Introduction to Computer Programing and Java

Textbook: Java How to Program, 10/e
Late Objects Version

References & Reading

- The content is mainly selected (sometimes modified) from the original slides provided by the authors of the textbook

- Readings
  - Chapter 1- Introduction to Computers, Internet, and Java
  - A short history about programming Languages (https://en.wikipedia.org/wiki/History_of_programming_languages)

Outline

1.1 Introduction

- Computers (often referred to as hardware) are controlled by software (i.e., the instructions you write to command the computer to perform actions and make decisions).
- You’ll learn to write instructions commanding computers to perform tasks.
- Java is one of the world’s most widely used computer programming languages.

1.1 Introduction (Cont.)

- For many organizations, the preferred language for meeting their enterprise programming needs is Java.
- Java is also widely used for implementing Internet-based applications and software for devices that communicate over a network.
- According to Oracle, 97% of enterprise desktops, 89% of PC desktops, three billion devices (Fig. 1.1) and 100% of all Blu-ray Disc™ players run Java, and there are over 9 million Java developers. (`http://www.oracle.com/technetwork/articles/java/javaone12review-1863742.html`.)

Devices		
Airplane systems	ATMs	Automobile infotainment systems
Blu-ray Disc™ players	Cable boxes	Copiers
Credit cards	CT scanners	Desktop computers
e-Readers	Game consoles	GPS navigation systems
Home appliances	Home security systems	Light switches
Lottery terminals	Medical devices	Mobile phones
MRIs	Parking payment stations	Printers
Transportation passes	Robots	Routers
Smart cards	Smart meters	Smartpens
Smartphones	Tablets	Televisions
TV set-top boxes	Thermostats	Vehicle diagnostic systems

Fig. 1.1 | Some devices that use Java.

1.1 Introduction (Cont.)

- Java Standard Edition (Java SE) contains the capabilities needed to develop desktop and server applications.
  - Java How to Program, 10/e is based on Java Standard Edition 7 (Java SE 7) and Java Standard Edition 8 (Java SE 8)
- Java Enterprise Edition (Java EE) is geared toward developing large-scale, distributed networking applications and web-based applications.
- Java Micro Edition (Java ME) a subset of Java SE, geared toward developing applications for resource-constrained embedded devices, such as: Smartwatches, MP3 players, and more.

1.2 Hardware and Software

- Computers can perform calculations and make logical decisions phenomenally faster than human beings *can*.
- Today’s personal computers can perform billions of calculations in one second—more than a human can perform in a lifetime.
- *Supercomputers* are already performing *thousands of trillions (quadrillions)* of instructions per second!
- Computers process data under the control of sequences of instructions called computer programs.
- A computer consists of various devices referred to as hardware.
- The programs that run on a computer are referred to as software.

1.2.2 Computer Organization

- Computers can be envisioned as divided into six logical units or sections:
  - Input unit. This "receiving" section obtains from input devices and places it at the disposal of the other units so that it can be processed.
  - Output unit. This "shipping" section takes information that the computer has processed and places it on various output devices to make it available for use outside the computer.
  - Memory unit. This rapid-access, relatively low-capacity "warehouse" section retains information that has been entered through the input unit, making it immediately available for processing when needed. It also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is volatile. The memory unit is often called either memory or primary memory.

1.2.2 Computer Organization (cont.)

- Arithmetic and logic unit (ALU). This “ manufacturing” section performs calculations. It also contains the computer’s decision mechanisms. In today’s systems, the ALU is usually implemented as part of the next logical unit, the CPU.
- Central processing unit (CPU). This “ administrative” section coordinates and supervises the operation of the other sections.
  - Tells the input unit when information should be read into the memory unit
  - Tells the ALU when information from the memory unit should be used in calculations
  - Tells the output unit when to send information from the memory unit to certain output devices.
  - Many of today’s computers have multiple CPUs and, hence, can perform many operations simultaneously—such computers are called multiprocessors. A multi-core processor implements multiprocessing on a single integrated circuit chip.

1.2.2 Computer Organization (cont.)

- Secondary storage unit. This is the long-term, high-capacity “warehousing” section. Programs or data not actively being used by the other units normally are placed on secondary storage devices (e.g., your hard drive) until they’re again needed, possibly hours, days, months or even years later. Therefore, information on secondary storage devices is said to be persistent.

1.4 Machine Languages, Assembly Languages and High-Level Languages

- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate *translation steps*.
- Programming languages may be divided into three general types:
  - Machine languages
  - Assembly languages
  - High-level languages

1.4 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

Machine Languages

- Any computer can directly understand only its own machine language (machine-dependent).
- Consist of strings of numbers (ultimately reduced to 1s and 0s) that instruct computers to perform their most elementary operations one at a time.
- Writing programs in machine language is very hard and error prone

Assembly Languages and Assemblers

- Low level languages where strings of 1s and 0s are replaced by mnemonics: ADD, SUB, etc.
- *Translator programs* called Assemblers convert early assembly-language programs to machine language.

1.4 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

High-Level Languages and Compilers

- High-level languages allow to write instructions that look almost like everyday English and contain commonly used mathematical notations: Single statements accomplish substantial tasks.
- *Translator programs* called Compilers convert high-level language programs into machine language.
- Writing programs in high-level language is faster and easier to maintain than in assembly language

Programming Paradigms

- Programming paradigm is a way of programming
- Approaches to problem-solving
- Java supports Procedural programming, Object-Oriented Programming (OOP), and other programming paradigms
- Procedural (Structured or Imperative) programming:
  - Use of high-level control structures: decisions, sequences, and loops
  - Rely on procedures: a procedure is a set of ordered actions
  - Divide the problem into small procedures
  - A single procedure provide a solution for a single problem
  - Ignore data design and not easy to be reused
- Object-Oriented Programming:
  - Rely on objects: contains data and subroutines
  - Reusable software components
  - Easier to understand, correct and modify.

1.6 Operating Systems

- Software systems that make using computers more convenient.
- Provide services that allow each application to execute safely, efficiently and *concurrently* (i.e., in parallel) with other applications.
- The software that contains the core components of the operating system is called the kernel.
- Popular desktop operating systems include Linux, Windows 7 and Mac OS X.
- Popular mobile operating systems used in smartphones and tablets include Google's Android, Apple's iOS (for its iPhone, iPad and iPod Touch devices), Windows Phone and Blackberry OS.

1.7 Programming Languages

Programming language	Description
Fortran	Fortran (FORmula TRANslator) was developed by IBM Corporation in the mid-1950s for scientific and engineering applications that require complex mathematical computations. It's still widely used, and its latest versions support object-oriented programming.
COBOL	COBOL (COmmon Business Oriented Language) was developed in the late 1950s by computer manufacturers, the U.S. government and industrial computer users based on a language developed by Grace Hopper, a U.S. Navy Rear Admiral and computer scientist who also advocated for the international standardization of programming languages. COBOL is still widely used for commercial applications that require precise and efficient manipulation of large amounts of data. Its latest version supports object-oriented programming.
Pascal	Research in the 1960s resulted in *structured programming*—a disciplined approach to writing programs that are clearer, easier to test and debug and easier to modify than large programs produced with previous techniques. One result of this research was the development in 1971 of the Pascal programming language, which was designed for teaching structured programming and was popular in college courses for several decades.

Fig. 1.5 | Some other programming languages. (Part 1 of 4.)

Programming language	Description
Ada	Ada, based on Pascal, was developed under the sponsorship of the U.S. Department of Defense (DOD) during the 1970s and early 1980s. The DOD wanted a single language that would fill most of its needs. The Ada language was named after Lady Ada Lovelace, daughter of the poet Lord Byron. She's credited with writing the world's first computer program in the early 1800s (for the Analytical Engine mechanical computing device designed by Charles Babbage). Ada also supports object-oriented programming.
Basic	Basic was developed in the 1960s at Dartmouth College to familiarize novices with programming techniques. Many of its latest versions are object oriented.
C	C was developed in the early 1970s by Dennis Ritchie at Bell Laboratories. It initially became widely known as the UNIX operating system's development language. Today, most of the code for general-purpose operating systems is written in C or C++.
C++	C++, which is based on C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides several features that "spruce up" the C language, but more important, it provides capabilities for object-oriented programming.

Fig. 1.5 | Some other programming languages. (Part 2 of 4.)

Programming language	Description
Objective-C	Objective-C is another object-oriented language based on C. It was developed in the early 1980s and later acquired by NeXT, which in turn was acquired by Apple. It has become the key programming language for the OS X operating system and all iOS-powered devices (such as iPods, iPhones and iPads).
Visual Basic	Microsoft's Visual Basic language was introduced in the early 1990s to simplify the development of Microsoft Windows applications. Its latest versions support object-oriented programming.
Visual C#	Microsoft's three object-oriented primary programming languages are Visual Basic (based on the original Basic), Visual C++ (based on C++) and Visual C# (based on C++ and Java, and developed for integrating the Internet and the web into computer applications).
PHP	PHP, an object-oriented, open-source scripting language supported by a community of users and developers, is used by millions of websites. PHP is platform independent—implementations exist for all major UNIX, Linux, Mac and Windows operating systems. PHP also supports many databases, including the popular open-source MySQL.

Fig. 1.5 | Some other programming languages. (Part 3 of 4.)

Programming language	Description
Perl	Perl (Practical Extraction and Report Language), one of the most widely used object-oriented scripting languages for web programming, was developed in 1987 by Larry Wall. It features rich text-processing capabilities.
Python	Python, another object-oriented scripting language, was released publicly in 1991. Developed by Guido van Rossum of the National Research Institute for Mathematics and Computer Science in Amsterdam (CWI), Python draws heavily from Modula-3—a systems programming language. Python is "extensible"—it can be extended through classes and programming interfaces.
JavaScript	JavaScript is the most widely used scripting language. It's primarily used to add dynamic behavior to web pages—for example, animations and improved interactivity with the user. It's provided with all major web browsers.
Ruby on Rails	Ruby, created in the mid-1990s, is an open-source, object-oriented programming language with a simple syntax that's similar to Python. Ruby on Rails combines the scripting language Ruby with the Rails web application framework developed by 37Signals. Their book, *Getting Real* (`gettingreal.37signals.com/toc.php`), is a must read for web developers. Many Ruby on Rails developers have reported productivity gains over other languages when developing database-intensive web applications.

Fig. 1.5 | Some other programming languages. (Part 4 of 4.)

1.8 Java

- Microprocessors have had a profound impact in intelligent consumer-electronic devices.
- 1991
  - Recognizing this, Sun Microsystems funded an internal corporate research project led by James Gosling, which resulted in a C++-based object-oriented programming language that Sun called Java.
  - Key goal of Java is to be able to write programs that will run on a great variety of computer systems and computer-controlled devices.
  - This is sometimes called "write once, run anywhere."

1.8 Java (Cont.)

- 1993
  - The web exploded in popularity
  - Sun saw the potential of using Java to add *dynamic content* to web pages.
- Java drew the attention of the business community because of the phenomenal interest in the web.
- Java is used to develop large-scale enterprise applications, to enhance the functionality of web servers, to provide applications for consumer devices and for many other purposes.

1.8 Java (Cont.)

Java Class Libraries

- Rich collections of existing classes and methods
- Also known as the Java APIs (Application Programming Interfaces).

1.9 A Typical Java Development Environment

- Normally there are five phases
  - edit
  - compile
  - load
  - verify
  - execute.

1.9 A Typical Java Development Environment (Cont.)

- Phase 1 consists of editing a file with an *editor program*
  - Using the editor, you type a Java program (source code).
  - Make any necessary corrections.
  - Save the program.
  - Java source code files are given a name ending with the .java extension indicating that the file contains Java source code.

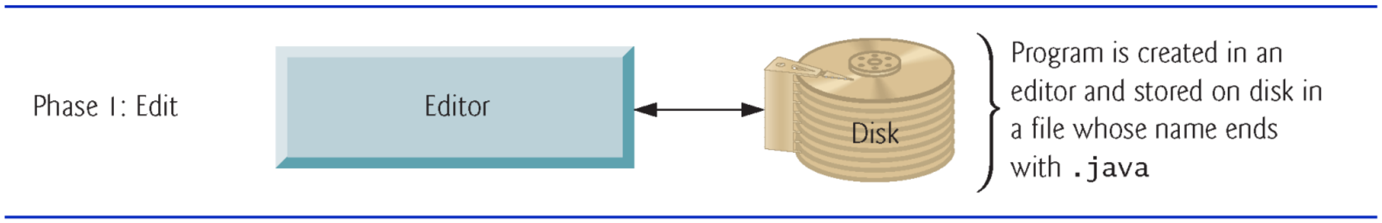
Phase 1: Edit
Editor
Disk
Program is created in an editor and stored on disk in a file whose name ends with .java

Fig. 1.6 | Typical Java development environment—editing phase.

1.9 A Typical Java Development Environment (Cont.)

- Linux editors: `vi` and `emacs`.
- Windows provides Notepad.
- OSX provides TextEdit.
- Many freeware and shareware editors available online:
  - Notepad++ (`notepad-plus-plus.org`)
  - EditPlus (`www.editplus.com`)
  - TextPad (`www.textpad.com`)
  - jEdit (`www.jedit.org`).
- Integrated development environments (IDEs)
  - Provide tools that support the software development process, such as editors, debuggers for locating logic errors (errors that cause programs to execute incorrectly) and more.

1.9 Java and a Typical Java Development Environment (Cont.)

- Popular Java IDEs
  - Eclipse (`www.eclipse.org`)
  - NetBeans (`www.netbeans.org`)
  - IntelliJ IDEA (`www.jetbrains.com`)
- On the book's website at `www.deitel.com/books/jhtp10`
  - Dive-Into® videos that show you how to execute this book's Java applications and how to develop new Java applications with Eclipse, NetBeans and IntelliJ IDEA.

1.9 A Typical Java Development Environment (Cont.)

- Phase 2: Compiling a Java Program into Bytecodes
  - Use the command javac (the Java compiler) to compile a program. For example, to compile a program called `Welcome.java`, you'd type
    - `javac Welcome.java`
  - If the program compiles, the compiler produces a .class file called `Welcome.class` that contains the compiled version.

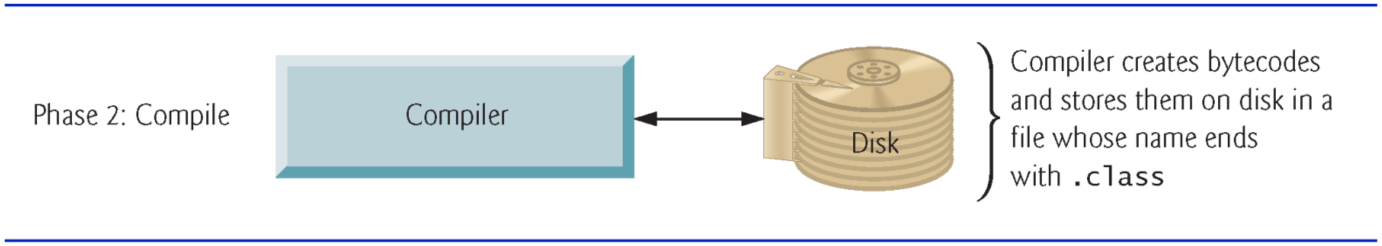
Phase 2: Compile
Compiler
Disk
Compiler creates bytecodes and stores them on disk in a file whose name ends with .class

Fig. 1.7 | Typical Java development environment—compilation phase.

1.9 A Typical Java Development Environment (Cont.)

- Java compiler translates Java source code into bytecodes that represent the tasks to execute.
- Bytecode instructions are *platform independent*
- The Java Virtual Machine (JVM)—a part of the JDK and the foundation of the Java platform—executes bytecodes.
- Virtual machine (VM)—a software application that simulates a computer
  - Hides the underlying operating system and hardware from the programs that interact with it.
- Bytecodes are portable – If the same VM is implemented on many computer platforms, applications written for that type of VM can be used on all those platforms.
- The JVM is invoked by the java command: `java Welcome`

1.9 A Typical Java Development Environment (Cont.)

- Phase 3: Loading a Program into Memory
  - The JVM places the program in memory to execute it—this is known as loading.
  - Class loader takes the `.class` files containing the program's bytecodes and transfers them to primary memory.
  - Also loads any of the `.class` files provided by Java that your program uses.

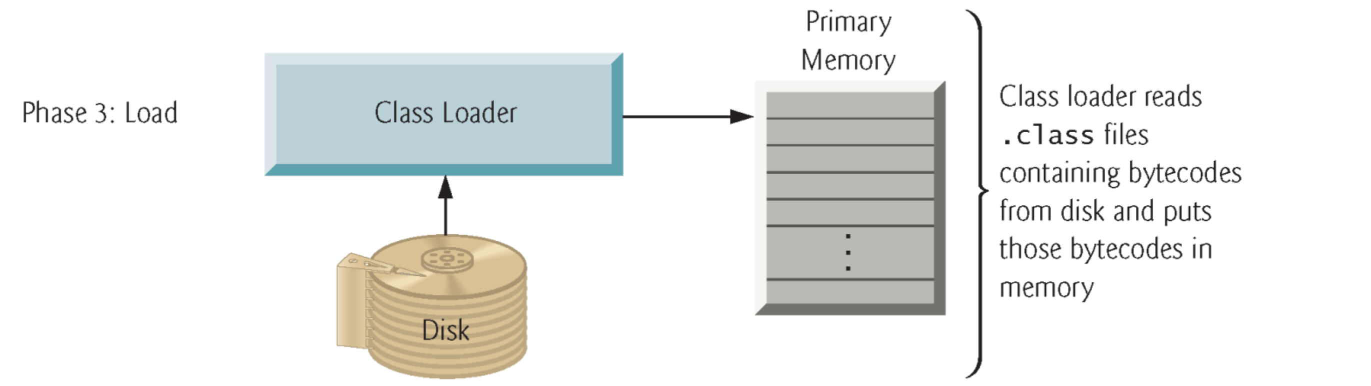
Phase 3: Load
Class Loader
Disk
Primary Memory
Class loader reads .class files containing bytecodes from disk and puts those bytecodes in memory

Fig. 1.8 | Typical Java development environment—loading phase.

1.9 A Typical Java Development Environment (Cont.)

- Phase 4: Bytecode Verification
  - As the classes are loaded, the bytecode verifier examines their bytecodes
  - Ensures that they're valid and do not violate Java's security restrictions.

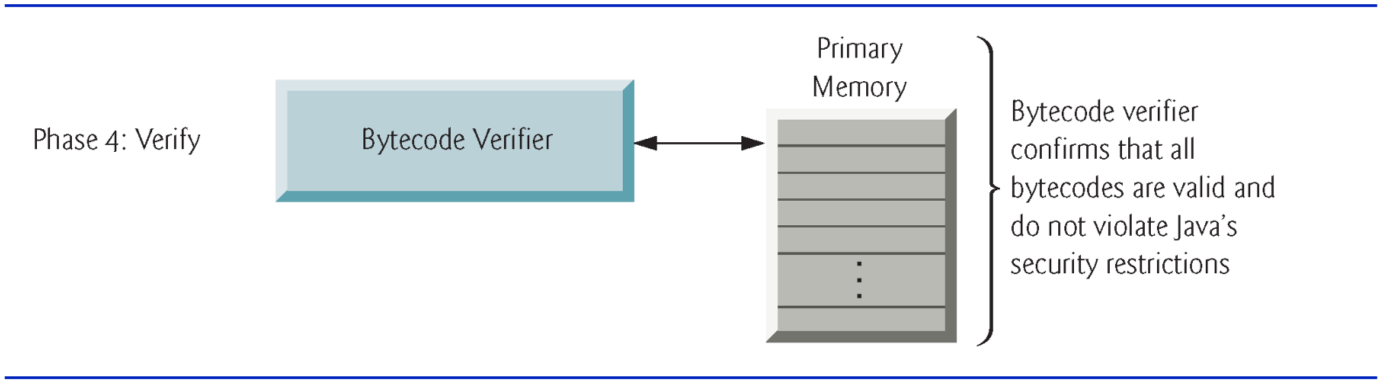
Phase 4: Verify
Bytecode Verifier
Primary Memory
Bytecode verifier confirms that all bytecodes are valid and do not violate Java's security restrictions

Fig. 1.9 | Typical Java development environment—verification phase.

1.9 A Typical Java Development Environment (Cont.)

- Phase 5: Execution
  - The JVM executes the program's bytecodes.
  - JVMs typically execute bytecodes using a combination of interpretation and so-called just-in-time (JIT) compilation.
  - Analyzes the bytecodes as they're interpreted
  - A just-in-time (JIT) compiler—such as Oracle's Java HotSpot™ compiler—translates the bytecodes into the underlying computer's machine language.

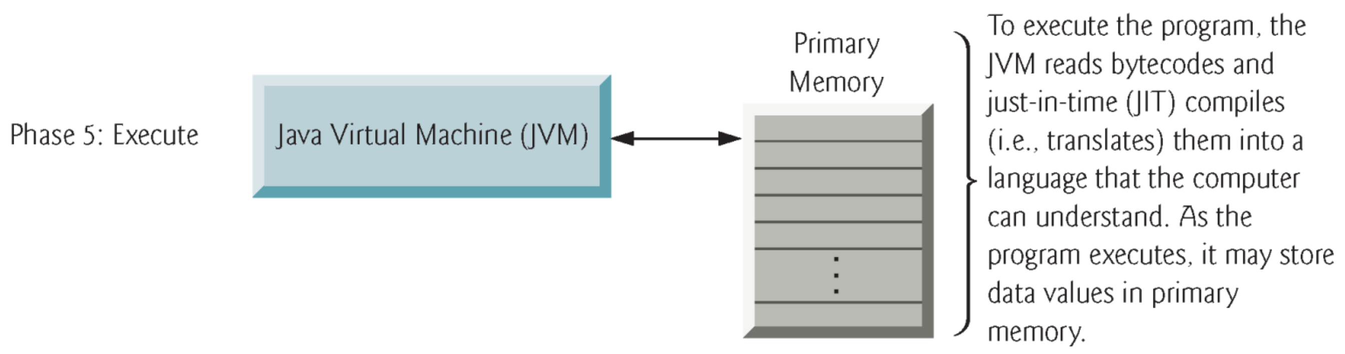
Phase 5: Execute
Java Virtual Machine (JVM)
Primary Memory
To execute the program, the JVM reads bytecodes and just-in-time (JIT) compiles (i.e., translates) them into a language that the computer can understand. As the program executes, it may store data values in primary memory.

Fig. 1.10 | Typical Java development environment—execution phase.

1.9 A Typical Java Development Environment (Cont.)

- When the JVM encounters these compiled parts again, the faster machine-language code executes.
- Java programs go through *two* compilation phases
- One in which source code is translated into bytecodes (for portability across JVMs on different computer platforms) and
- A second in which, during execution, the bytecodes are translated into *machine language* for the actual computer on which the program executes.