ORACLE®

JAVA

THE COMPLETE REFERENCE GUIDE 2019 EDITION

The Definitive Java Programming Guide

MR KOTIYANA PUBLISHING

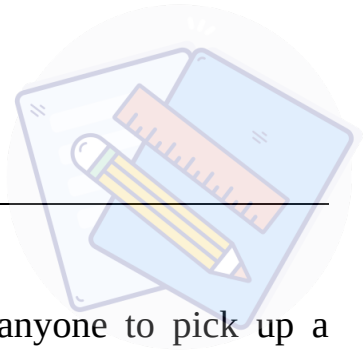
JAVA THE COMPLETE REFERENCE

11TH EDITION

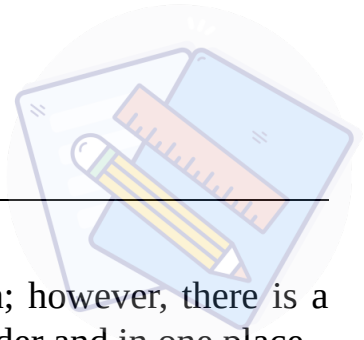
Table of Contents

CHAPTER 1 | INTRODUCTION

What This Book is about?



This java reference book was written as an answer for anyone to pick up a programming language and be productive. You will be able to start from scratch without having any previous exposure to any programming language. By the end of this book, you will have the skills to be a capable programmer, or at least know what is involved with how to read and write code. Afterward you should be armed with the knowledge required to feel confident in learning more. You should have general computer skills before you get started. After this you'll know what it takes to at least look at code without your head spinning.



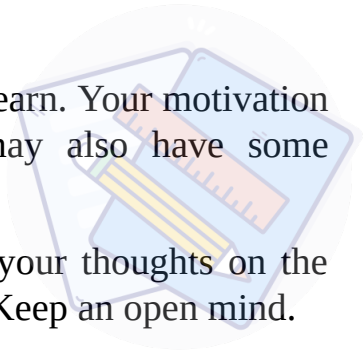
Why Read This Book?

You could go online and find videos and tutorials to learn; however, there is a distinct disadvantage when it comes to learning things in order and in one place.

Most YouTube or tutorial websites either gloss over a topic or dwell at a turtle's pace for an hour on a particular subject. Online content is often brief and doesn't go into much depth on any given topic. It is incomplete or still a work in progress. You'll often find yourself waiting weeks for another video or tutorial to come out.

Most online tutorials for Java are scattered, disordered, and in cohesive. It is difficult to find a good starting point and even more difficult to find a continuous list of tutorials to bring you to any clear understanding of the Java programming language. Just so you know, you should find the act of learning exciting. If not, then you'll have a hard time continuing through to the end of this book. To learn any new skill, a lot of patience is required.

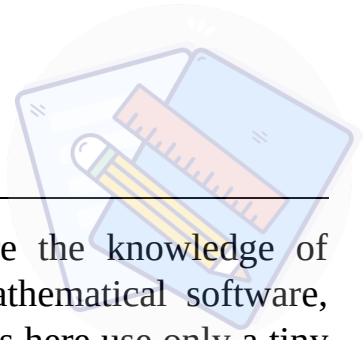
I remember asking an expert programmer how I'd learn to program. He told me to write a compiler. At that time, it seemed rather mean, but now I understand why he said that. It is like telling someone who wants to learn how to drive Formula 1 cars to go compete in a race. In both cases, the "learn" part was left out of the process of learning. It is very difficult to tell someone who wants to learn to write code where to begin.



However, it all really does start with your preparedness to learn. Your motivation must extend beyond the content of this book. You may also have some preconceived notions about what a programming is.

I can’t change that, but you should be willing to change your thoughts on the topic if you make discoveries contrary to your knowledge. Keep an open mind.

Computer artists often believe that programming is a technical subject that is incompatible with art. I find the contrary to be true. Programming is an art, much as literature and design is an art. Programming just has a lot of obscure rules that need to be understood for anything to work.

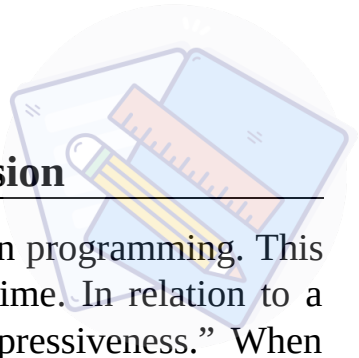


Do I Need to Know Math?

With complex rules in mind, does programming require the knowledge of complex mathematics? Actually, unless you program mathematical software, only a bit of geometry is nice to have. Most of the examples here use only a tiny bit of math to accomplish their purposes.

Mathematics and programming do overlap quite a lot in their methodology. Math taught in schools provides a single solution. Programming results tend to behave a bit like a math proof, but only the proof isn’t just another bit of math. Rather, the proof of your code means that your zombies chase after humans.

A considerable amount of math required has been done for you by the computer. It is just up to you to know what math to use, and then plug in the right variables.



Programming as a Form of Expression

There is a deeper connection between words and meaning in programming. This connection is more mechanical, but flexible at the same time. In relation to a programming language, programmers like to call this “expressiveness.” When you write words in literature, you infer most if not all of the meaning. When programming inference isn’t implied, then you’re free to define your own meanings for words and invent words freely.

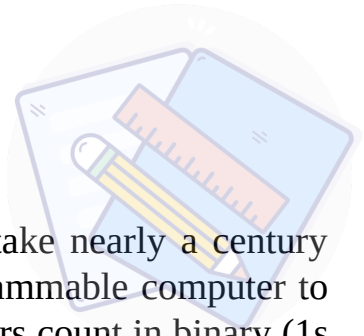
One common merging of art and code appears within a video game. Anytime characters can react to each other and react to user interaction, which conveys a greater experience. An artist, an animator, or an environment designer should have some ability to convey a richer experience to the audience. A character artist can show a monster to his or her audience; but to really experience the creature, the monster should glare, grunt, and attack in reaction to the audience.

A Brief History of Computer Programming:

How Programming Came to Be

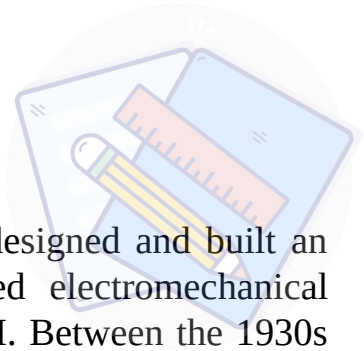
Mechanical Computers:

The first programmable computer is arguably the Babbage machine built by Charles Babbage in the 1820s. Being made of tens of thousands of moving parts and weighing several tons, it was a gigantic mechanical calculator that could display a 31-digit number. It was called the Difference Engine, and in 1824 Babbage won a gold medal from the British Astronomical Society for correcting a set of tables used to chart the movement of planets through the sky. In 1833, a countess named Augusta Ada King of Lovelace, commonly known as Ada Lovelace, met Babbage at a party where she was introduced to the Difference Engine. Several years later, it became evident that she understood the machine better than Charles himself. Despite being housewife to the Earl of Lovelace, she wrote several documents on the operation of the Difference Engine as well as its upgraded version, the Analytical Engine. She is often considered to be the first computer programmer for her work with the Difference Engine and its documentation.



Logic:

In 1848, George Boole gave us Boolean logic. It would take nearly a century between the Difference Engine and the first general programmable computer to make its appearance. Thanks to George, today our computers count in binary (1s and 0s), and our software thinks in terms of true or false. In 1887, Dorr Eugene Felt built a computing machine with buttons; thanks to him, our computers have keyboards. In the 1890s, a tabulating machine used paper with holes punched in it representing 1s and 0s to record the US census. Back then it saved US$2 million and 6 years of work. In the 1900s, between Nicola Tesla and Alexander Graham Bell, modern computers were imminent with the invention of the transistor. In 1936, Konrad Zuse made the Z1 computer, another mechanical computer like that of Babbage, but this time it used a tape with holes punched in it like the tabulating machine. He’d later move on to make the Z3 in 1941.



Computer Science:

In the 1940s, Alan Turing, a British computer scientist, designed and built an encrypted message decoder called the Bombe that used electromechanical switches for helping the Allied forces during World War II. Between the 1930s and the 1950s, Turing informed the computer scientists that computers can, in theory, solve anything calculable, calling this concept Turing completeness. All of these components began to lead toward our modern computer. In the mid-1940s, John Von Neumann demonstrated with the help of his theory that the computer can be built using simple components. This way the software that controls the hardware can add the complex behavior. Thanks to Tesla and Bell, the Z3 was made completely of electronic parts. It included the first use of logic while doing calculations, making it the first complete Turing-complete computer. In 1954, Gordon Teal at Texas Instruments introduced the silicon-based transistor, a key component for building solid-state electronics that are used today in every computer.