Rogers Cadenhead

EIGHTH EDITION

Covers Java 9 and Android

Sams Teach Yourself

# Java™ in 24 Hours

SAMS

Rogers Cadenhead

# Sams Teach Yourself Java in 24 Hours

Eighth Edition

Pearson

**Sams Teach Yourself Java in 24 Hours, Eighth Edition**

Copyright © 2018 by Pearson Education, Inc.

ISBN-13: 978-0-672-33794-9
ISBN-10: 0-672-33794-0

Library of Congress Control Number: 2017946911
Printed in the United States of America

# Contents at a Glance

# Contents

# Accessing the Free Web Edition

Your purchase of this book in any format, print or electronic, includes access to the corresponding Web Edition, which provides several special features to help you learn:

- The complete text of the book online
- Updates and corrections as they become available

The Web Edition can be viewed on all types of computers and mobile devices with any modern web browser that supports HTML5.

To get access to the Web Edition of *Sams Teach Yourself Java in 24 Hours, Eighth Edition,* all you need to do is register this book:

1. Go to www.informit.com/register
2. Sign in or create a new account
3. Enter ISBN: 9780672337949
4. Answer the questions as proof of purchase

The Web Edition will appear under the Digital Purchases tab on your Account page.

Click the Launch link to access the product.

# Introduction

As the author of computer books, I spend a lot of time lurking in the computer section of bookstores, observing the behavior of readers while I'm pretending to read the latest issue of *Soap Opera Digest* magazine.

Because of my research, I've learned that if you have picked up this book and turned to this introduction, I only have 13 more seconds before you put it down and head to the coffee bar for a double-tall-decaf-skim-with-two-shots-of-vanilla-hold-the-whip latte.

So I'll keep this brief: Computer programming with Java is easier than it looks.

I'm not supposed to tell you that because thousands of programmers have used their Java skills to get high-paying jobs in software development, server programming, and Android app creation. The last thing any programmer wants is for the boss to know that anyone with persistence and a little free time can learn this language, the most popular programming language on the planet. By working your way through each of the one-hour tutorials in *Sams Teach Yourself Java in 24 Hours*, you'll be able to learn Java programming quickly.

Anyone can learn how to write computer programs, even if you can't program a DVR. Java is one of the best programming languages to learn because it's a useful, powerful, modern technology that's embraced by companies around the world.

This book is aimed at non-programmers, new programmers who think they hate this stuff, and experienced programmers who want to get up to speed swiftly with Java. It uses Java 9, the latest and greatest version of the language.

Java is an enormously popular programming language because of the things it makes possible. You can create programs that feature a graphical user interface, connect to web services, run on an Android phone or tablet, and more.

This language turns up in some amazing places. One of them is Minecraft, the gaming phenomenon written entirely in Java. (In this book you learn how to create Java programs that run in that game alongside creepers and zombie pigmen!)

This book teaches Java programming from the ground up. It introduces the concepts in English instead of jargon with step-by-step examples of working programs you will create. Spend 24 hours with this book and you'll be writing your own Java programs, confident in your ability to use the language and learn more about it. You also will have skills that are becoming increasingly important—such as Internet computing, graphical user interface design, app creation, and object-oriented programming.

These terms might not mean much to you now. In fact, they're probably the kind of thing that makes programming seem intimidating and difficult. However, if you can use a computer to create a photo album on Facebook, pay your taxes, or work an Excel spreadsheet, you can learn to write computer programs by reading *Sams Teach Yourself Java in 24 Hours*.

NOTE

At this point, if you would rather have coffee than Java, please reshelve this book with the front cover facing outward on an endcap near a lot of the store's foot traffic.

# HOUR 1
# Becoming a Programmer

**This Hour's To-Do List:**

- ▶ Find out the reasons to learn Java.
- ▶ Discover how programs work.
- ▶ Select a Java development tool.
- ▶ Get ready to write your first program.

You've probably heard that computer programming is insanely difficult. It requires a degree in computer science, thousands of dollars in computer hardware and software, a keen analytical mind, the patience of Job, and a strong liking for caffeinated drinks.

Aside from the part about caffeine, you heard wrong. Programming is easier than you might think, despite what programmers have been telling people for years to make it easier for us to land high-paying jobs.

This is a great time to learn programming. Countless programming tools are being made available as free downloads on the Web, and thousands of programmers distribute their work as open source so other people can examine how the software was written, fix errors, and contribute improvements. In a recovering economy, many companies are hiring programmers.

It's a great time to learn Java, because the language is everywhere. Billions of mobile devices use Android, an operating system whose apps are all written in Java. If you have an Android phone, you've been enjoying the work of Java programmers every time you look up a movie, rock out on streaming radio, or sling an antagonistic avian at a poorly built fortress of swine.

This book aims to teach Java programming to three kinds of people:

1. Nervous novices who never tried to program before
2. Bitter beginners who tried programming but hated it like Lord Voldemort hates orphaned British schoolchildren
3. Impatient intellectuals who know another programming language and want to get up to speed quickly on Java

To achieve this goal, this book uses the English language as much as possible instead of technical jargon or obscure acronyms. All new programming terms are thoroughly explained as they are introduced.

If I've succeeded, you will finish this book with enough programming skills to be a danger to yourself and others. You'll be able to write programs, plunge into programming classes and books with more confidence, and learn new languages more easily. (Programming languages, to be clear. This book won't help you master Spanish, Esperanto, or Klingon.)

You also will have skills with Java, the most widely used programming language on the planet.

The first hour of this book provides an introduction to programming and guidance on setting up your computer so you can use it to write and run Java programs.

## Choosing a Language

If you're comfortable enough with a computer to prepare a nice-looking résumé, balance a checkbook, or share your vacation photos on Instagram, you can create computer software.

The key to learning how to program is to start with the right language. The programming language you choose often depends on the tasks you want to accomplish. Each language has strengths and weaknesses. Back in my day, young whippersnappers, people learned to program with the BASIC language because it was created with beginners in mind.

NOTE

The BASIC language was invented to be easy for students to learn (the B in BASIC stands for Beginner's). The downside to using some form of BASIC is that it's easy to fall into sloppy programming habits with the language.

The most popular language that employs BASIC today is Visual Basic, a programming language from Microsoft that has moved far beyond its roots. VB, as it also is called, is designed for creating programs to run on computers and mobile devices that use the Windows operating system. Another popular language is PHP, a scripting language for creating websites. Other widely used languages you may have heard about include C++, Ruby, Javascript, and Python.

Each of these languages has its adherents, but the most widely taught in computer science classes at the high school and collegiate level is Java.

The Java programming language, which is offered by Oracle, is more difficult to learn than some other languages such as VB and PHP, but it's a great starting place for several reasons. One advantage of learning Java is that you can use it across a variety of operating systems and computing environments. Java programs can be desktop software, web applications, web servers, Android apps, and more, running on Windows, Mac, Linux, and other operating systems. This versatility is referenced by the ambitious early Java slogan “Write once, run anywhere.”

NOTE

Early Java programmers had a less flattering slogan: “Write once, debug everywhere.” The language has come a long way, baby, since the first version was released in 1996.

Another important advantage is that Java requires a highly organized approach for getting programs to work. You must be particular about how you write programs and how they store and alter data.

When you start writing Java programs, you might not see the language’s persnickety behavior as an advantage. You could tire of writing a program and having several errors to fix before the program even can be run. The benefit of this extra effort is that the software you create is more reliable, useful, and error-free.

In the coming hours, you learn all of Java’s rules and the pitfalls to avoid.

Java was invented by the Canadian computer scientist James Gosling as a better way to create computer programs. While working at Sun Microsystems in 1991, Gosling was unhappy with the way the C++ programming language was performing on a project, so he created a new language that did the job better. It’s a matter of contentious debate whether Java is superior to other programming languages, of course, but the success of the language demonstrates the strength of his initial design. Fifteen billion devices across the world are running Java, a number so amazing I’m going to repeat it. Fifteen billion! More than 1,000 books have been published about the language since its introduction. (This is my twentieth.)

Regardless of whether Java is the best language, it definitely is a great language to learn. You get your first chance to try out Java during Hour 2, “Writing Your First Program.”

Learning one programming language makes it much easier to learn subsequent languages. Many are similar to each other, so you aren’t starting from scratch when you plunge into a new one. For instance, many C++ and Smalltalk programmers find it fairly easy to learn Java because Java borrows ideas from those earlier languages. Similarly, C# adopts many ideas from Java, so it’s easier to pick up for Java programmers.

NOTE

C++ is mentioned several times this hour, so you might be tripping over the term, wondering what it means—and how it’s pronounced. C++ is pronounced “C-Plus-Plus,” and it’s a programming language developed by Danish computer scientist Bjarne Stroustrop at Bell Laboratories. C++ is an enhancement of the C programming language, hence the Plus-Plus part of the name. Why not just call it C+? The Plus-Plus part is a computer programming joke you’ll understand later in this book.

## Telling the Computer What to Do

A computer program, also called software, is a way to tell a computer to perform a task. Everything that the computer does, from booting up to shutting down, is done by a program. Mac OS X is a program; Minecraft is a program; the driver software that controls your printer is a program; even the dreaded blue screen of death on a crashed Windows PC is a program.

Computer programs are made up of a list of commands the computer handles in a specific order when the program is run. Each command is called a statement.

If your house had its own butler and you were a control freak with a Type-A personality, you could give your servant a detailed set of instructions to follow every day, like this:

> Dear Mr. Jeeves,
>
> Please take care of these errands for me while I'm out asking Congress for a bailout:
>
> Item 1: Vacuum the living room.
>
> Item 2: Go to the store.
>
> Item 3: Pick up soy sauce, wasabi, and as many California sushi rolls as you can carry.
>
> Item 4: Return home.
>
> Sincerely, your lord and master,
>
> Bertie Wooster

If you tell a human butler what to do, there's a certain amount of leeway in how your requests are fulfilled. If California rolls aren't available, Jeeves could bring Boston rolls home instead.

Computers don't do leeway. They follow instructions literally. The programs that you write are followed precisely, one instruction at a time.

The following example is a three-line computer program, written in BASIC. Take a look at it, but don't worry too much about what each line is supposed to mean.

```
1 PRINT "Hey Tom, it's Bob from the office down the hall."
2 PRINT "It's good to see you buddy, how've you been?"
3 INPUT A$
```

Translated into English, this program is equivalent to giving a computer the following to-do list:

> Dear personal computer,
>
> Item 1: Display the message, "Hey Tom, it's Bob from the office down the hall."
>
> Item 2: Ask the question, "It's good to see you buddy, how've you been?"

Item 3: Give the user a chance to answer the question.

Sincerely, your lord and master,

Ima Coder

Each line in a computer program is called a *statement*. A computer handles each statement in a program in a specific order, in the same way that a cook follows a recipe or Mr. Jeeves the butler follows the orders of Bertie Wooster. In BASIC, the line numbers are used to put the statements in the correct order. Other languages such as Java do not use line numbers, favoring different ways to tell the computer how to run a program.

Because of the way programs function, you can't blame the computer when something goes wrong as your program runs. The computer is doing exactly what you told it to do, so the blame for any errors usually lies with the programmer.

That's the bad news. The good news is you can't do any permanent harm. No computers will be injured as you learn to program in Java.

## How Programs Work

The collection of statements that make up a computer program is called its *source code*.

Most computer programs are written in the same way that you write an email—by typing each statement into a text window. Some programming tools come with their own source code editor and others can be used with any text-editing software.

When you have finished writing a computer program, you save the file to disk. Computer programs often have their own filename extension to indicate what type of file they are. Java programs must have the extension `.java`, as in `Calculator.java`.

### NOTE

Computer programs should be prepared as text files with no special formatting. Notepad, a text editor that comes with Windows, saves all files as unformatted text. You also can use TextEdit on Macs or the vi editor or emacs on Linux systems to create text files without formatting. An easier solution is coming up later this hour.

To run a program you have saved as a file, you need some help. The kind of help required depends on the programming language you're using. Some languages require an interpreter to run their programs. The interpreter examines each line of a computer program and executes that line, then proceeds to the next line. Many versions of BASIC are interpreted languages.

The biggest advantage of interpreted languages is that they are faster to test. When you are writing a BASIC program, you can try it out immediately, fix errors, and try again. The primary disadvantage is that interpreted languages run slower than other programs. Each line has to be translated into instructions the computer can run, one line at a time.

Other programming languages require a compiler. The compiler takes a program and translates it into a form that the computer can understand. It also makes the program run as efficiently as possible. The compiled program can be run directly without the need for an interpreter.

Compiled programs run more quickly than interpreted programs but take more time to test. You have to write your program and compile the whole thing before trying it out. If you find an error and fix it, you must compile the program again.

Java is unusual because it requires both a compiler and an interpreter. The compiler converts the statements that make up the program into bytecode. Once this bytecode has been created successfully, it can be run by an interpreter called the Java Virtual Machine.

The Java Virtual Machine, also called a JVM, is the thing that makes it possible for the same Java program to run without modification on different operating systems and different kinds of computing devices. The virtual machine turns bytecode into instructions that a particular device's operating system can execute.

NOTE

Java 9 introduces a new tool called JShell that acts like an interpreter, running a Java statement right when it is typed in. JShell works by putting the statement into a Java program, compiling that program into bytecode, and running it. This is a useful tool for learning and testing.

## When Programs Don't Work

Many new programmers become discouraged when they start to test their programs. Errors appear everywhere. Some of these are syntax errors, which are identified by the computer as it looks at the program and becomes confused by the way a statement has been written. Other errors are logic errors, which only are noticed by the programmer as the program is being tested (or might be overlooked entirely). Logic errors often cause it to do something unintended.

As you begin writing your own programs, you become well acquainted with errors. They're a natural part of the process. Programming errors are called *bugs*, a term that dates back a century or more to describe errors in technical devices.

The process of fixing errors also has its own term: *debugging*.

It's no coincidence that there are so many ways to describe errors. You get a lot of debugging experience as you learn programming—whether you want it or not.

NOTE

One of the first computer bugs was discovered in 1947 by a team that included the American computer scientist Grace Hopper. Hopper was testing a computer at Harvard when a relay malfunctioned. The cause wasn't a software problem—it was an actual bug! A team member debugged the computer by removing a dead moth and taped it into a logbook with the note, "First actual case of bug being found." The bug and logbook page can be viewed at www.doncio.navy.mil/CHIPS/ArticleDetails.aspx?id=3489.

## Choosing a Java Programming Tool

To start writing Java programs, you must have a Java programming tool. Several such programs are available for Java, including the simple Java Development Kit and the more sophisticated Eclipse, IntelliJ IDEA, and NetBeans. The latter three tools are each an integrated development environment (IDE), a powerful tool used by professional programmers to get work done.

Whenever Oracle releases a new version of Java, the first tool that supports it is the Java Development Kit (JDK).

To create the programs in this book, you must use JDK version 9 or a programming tool that works on top of it. The JDK is a set of free command-line tools for creating Java software. It lacks a graphical user interface, so if you have never worked in a non-graphical environment such as the Windows command prompt or Linux command-line interface, you will find it challenging to use the JDK.

The NetBeans IDE, also offered for free by Oracle, is a much easier way to write and test Java code than the JDK. NetBeans includes a graphical user interface, source code editor, user interface designer, and project manager. It works in complement to the JDK, running it behind the scenes, so you must have both tools on your system when you begin developing Java programs.

Most of the programs in this book were created with NetBeans, which you can download and install separately from the JDK. You can use other Java tools as long as they support JDK 9.

NOTE

You don't have to use NetBeans in this book. If you can use the JDK or another tool to create, compile, and run a program, those tasks are all that most projects require. NetBeans is covered because for readers of past editions it has proven easier than the JDK. I use NetBeans for most of my Java programming.