the effects of your code, and
- **loops** to perform this process again for the next snippet.

The appendix is **example-intensive**. This will help you master the basics of JShell quickly. You’ll see how JShell and its **instant feedback** keep your attention and speed the learning and software-development processes.

You’ll find JShell easy and fun to use. It will help you learn Java features faster and more deeply and will help you verify that these features work how they’re supposed to. If you are an instructor, you’ll appreciate how JShell encourages your students to dig in and how it leverages the learning process. If you are a professional, you’ll appreciate how JShell helps you rapidly prototype key code segments and discover and experiment with new APIs.

**Appendix B, Formatted Output**, presents formatting features for output and for Strings in memory. The appendix summarizes the formatting features discussed throughout the book and introduces additional capabilities.

**Appendix C, Number Systems**, discusses the binary (base 2), octal (base 8), decimal (base 10) and hexadecimal (base 16) number systems programmers use.

## Key Developer Sites—GitHub and StackOverflow

*“The best way to prepare [to be a programmer] is to write programs, and to study great programs that other people have written. In my case, I went to the garbage cans at the Computer Science Center and fished out listings of their operating systems.”*<sup>32</sup>—William Gates (Microsoft founder)

You’re probably already working with such popular websites as StackOverflow and GitHub. If not, check them out:

- **StackOverflow** is one of the most popular programming question-and-answer sites. Many problems you might encounter have already been discussed here. It’s a great place to get your code-oriented questions answered. Many of our Google searches for various, often complex, issues throughout our writing effort returned StackOverflow posts as their first results.
- **GitHub** (now owned by Microsoft<sup>33</sup>) is an excellent venue for finding free, open-source code to explore and incorporate into your projects—and for you to contribute your code to the open-source community if you’d like. “Open source is software with source code that anyone can inspect, modify, and enhance.”<sup>34</sup> We encourage you to try lots of demos and view free, open-source code examples (available on sites such as GitHub) for inspiration. Over one hundred million developers worldwide use GitHub.<sup>35</sup> The site hosts over 518 million code repositories in hundreds of programming languages—developers made 5.2 billion contributions to those repositories in 2024.<sup>36</sup> GitHub includes version-control tools that help developers manage public open-source projects and private projects. **There is a massive Java open-source community on GitHub where developers contribute to over 250 thousand Java code repositories.**<sup>37</sup> We encourage you to study and execute lots of developers’ open-source Java code. It’s a great way to learn and is a natural extension of our live-code and objects-natural teaching approaches enhanced with generative AI.<sup>38</sup>

32. William Gates, quoted in *Programmers at Work: Interviews With 19 Programmers Who Shaped the Computer Industry* by Susan Lammers. Microsoft Press, 1986, p. 83.

33. Microsoft, “Microsoft to Acquire GitHub for \$7.5 Billion,” Microsoft News, June 4, 2018. Accessed January 29, 2025. <https://news.microsoft.com/2018/06/04/microsoft-to-acquire-github-for-7-5-billion/>.

34. “What is open source?” Accessed January 18, 2025. <https://opensource.com/resources/what-open-source>.

35. “Let’s build from here: The complete developer platform to build, scale, and deliver secure software.” Accessed January 18, 2025. <https://github.com/about>.

36. “Octoverse 2024: The state of open source software.” Accessed January 18, 2025. <https://octoverse.github.com/>.

37. “Java.” Accessed January 18, 2025. <https://github.com/topics/java>.

38. You’ll need to become familiar with the variety of open-source licenses for software on GitHub.

## Deitel.com and the Deitel Blog

On the book's webpage at <https://deitel.com/javafp5>, we provide:

- The book's reviewer testimonials.
- A link to the book's GitHub repository for downloading the example code.
- Book updates.

For more information about downloading the code examples and setting up your Java development environment, see the **Before You Begin** section that follows this Preface. When we create Java-related blog posts at

<https://deitel.com/blog>

we'll also link each to the book's webpage at <https://deitel.com/javafp5>.

## Software Used in *Java for Programmers with Generative AI, 5/e*

Most of the software you'll need for this book is free for download, though the code examples in Chapter 19, Building API-Based Java Generative AI Applications, use paid web services. See the **Before You Begin** section following this Preface for download links. We wrote the Java code examples using the free OpenJDK on macOS and Windows. We developed a **utility class** used by Chapter 19's examples in the free **JetBrains IntelliJ IDEA Community Edition** integrated development environment (IDE).

## Java Documentation Links

The book includes abundant citations to research papers, articles, white papers, videos, blog posts, online documentation and more that we studied while writing the manuscript. You may want to access some of these resources for additional information and insights. The latest Java versions' documentation is available at:

<https://docs.oracle.com/en/java/javase/index.html>

We reference the API documentation frequently so you can get more details about the Java API capabilities we use:

<https://docs.oracle.com/en/java/javase/23/docs/api/index.html>

As new Java versions are released every six months, you can access their API documentation by replacing 23 in the preceding link with the new version number (24, 25, etc.).

We also frequently cite the latest Java SE (Standard Edition) Specification, which is available in both HTML and PDF forms at:

<https://docs.oracle.com/javase/specs/>

You can keep track of future changes at:

<https://openjdk.org/projects/jdk/>

## Getting Your Questions Answered

Popular Java and general programming online forums include

- <https://stackoverflow.com>
- <https://www.reddit.com/r/java/>
- <https://dev.to/t/java>

Also, vendors often provide forums for their tools and libraries. Many libraries are managed and maintained at <https://github.com>. Some library maintainers offer support through the Issues tab on a given library's GitHub page.

### GenAIs

You can conveniently ask genAIs Java questions and get immediate responses. We recommend that you ask multiple genAIs and compare the results. Keep in mind that genAIs sometimes make mistakes and even hallucinate, so you should verify the results with additional online searches. Even though we experienced these problems many times as we developed this manuscript, working with genAIs has leveraged our ability to create effective and engaging pedagogy. We always look forward to their responses to our prompts.

### Communicating with the Authors

As you read the book, if you have questions, we're easy to reach at

`deitel@deitel.com`

We'll respond promptly.

## Join the Deitel & Associates, Inc. Social Media Communities

You can keep up to date with Deitel on the following social media platforms:

- **LinkedIn** (<https://linkedin.com/company/deitel-&-associates>)
- **Facebook** (<https://facebook.com/deite1fan>)
- **Instagram** (<https://instagram.com/Deite1Fan>)
- **X** (@deitel or <https://x.com/deitel>)
- **YouTube** (<https://youtube.com/Deite1TV>)
- **Mastodon** (<https://mastodon.social/@deitel>)

## College Textbook

If you're an instructor teaching college courses, you might want to consider using *Java How to Program: An Objects-Natural Approach, 12/e*, which is the college-textbook version of *Java for Programmers with Generative AI, 5/e*, and is available in various digital formats. For more information, see the textbook's Preface at

<https://deitel.com/jhttp12>

and contact your Pearson Education representative:

<https://pearson.com/en-us/replocator>

## Acknowledgments

Thanks to Barbara Deitel for the long hours devoted to technical research on this project. We're fortunate to have worked with Pearson's dedicated team of publishing professionals. We appreciate the guidance, wisdom and energy of Laura Lewin, Executive Editor of Pearson, IT & Professional group. Laura and her team handle all of our professional books in print and digital formats. Jackleen Sougrakpam and Erin Sullivan recruited the book's professional and academic reviewers, respectively, and managed the review process. Julie Nahil managed the production of the book's suite of print and digital versions. Ashley Santora, Kaitlyn Coddington and Jenell Forschler helped us determine which genAIs in general and which genAI outputs in particular were acceptable to our publisher, Pearson, from a permissions perspective. Chuti Prasertsith located the cover art from Pearson's approved sources and designed several alternative covers. Jessica Deitel's design sense helped us select the final cover from those alternatives.

## Reviewers

We wish to acknowledge the efforts of our reviewers—Oracle Java team members, Oracle Java Champions, other industry professionals and distinguished academics. They scrutinized the text and the programs and provided countless suggestions for improving the presentation. For this fifth edition they helped us get our new generative AI and objects-natural pedagogic approaches right. Any remaining faults in the book are our own.

We appreciate the guidance of JavaFX experts Johan Vos and Carl Dea (and for previous editions, Jim Weaver, Jonathan Giles and Simon Ritter) on the JavaFX GUI, graphics and multimedia Chapters 15–17.

### *Fifth Edition Professional and Academic Reviewers:*

- Jeanne Boyarsky (CodeRanch, Java Champion, co-author of the *OCP Oracle Certified Professional Java SE 21 Developer Study Guide*).
- Carl Dea (Lead Software Developer at a global business consulting and services company, co-author *JavaFX 9 by Example*).
- Trisha Gee (Java Champion).
- Simon Roberts (President, Dancing Cloud Services, LLC).
- José Antonio González Seco (Parliament of Andalusia, Spain).
- Ron Veen (Java Developer, Special Agent at Team Rockstars IT) and David Vlijmincx (Senior Software Developer, JPoint), co-authors of *Virtual Threads, Structured Concurrency, and Scoped Values: Explore Java's New Threading Model*, and *Cloud-Native Development and Migration to Jakarta EE*.
- Johan Vos (Co-founder and CTO, Cloud Products at Gluon, Java Champion, co-author *The Definitive Guide to Modern Java Clients with JavaFX: Cross-Platform Mobile and Cloud Development Updated for JavaFX 21 and 23*).
- Brian Canada (Professor of Computational Science, University of South Carolina Beaufort).
- Bob Myers (Computer Science Department, Florida State University).
- Emily Navarro (Continuing Lecturer, Department of Informatics, University of California, Irvine).

### A Special Thank You to Brian Goetz

Brian Goetz, Oracle’s Java Language Architect and co-author of *Java Concurrency in Practice*, has reviewed several editions of our Java content over the years. His insights and constructive comments greatly enhanced our understanding of the subtleties of Java concurrency and Java object-oriented programming. He patiently guided us to “get it right.” It was a privilege to be his student.

### A Special Thank You to Robert Field

Robert Field, Oracle’s JShell Architect reviewed the JShell appendix while the tool was under development, responding to our numerous emails in which we asked questions, reported bugs we encountered and suggested improvements. Having our content scrutinized by the person responsible for JShell and seeing some of our suggestions incorporated into the product was a special professional experience for us.

### Google Search

Thanks to Google, whose search engine answers our constant stream of queries, each in a fraction of a second, at any time—and at no charge. It’s one of the best productivity enhancement tools we’ve added to our research process in the last 20 years.

### Grammarly

We use the paid version of Grammarly on all our manuscripts. They describe their product as an “AI-powered writing partner.”<sup>39</sup> Grammarly also provides powerful free tools that can be integrated into several popular web browsers, Microsoft Office 365 and Google Docs.

### GenAIs

Throughout the book-development process, we used the paid versions of **OpenAI’s ChatGPT**, **Google’s Gemini** and **Anthropic’s Claude**, and the free version of **Perplexity** to research new Java topics, verify facts, check our code for the latest Java idioms and proper commenting, suggest enhancements, ensure thorough coverage and more. Much of what we learned as we used them is reflected in the approximately 600 Generative AI prompting exercises and API-based coding exercises integrated throughout the book. The major API-based coding case studies are in Chapter 19.

### Oracle, Gluon and the OpenJFX Community

Thanks to Oracle Corporation, Gluon and the OpenJFX Community, which collaborate on JavaFX’s evolution, releasing a new JavaFX version with each new Java SDK version. Also, thanks to Gluon for their friendly and convenient drag-and-drop Scene Builder GUI development tool.

Welcome to the exciting world of Java programming enhanced with generative AI. We’ve enjoyed writing many editions of our professional and academic Java content over the last 30 years. We hope you have an informative, challenging and entertaining learning experience using *Java for Programmers with Generative AI, 5/e*, and enjoy this look at modern Java software development.

---

39. “Grammarly.” Accessed January 18, 2025. <https://grammarly.com>.



As you read the book and work through the code examples and generative AI exercises, we'd appreciate your comments, criticisms, corrections and suggestions for improvement. Please send all correspondence, including questions, to

`deitel@deitel.com`

We'll respond promptly. Our best wishes for your success,

*Paul Deitel*

*Harvey Deitel*

## About the Authors



**Paul J. Deitel**, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of MIT with over 44 years of experience in computing. He holds the Java Certified Programmer and Java Certified Developer designations and is an Oracle Java Champion. He is one of the world's

most experienced programming-languages trainers, having taught hundreds of professional courses to software developers since 1992. His video and live-training courses on platforms like O'Reilly Online Learning have garnered millions of contact minutes, with his Java Fundamentals, Python Fundamentals and C# Fundamentals video courses each ranking #1 for several years among thousands of video products. He has delivered programming courses to academic, industry, government and military clients of Deitel & Associates, Inc. internationally, including UCLA, SLB (formerly Schlumberger), IBM, Siemens, Sun Microsystems (now Oracle), Dell, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, Cisco, Puma, iRobot and many more. He and his co-author, Dr. Harvey M. Deitel, are among the world's best-selling programming-language textbook/professional book/video/interactive multimedia authors.

**Dr. Harvey M. Deitel**, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., is recognized as a thought leader and innovator in programming languages education. He has over 64 years of experience in computing. Dr. Deitel earned B.S. and M.S. degrees in Electrical Engineering from MIT and a Ph.D. in Mathematics from Boston University—he studied computing in each of these programs before they spun off Computer Science programs. He has extensive college teaching experience, including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates in 1991 with his son, Paul. The Deitels' publications have earned international recognition, with more than 100 translations published in Japanese, German, Russian, Spanish, French, Polish, Italian, Simplified Chinese, Traditional Chinese, Korean, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of programming courses to academic, corporate, government and military clients.

## About Deitel® & Associates, Inc.

Deitel & Associates, Inc., founded by Paul Deitel and Harvey Deitel, is an internationally recognized authoring and corporate-training organization specializing in computer programming languages, object technology, mobile app development, Internet and web software technology and generative AI. The company's training clients include some of the world's largest companies, government agencies, branches of the military, and academic institutions. The company offers instructor-led training courses delivered live at client sites worldwide and virtually worldwide for Pearson.

Through its five-decade publishing partnership with Pearson, Deitel & Associates, Inc. publishes leading-edge computer programming college textbooks and professional books in print and digital formats, video courses, O'Reilly Online Learning live training courses and Pearson+ eText and Revel interactive multimedia college courses.

To contact Deitel & Associates, Inc. and the authors or to request a proposal for virtual or on-site, instructor-led training worldwide, write to

`deitel@deitel.com`

To learn more about Deitel corporate training, visit

<https://deitel.com/training>

Individuals wishing to purchase Deitel print and digital publications online can do so at

<https://amazon.com>

<https://barnesandnoble.com/>

For convenient access to updates and corrections as they become available, register your copy of *Java for Programmers with Generative AI, 5/e*, on Pearson's InformIT site. To start the registration process, go to

<https://informit.com/register>

and log in or create an account. Enter the product ISBN (9780137574735) and click Submit.

Deitel e-books are available in various formats from

<https://amazon.com/>

<https://vitalsource.com/>

<https://barnesandnoble.com/>

<https://redshelf.com/>

<https://informit.com/>

To register for a free 10-day trial to O'Reilly Online Learning, visit

<https://learning.oreilly.com/register>