LEARNING MADE EASY

7th Edition

Java®

for dummies®
A Wiley Brand

Use the new features and tools in Java 9

Create basic Java objects and reuse code

Handle exceptions and events

Barry Burd, PhD

Author of Java Programming for Android Developers For Dummies

Java®

7th Edition

by Barry Burd, PhD

for dummies®
A Wiley Brand

Java® For Dummies®, 7th Edition

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

Library of Congress Control Number: 2017932837

ISBN: 978-1-119-23555-2; 978-1-119-23558-3 (ebk); 978-1-119-23557-6 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Table of Contents

Introduction

Java is good stuff. I've been using it for years. I like Java because it's orderly. Almost everything follows simple rules. The rules can seem intimidating at times, but this book is here to help you figure them out. So, if you want to use Java and you want an alternative to the traditional techie, soft-cover book, sit down, relax, and start reading *Java For Dummies,* 7th Edition.

How to Use This Book

I wish I could say, "Open to a random page of this book and start writing Java code. Just fill in the blanks and don't look back." In a sense, this is true. You can't break anything by writing Java code, so you're always free to experiment.

But let me be honest. If you don't understand the bigger picture, writing a program is difficult. That's true with any computer programming language — not just Java. If you're typing code without knowing what it's about and the code doesn't do exactly what you want it to do, you're just plain stuck.

In this book, I divide Java programming into manageable chunks. Each chunk is (more or less) a chapter. You can jump in anywhere you want — Chapter 5, Chapter 10, or wherever. You can even start by poking around in the middle of a chapter. I've tried to make the examples interesting without making one chapter depend on another. When I use an important idea from another chapter, I include a note to help you find your way around.

In general, my advice is as follows:

- If you already know something, don't bother reading about it.
- If you're curious, don't be afraid to skip ahead. You can always sneak a peek at an earlier chapter, if you really need to do so.

Conventions Used in This Book

Almost every technical book starts with a little typeface legend, and *Java For Dummies*, 7th Edition, is no exception. What follows is a brief explanation of the typefaces used in this book:

- New terms are set in *italics.*
- If you need to type something that's mixed in with the regular text, the characters you type appear in bold. For example: "Type **MyNewProject** in the text field."
- You also see this `computerese` font. I use computerese for Java code, filenames, web page addresses (URLs), onscreen messages, and other such things. Also, if something you need to type is really long, it appears in computerese font on its own line (or lines).
- You need to change certain things when you type them on your own computer keyboard. For instance, I may ask you to type

  ```
  public class Anyname
  ```

  which means that you type **public class** and then some name that you make up on your own. Words that you need to replace with your own words are set in *`italicized computerese`*.

What You Don't Have to Read

Pick the first chapter or section that has material you don't already know and start reading there. Of course, you may hate making decisions as much as I do. If so, here are some guidelines that you can follow:

- If you already know what kind of an animal Java is and know that you want to use Java, skip Chapter 1 and go straight to Chapter 2. Believe me, I won't mind.
- If you already know how to get a Java program running, and you don't care what happens behind the scenes when a Java program runs, skip Chapter 2 and start with Chapter 3.
- If you write programs for a living but use any language other than C or C++, start with Chapter 2 or 3. When you reach Chapters 5 and 6, you'll probably find them to be easy reading. When you get to Chapter 7, it'll be time to dive in.

- If you write C (not C++) programs for a living, start with Chapters 2, 3, and 4 and just skim Chapters 5 and 6.
- If you write C++ programs for a living, glance at Chapters 2 and 3, skim Chapters 4 through 6, and start reading seriously in Chapter 7. (Java is a bit different from C++ in the way it handles classes and objects.)
- If you write Java programs for a living, come to my house and help me write *Java For Dummies,* 8th Edition.

If you want to skip the sidebars and the Technical Stuff icons, please do. In fact, if you want to skip anything at all, feel free.

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 . . . well, 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 that you can't use to run this code are ancient things that are more than ten years old (give or take a few years).
- **I assume that you can navigate through 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 practice the stuff in this book, you're typing code on the keyboard, not pointing and clicking the mouse.

  On those rare occasions when you need to drag and drop, cut and paste, or plug and play, I guide you carefully through the steps. But your computer may be configured in any of several billion ways, and my instructions may not quite fit your special situation. When you reach one of these platform-specific tasks, try following the steps in this book. If the steps don't quite fit, consult a book with instructions tailored to your system.
- **I assume that you can think logically.** That's all there is to programming in Java — thinking logically. If you can think logically, you've got 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: I've tried to make the book interesting for experienced programmers yet accessible to people with little or no programming experience. This means that I don't assume any 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.

How This Book Is Organized

This book is divided into subsections, which are grouped into sections, which come together to make chapters, which are lumped finally into five parts. (When you write a book, you get to know your book's structure pretty well. After months of writing, you find yourself dreaming in sections and chapters when you go to bed at night.) The parts of the book are listed here.

Part 1: Getting Started with Java

This part is your complete, executive briefing on Java. It includes some "What is Java?" material and a jump-start chapter — Chapter 3. In Chapter 3, you visit the major technical ideas and dissect a simple program.

Part 2: Writing Your Own Java Program

Chapters 4 through 6 cover the fundamentals. These chapters describe the things that you need to know so that you can get your computer humming along.

If you've written programs in Visual Basic, C++, or any another language, some of the material in Part 2 may be familiar to you. If so, you can skip some sections or read this stuff quickly. But don't read too quickly. Java is a little different from some other programming languages, especially in the things that I describe in Chapter 4.

Part 3: Working with the Big Picture: Object-Oriented Programming

Part 3 has some of my favorite chapters. This part covers the all-important topic of object-oriented programming. In these chapters, you find out how to map solutions to big problems. (Sure, the examples in these chapters aren't big, but the examples involve big ideas.) In bite-worthy increments, you discover how to design classes, reuse existing classes, and construct objects.

Have you read any of those books that explain object-oriented programming in vague, general terms? I'm proud to say that *Java For Dummies*, 7th Edition, isn't like that. In this book, I illustrate each concept with a simple-yet-concrete program example.

Part 4: Smart Java Techniques

If you've tasted some Java and you want more, you can find what you need in this part of the book. This part's chapters are devoted to details — the things that you don't see when you first glance at the material. After you read the earlier parts and write some programs on your own, you can dive in a little deeper by reading Part 4.

Part 5: The Part of Tens

The Part of Tens is a little Java candy store. In the Part of Tens, you can find lists — lists of tips for avoiding mistakes, for finding resources, and for all kinds of interesting goodies.

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, a side comment, or something 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 of setting 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 — something helpful 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

Question: What's stronger than a Tip, but not as strong as a Warning?

Answer: A 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, but some are a bit more challenging. When you first start putting things 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 (`www.allmycode.com/JavaForDummies`) for hints and solutions.

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

TECHNICAL STUFF

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 (more geeky) 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 Android program. To get this Cheat Sheet, simply go to `www.dummies.com` and type **Java For Dummies Cheat Sheet** in the Search box.

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. If email and chat aren't your favorites, you can reach me instead on Twitter (`@allmycode`) and on Facebook (`www.facebook.com/allmycode`). And don't forget — for the latest updates, visit this book's website. The site's address is `www.allmycode.com/JavaForDummies`.

1

Getting Started with Java

IN THIS PART . . .

Find out about the tools you need for developing Java programs.

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

See your first complete Java program.

IN THIS CHAPTER

» What Java is

» Where Java came from

» Why Java is so cool

» How to orient yourself to object-oriented programming

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.** Computers are our electronic slaves. I have my computer working 24/7 doing calculations for `Cosmology@Home` — a distributed computing project to investigate models describing the universe. Do I feel sorry for my computer because it's working so hard? Does the computer complain? Will the computer report me to the National Labor Relations Board? No.

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

- **Computers move ideas, not paper.** Not long ago, when you wanted to send a message to someone, you hired a messenger. The messenger got on his or her horse and delivered your message personally. The message was on paper, parchment, a clay tablet, or whatever 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 things like ink, paper, and horses have little or nothing to do with real ideas; they're just temporary carriers for ideas (even though people used them to carry ideas for several centuries). 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 a little 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 the 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 before.

What You Can Do with Java

It would be so 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 say simple things to 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.** "Chew one tablet three times a day until finished."
- **It's difficult to weave a very 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."
- **An English sentence has lots of extra baggage.** "Sentence has unneeded words."
- **English is difficult to interpret.** "As part of this Publishing Agreement between John Wiley & Sons, Inc. ('Wiley') and the Author ('Barry Burd'), Wiley shall pay the sum of one-thousand-two-hundred-fifty-seven dollars and sixty-three cents ($1,257.63) to the Author for partial submittal of *Java For Dummies,* 7th Edition ('the Work')."

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:

```
public class PayBarry {

    public static void main(String args[]) {
        double checkAmount = 1257.63;
        System.out.print("Pay to the order of ");
        System.out.print("Dr. Barry Burd ");
        System.out.print("$");
        System.out.println(checkAmount);
    }
}
```

Why You Should Use Java

It's time to celebrate! You've just picked up a copy of *Java For Dummies,* 7th 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.

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 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 United States alone 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 very long, but that's not the point. Many interesting and useful programs can be written in hours or even minutes.)

---

*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.

**I made up this fact all by myself.

Getting Perspective: Where Java Fits In

Here's a brief history of modern computer programming:

- **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.

- **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 so on.

  In terms of power, you can use C to solve the same problems that you can solve by using FORTRAN, Java, or any other modern programming language. (You can write a scientific calculator program in COBOL, but doing that sort of thing would feel really strange.) 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 (again at AT&T Bell Labs) develops C++.**

  Unlike its C language ancestor, the language C++ 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's "Write Once, Run Anywhere" philosophy makes the language ideal for distributing code across the Internet.

  Additionally, Java is a great general-purpose programming language. With Java, you can write windowed applications, build and explore databases, control handheld devices, and more. Within five short years, the Java programming language had 2.5 million developers worldwide. (I know. I have a commemorative T-shirt to prove it.)

» **November 2000: The College Board announces that, starting in the year 2003, the Computer Science Advanced Placement exams will be based on Java.**

  Wanna know what that snot-nosed kid living down the street is learning in high school? You guessed it — Java.

» **2002: Microsoft introduces a new language, named C#.**

  Many of the C# language features come directly from features in Java.

» **June 2004: Sys-Con Media reports that the demand for Java programmers tops the demand for C++ programmers by 50 percent (**`http://java.sys-con.com/node/48507`**).**

  And there's more! The demand for Java programmers beats the combined demand for C++ and C# programmers by 8 percent. Java programmers are more employable than Visual Basic (VB) programmers by a whopping 190 percent.

» **2007: Google adopts Java as the primary language for creating apps on Android mobile devices.**

» **January 2010: Oracle Corporation purchases Sun Microsystems, bringing Java technology into the Oracle family of products.**

» **June 2010: *eWeek* ranks Java first among its "Top 10 Programming Languages to Keep You Employed" (**`www.eweek.com/c/a/Application-Development/Top-10-Programming-Languages-to-Keep-You-Employed-719257`**).**

» **2016: Java runs on 15 billion devices (**`http://java.com/en/about`**), with Android Java running on 87.6 percent of all mobile phones worldwide (**`www.idc.com/prodserv/smartphone-os-market-share.jsp`**).**

  Additionally, Java technology provides interactive capabilities to all Blu-ray devices and is the most popular programming language in the TIOBE Programming Community Index (`www.tiobe.com/index.php/content/paperinfo/tpci`), on PYPL: the PopularitY of Programming Language Index (`http://sites.google.com/site/pydatalog/pypl/PyPL-PopularitY-of-Programming-Language`), and on other indexes.

  Well, I'm impressed.

Object-Oriented Programming (OOP)

It's three in the morning. I'm dreaming about the history course that 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 *New York Times.* I'm writing an article about Java, and I need to know all about the programming language in five 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 tell the reporter where he could put his article about Java?

I put on my robe and rush out to my driveway. As I pick up the morning paper, I glance at the front page and see this 2-inch headline:

Burd Calls Java "A Great Object-Oriented Language"

Object-oriented languages

Java is object-oriented. What does that mean? Unlike languages, such as FORTRAN, that focus on giving the computer imperative "Do this/Do that" commands, object-oriented languages focus on data. Of course, object-oriented programs still tell the computer what to do. They start, however, by organizing the data, and the commands come later.

Object-oriented languages are better than "Do this/Do that" languages because they organize data in a way that helps people do all kinds of things with it. To modify the data, you can build on what you already have rather than scrap everything you've done and start over each time you need to do something new. Although computer programmers are generally smart people, they took a while to figure this out. For the full history lesson, see the sidebar "The winding road from FORTRAN to Java" (but I won't make you feel guilty if you don't read it).