LEARNING MADE EASY

9th Edition

Java®

for dummies®

A Wiley Brand

Create your own Java programs

Diagnose and solve coding errors

Organize data with objects and classes

Barry Burd, PhD

Author of *Beginning Programming with Java For Dummies* and *Flutter For Dummies.*

Java®

for dummies®

A Wiley Brand

Java®

9th Edition

by Barry Burd, PhD

for dummies®
A Wiley Brand

Java® For Dummies®, 9th Edition

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Published simultaneously in Canada.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit https://hub.wiley.com/community/support/dummies.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2024951544

ISBN: 978-1-394-28924-0 (pbk); 978-1-394-28925-7 (ebk); 978-1-394-28926-4 (ebk)

Contents at a Glance

Table of Contents

Introduction

What's all the fuss about Java? To help answer that question, I offer a few facts:

- Nine million of the world's developers work on applications in Java, and Java runs on approximately 7 billion devices.*
- Ninety percent of all Fortune 500 companies use Java.**
- According to the TIOBE Programming Community Index, Java is the world's third most popular programming language.***
- In 2021, Glassdoor, Inc., ranked jobs based on earnings potential, job satisfaction, and number of available job openings. Among the company's "50 Best Jobs in America for 2021," a career as a Java developer ranked number one.****

Sounds good. Right?

Please, read on.

About This Book

This book isn't the usual dry techie guide. It's written for normal human beings — people with little or no programming experience. In this book, I divide Java into manageable chunks. Each chunk is (more or less) a chapter on its own. I explain concepts in plain language using complete code examples that you can download and run. I keep each code example focused on a few key concepts. I resist the urge to use fancy tricks that impress professional programmers. I expand on concepts that may be difficult for newcomers. I add diagrams to help you visualize important ideas. I provide exercises with each chapter along with solutions to the exercises on the book's website.

---

* `www.geeksforgeeks.org/reasons-to-learn-java`

** `https://softjourn.com/insights/is-java-still-used`

*** `www.tiobe.com/tiobe-index`

**** `www.glassdoor.com/blog/best-jobs-in-america-for-2021`

Finally, and most importantly — and without question the most significant of all this book's features — I throw in some jokes. I've written some good jokes and lots of bad jokes. (I should say "lots and lots" of bad jokes.) I've hidden Easter eggs in the text. I've added anecdotes about all kinds of topics. Some of the anecdotes are true, and many of them are . . . well, you figure it out.

Foolish Assumptions

In this book, I make a few assumptions about you, the reader. If one of these assumptions is incorrect, you're probably okay. If all these assumptions are incorrect, please buy the book anyway:

- **I assume that you have access to a computer.** Here's the good news: You can run most of the code in this book on almost any computer. The only computers you can't use to run this code are ancient boxes that are more than ten years old (give or take a few years).
- **I assume that you can navigate your computer's common menus and dialog boxes.** You don't have to be a Windows, Linux, or Macintosh power user, but you should be able to start a program, find a file, put a file into a certain directory — that sort of thing. Most of the time, when you follow instructions in this book, you're typing code on the keyboard, not pointing-and-clicking the mouse.
- **I assume that you can think logically.** That's all there is to programming in Java — thinking logically. If you can think logically, you have it made. If you don't believe that you can think logically, read on. You may be pleasantly surprised.
- **I make few assumptions about your computer programming experience (or your lack of such experience).** In writing this book, I've tried to do the impossible: Make the book interesting for experienced programmers yet accessible to people with little or no programming experience. So, I assume no particular programming background on your part. If you've never created a loop or indexed an array, that's okay.

  On the other hand, if you've done these things (maybe in Visual Basic, Python, or C++), you'll discover some interesting plot twists in Java. The developers of Java took the best ideas in object-oriented programming, streamlined them, reworked them, and reorganized them into a sleek, powerful way of thinking about problems. You'll find many new, thought-provoking features in Java. As you find out about these features, many of them will seem quite natural to you. One way or another, you'll feel good about using Java.

Icons Used in This Book

If you could watch me write this book, you'd see me sitting at my computer, talking to myself. I say each sentence in my head. Most of the sentences, I mutter several times. When I have an extra thought or a side comment that doesn't belong in the regular stream, I twist my head a little bit. That way, whoever's listening to me (usually, nobody) knows that I'm off on a momentary tangent.

Of course, in print, you can't see me twisting my head. I need some other way to set a side thought in a corner by itself. I do it with icons. When you see a Tip icon or a Remember icon, you know that I'm taking a quick detour.

Here's a list of icons that I use in this book:


TIP

A tip is an extra piece of information — a helpful tidbit that the other books may forget to tell you.


WARNING

Everyone makes mistakes. Heaven knows that I've made a few in my time. Anyway, when I think people are especially prone to make a mistake, I mark it with a Warning icon.


REMEMBER

Sometimes I want to hire a skywriting airplane crew. "Barry," says the white smoky cloud, "if you want to compare two numbers, use the double equal sign. Please don't forget to do this." Because I can't afford skywriting, I have to settle for a more modest option: I create a paragraph marked with the Remember icon.


CROSS REFERENCE

"If you don't remember what such-and-such means, see blah-blah-blah," or "For more information, read blahbity-blah-blah."


TRY IT OUT

Writing computer code is an activity, and the best way to learn an activity is to practice it. That's why I've created things for you to try in order to reinforce your knowledge. Many of these are confidence-builders, and some are more challenging. When you first start putting concepts into practice, you'll discover all kinds of issues, quandaries, and roadblocks that didn't occur to you when you started reading about the material. But that's a good thing. Keep at it! Don't become frustrated. Or, if you do become frustrated, visit this book's website (`https://javafordummies.allmycode.com`) for hints and solutions.

This icon calls attention to useful material that you can find online. Check it out!

Occasionally, I run across a technical tidbit. The tidbit may help you understand what the people behind the scenes (the people who developed Java) were thinking. You don't have to read it, but you may find it useful. You may also find the tidbit helpful if you plan to read other (geekier) books about Java.

Beyond the Book

In addition to what you're reading right now, this book comes with a free, access-anywhere Cheat Sheet containing code that you can copy and paste into your own Java program. To get this Cheat Sheet, simply go to `www.dummies.com` and type **Java For Dummies Cheat Sheet** in the Search box.

For your extra reading pleasure, I've also arranged for my publishers to make available for download two appendices and a bonus chapter: The details are as follows:

- **Appendix A: Setting Up Your Computer,** which gives you the skinny on how to prep your computer for your upcoming Java adventure
- **Appendix B: Who's Afraid of Java Documentation?,** a non-nonsense guide to Java's API documentation
- **Bonus Chapter 1: Pat Your Head While You Rub Your Belly**, a guide for those interested in multitasking with your Java code

All three can be found at `www.dummies.com/go/javafd9e`.

Where to Go from Here

If you've gotten this far, you're ready to start reading about Java application development. Think of me (the author) as your guide, your host, your personal assistant. I do everything I can to keep things interesting and, most importantly, to help you understand.

If you like what you read, send me a note. My email address, which I created just for comments and questions about this book, is `JavaForDummies@allmycode.com`. (In case you're wondering, I answer my own emails. I don't use bots or paid assistants.) And don't forget — for the latest updates, visit this book's website. The site's address is `https://javafordummies.allmycode.com`.

1 Getting Started with Java

IN THIS PART . . .

Install the software you need for developing Java programs.

Find out how Java fits into today's technology scene.

Run your first complete Java program.

IN THIS CHAPTER

- What Java is
- Where Java came from
- Why Java is so cool
- Three ways to write computer code

Chapter 1
All about Java

Say what you want about computers. As far as I'm concerned, computers are good for just two simple reasons:

- **When computers do work, they feel no resistance, no stress, no boredom, and no fatigue.** Your computer can work 24/7 making calculations for `www.climateprediction.net` — a distributed computing project to model the world's climate change. Or, have your computer crunch numbers for `Rosetta@home` — a site that models proteins to help cure major illnesses. Will you feel sorry for my computer because it's working so hard? Will the computer complain? No.

  You can make demands, give the computer its orders, and crack the whip. Will you (or should you) feel the least bit guilty? Not at all.

- **Computers move ideas, not paper.** Not long ago, whenever you wanted to send a message to someone, you hired a messenger. The messenger mounted a horse and delivered your message personally. The message was recorded on paper or parchment or a clay tablet or whatever other physical medium was available at the time.

  This whole process seems wasteful now, but that's only because you and I are sitting comfortably in the electronic age. Messages are ideas, and physical objects like ink, paper, and horses have little or nothing to do with real ideas; they're just temporary carriers of ideas (even though people used them for

several centuries to carry ideas). Nevertheless, the ideas themselves are paperless, horseless, and messengerless.

The neat thing about computers is that they carry ideas efficiently. They carry nothing but the ideas, a couple of photons, and some electrical power. They do this with no muss, no fuss, and no extra physical baggage.

When you start dealing efficiently with ideas, something very nice happens: Suddenly, all overhead is gone. Instead of pushing paper and trees, you're pushing numbers and concepts. Without the overhead, you can do things much faster and do things that are far more complex than ever.

What You Can Do with Java

It would be nice if all this complexity were free, but, unfortunately, it isn't. Someone has to think hard and decide exactly what to ask the computer to do. After that thinking takes place, someone has to write a set of instructions for the computer to follow.

Given the current state of affairs, you can't write these instructions in English or any other language that people speak. Science fiction is filled with stories about people who make simple requests of robots and get back disastrous, unexpected results. English and other such languages are unsuitable for communication with computers, for several reasons:

- **An English sentence can be misinterpreted.** "No parking. Violators will be towed away at the owner's expense." So, the parking space's owner will pay to tow my car? That's not so bad!
- **It's difficult to weave a complicated command in English.** "Join flange A to protuberance B, making sure to connect only the outermost lip of flange A to the larger end of the protuberance B while joining the middle and inner lips of flange A to grommet C."
- **English sentences have lots of extra baggage.** "Sentences have unneeded words."
- **English can be difficult to interpret.** "John Wiley & Sons, Inc. shall pay net earnings to the Author ('Barry Burd') or the Author's assigns upon submittal of *Java For Dummies,* 9th Edition ('the Work') minus prepaid and accrued expenses and deferred charges, either directly or indirectly, within the meaning of section 4942(j)(3) or 4965(k)(5) for calendar year 2024."

To tell a computer what to do, you have to use a special language to write terse, unambiguous instructions. A special language of this kind is called a *computer programming language.* A set of instructions written in such a language is called a *program.* When looked at as a big blob, these instructions are called *software* or *code.* Here's what code looks like when it's written in Java:

```
void main() {
    var checkAmount = 1257.63;
    println("Pay to the order of Dr. Barry Burd $" + checkAmount);
}
```

You may argue that ChatGPT and other generative AI tools narrow the gap between informal English and carefully written code. That's true to some extent. But computer code isn't going away anytime soon. Generative AI may hallucinate. And, in some critical applications, hallucinations are unacceptable. Besides, GPT tools don't create themselves. At this very moment (whenever you're reading my book, that is), thousands of people around the world are writing code to make GPT tools faster and more versatile. Programming isn't a dying art. It's a growing endeavor.

Why You Should Use Java

It's time to celebrate! You've just picked up a copy of *Java For Dummies,* 9th Edition, and you're reading Chapter 1. At this rate, you'll be an expert Java programmer* in no time at all, so rejoice in your eventual success by throwing a big party.

To prepare for the party, I'll bake a cake. I'm lazy, so I'll use a ready-to-bake cake mix. Let me see: Add water to the mix and then add butter and eggs — hey, wait! I just looked at the list of ingredients. What's MSG? And what about propylene glycol? That's used in antifreeze, isn't it?

I'll change plans and make the cake from scratch. Sure, it's a little harder, but that way, I get exactly what I want.

Computer programs work the same way: You can use somebody else's program or write your own. If you use somebody else's program, you use whatever you get. When you write your own program, you can tailor the program especially for your needs.

* In professional circles, a developer's responsibilities are usually broader than those of a programmer. But, in this book, I use the terms *programmer* and *developer* almost interchangeably.

Writing computer code is a big, worldwide industry. Companies do it, freelance professionals do it, hobbyists do it — all kinds of people do it. A typical big company has teams, departments, and divisions that write programs for the company. But you can write programs for yourself or for someone else, for a living or for fun. In a recent estimate, the number of lines of code written each day by programmers in the world exceeds the number of methane molecules on the planet Jupiter.** Take almost anything that can be done with a computer — with the right amount of time, you can write your own program to do it. (Of course, the "right amount of time" may be quite long, but that's not the point. Many interesting and useful programs can be written in hours or even minutes.)

Gaining Perspective: Where Java Fits In

Here's a brief history:

- **1954–1957: FORTRAN is developed.**

  FORTRAN was the first modern computer programming language. For scientific programming, FORTRAN is a real racehorse. Year after year, FORTRAN is a leading language among computer programmers throughout the world.

- **1959: Grace Hopper at Remington Rand develops the COBOL programming language.**

  The letter *B* in COBOL stands for *Business,* and business is just what COBOL is all about. The language's primary feature is the processing of one record after another, one customer after another, or one employee after another.

  Within a few years after its initial development, COBOL became the most widely used language for business data processing.

  (Fun fact: I once foolishly tried to write a COBOL program to create musical compositions. Talk about picking the wrong tool for the job! It was like brushing my teeth with a nail file.)

- **1960: John McCarthy writes a paper on his new LISP programming language.**

  LISP sets the stage for a style of programming called *functional programming*. For a few decades, functional programming waits in the background while other programming techniques take center stage.

** I made up this fact all by myself.

- **1967: Scientists at the Norwegian Computing Center develop the Simula 67 language.**

  With Simula 67, object-oriented programming is born.

- **1972: Dennis Ritchie at AT&T Bell Labs develops the C programming language.**

  The "look and feel" that you see in this book's examples comes from the C programming language. Code written in C uses curly braces, `if` statements, `for` statements, and other elements.

  In terms of power, you can use C to solve the same problems that you can solve by using FORTRAN or Java or any other modern programming language. The difference between one programming language and another isn't power — the difference is ease and appropriateness of use. That's where the Java language excels.

- **1986: Bjarne Stroustrup (also at AT&T Bell Labs) develops C++.**

  Unlike its C language ancestor, the C++ language supports object-oriented programming. This support represents a huge step forward. (See the next section in this chapter.)

- **May 23, 1995: Sun Microsystems releases its first official version of the Java programming language.**

  Java improves upon the concepts in C++. Java not only supports object-oriented programming but also *enforces the use of* object-oriented programming.

  Additionally, Java is a great general-purpose programming language. A program written in Java runs seamlessly on all major platforms, including Windows, Macintosh, and Linux. With Java, you can write windowed applications, build and explore databases, control handheld devices, and more. Within five short years, the Java programming language has 2.5 million developers worldwide. (I know — I have a commemorative T-shirt to prove it.)

  By 2001, Java has become the number-one language on the world-famous TIOBE Programming Community Index.

- **2007-2017: Java grows up.**

  For a while, updates to Java come very slowly. With disagreements over the implementation of new features, the gap between Java 6 and Java 7 releases is a whopping five years.

  In the meantime, Oracle Corporation purchases Sun Microsystems — a terrific bargain for only $7.4 billion! Then, in 2014, Oracle releases Java 8 — the first Java version with functional programming capabilities.

  For most of the months between 2001 and 2017, Java has maintained its top position on the TIOBE Index.

» **August 2017: Oracle announces its plan to release new versions of Java every six months.**

The release of Java 9 in September 2017 is followed by the rollout of Java 10 in March 2018. Up next is Java 11 in September 2018.

In September 2023, Java 21 is a *l*ong-*t*erm *s*upport (LTS) release. This means that Oracle promises to keep Java 21 running smoothly at least until September 2028. These LTS releases come every two years, so the next rock-solid, take-no-prisoners version of Java is Java 25 in September 2025.

The new release cycle has injected energy into the evolution of the Java programming language.

Java technology powers applications from companies like Microsoft, Uber, LinkedIn, PayPal, Netflix, Airbnb, Google, eBay, Spotify, TripAdvisor, Intel, Pinterest, and Groupon.* The job search site `Monster.com` says:

> "Java is one of the most popular programming languages in use, so it's no surprise it came in as the No. 1 skill tech companies were looking for. According to Oracle, 3 billion mobile phones run Java, along with 125 million TV devices and 89% of desktop computers in the U.S. Java is everywhere and the demand for strong developers is high."**

Well, I'm impressed.

Three Ways to Write Computer Programs

It's 4 in the morning. I'm dreaming about the history course I failed in high school. The teacher is yelling at me: "You have two days to study for the final exam, but you won't remember to study. You'll forget and feel guilty, guilty, guilty."

Suddenly, the phone rings. I'm awakened abruptly from my deep sleep. (Sure, I disliked dreaming about the history course, but I like being awakened even less.) At first, I drop the telephone on the floor. After fumbling to pick it up, I issue a grumpy, "Hello. Who's this?" A voice answers, "I'm a reporter from the Reuters

---

Sources:

*`https://codegym.cc/groups/posts/771-is-java-still-relevant-what-big-companies-use-it`

**`www.monster.com/career-advice/article/programming-languages-you-should-know`

news agency. I'm writing an article about Java, and I need to know all about the programming language in seven words or less. Can you explain it?"

My mind is too hazy. I can't think. So I say the first thing that comes to my mind and then go back to sleep.

Come morning, I hardly remember the conversation with the reporter. In fact, I don't remember how I answered the question. Did I utter a few obscenities and then go back to sleep?

I put on my robe and rush to my computer. When I visit the Reuters website, I see this enormous headline:

Burd Calls Java "A Great Language in Many Different Ways"

A word you can use to impress people

Today's vocabulary word is the word *paradigm*.

First things first: How do you pronounce the word *paradigm*? The *g* is silent. You say "PAA-ra-dime."

Next, what in the world does *paradigm* mean? I looked up *paradigm* in a few dictionaries, and none of the definitions impressed me much. To my mind, a *paradigm* is a way of thinking about something. For example, people used to think about the sun orbiting the Earth. But, in 1543, Copernicus proposed that the Earth orbits the sun. The Copernican Revolution was a paradigm shift.

Over the years, several computer programming paradigms have emerged. The following sections describe three of them: imperative programming, object-oriented programming, and functional programming.

Imperative programming

Once upon a time, almost every programming language was imperative in nature. With an imperative language, your programs consist mostly of "Do this, then do that" commands. Suppose that you run a small business and you have the three employees listed in Table 1-1.

TABLE 1-1 A Table of Employees

Name	Hours Per Week	Pay
Alice	40	450.00
Bob	20	200.00
Carol	35	300.00

You want each of your employees to work an additional hour every week. In exchange for this, you'll pay each employee $50 more. Here's how you might do it with an imperative style:

```
For each of my employees:
    Get the employee's hours per week,
    add 1 to that number,
    make that the new value of the employee's hours per week.
    Get the employee's current pay,
    add $50 to that amount,
    make that amount the new value of the employee's pay.
```


REMEMBER

What you see here isn't a real computer program. It's an informal, English-language description of the way a real program might work. Since it's not real code, we call it *pseudocode*.

In an *imperative program*, your code is simply a long list of commands. You may set aside some commands to be run separately and give that block of commands a name. But, one way or another, the code's fundamental building blocks are individual commands.

A typical command either gets a stored value or sets a stored value. For example, one command may get an employee's current pay while another command sets the employee's new pay. With imperative programming, each value is loosely connected to all the other values. For example, when your code changes an employee's hours per week, the code happens to change the employee's pay. Other than that, there may be no relationship between hours per week and pay. Each value is an entity on its own.

Object-oriented programming

I often say that imperative programming is "verbs first" and that object-oriented programming (OOP) is "nouns first." With imperative programming, you begin with verbs like *get* and *add.* With object-oriented programming, you begin by defining the kinds of things that your code deals with.