现代计算机教育系列教材（英文版）——国外著名大学教授鼎力之作
丛书主编 金兰

Java Programming

Java 程序设计

Danny Poo
（新加坡）潘祥春 编著

清华大学出版社
Tsinghua University Press

现代计算机教育系列教材（英文版）——国外著名大学教授鼎
丛书主编 金兰

Java Programming

Java 程序设计

Danny Poo
（新加坡）潘祥君 编著

清华大学出版社
Tsinghua University Press

内 容 简 介

本书使用 Java 语言讲解面向对象的程序开发方法。全书内容共 15 章，内容包括：Java 编程环境；Java 组成；表达式、语句和运算符；程序流程控制机制；数组；方法；类和对象；Java 应用程序界面（API）；输入和输出；文件控制；单类继承；封装；多态性；抽象；排序、搜索和递归。本书内容深入浅出，循序渐进，对编程过程的每一步均给出详细的指导，每个范例均提供完整的源代码，非常适合于没有编程知识的初学者学习 Java 语言编程方法。

本书可作为高等院校相关专业本科生 Java 语言程序设计课程教材，也可作为软件开发设计人员学习 Java 面向对象编程方法的自学参考书。

图书在版编目（CIP）数据

Java 程序设计 = Java Programming：英文/（新加坡）潘祥春（Danny Poo）编著. —北京：清华大学出版社，2010.1
（现代计算机教育系列教材（英文版）——国外著名大学教授鼎力之作/金兰主编）
ISBN 978-7-302-20722-1

Ⅰ. J…　Ⅱ. 潘…　Ⅲ. JAVA 语言－程序设计－教材－英文　Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2009）第 146243 号

责任编辑：战晓雷　薛　阳
责任校对：时翠兰
责任印制：杨　艳

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京市清华园胶印厂
经　　销：全国新华书店
开　　本：185×230　　**印　张**：19.5　　**字　数**：409 千字
版　　次：2010 年 1 月第 1 版　　**印　次**：2010 年 1 月第 1 次印刷
印　　数：1～3000
定　　价：35.00 元

Contents

CHAPTER 1 The Java Programming Environment

CHAPTER 2 The Java Language Components

CHAPTER 3 Expressions, Statements and Operators

CHAPTER 4 Program Flow Controls

CHAPTER 5 Arrays

CHAPTER 6 Methods

CHAPTER 7 Class and Objects

CHAPTER 8 The Java Application Programming Interface (API)

CHAPTER *9* Inputs and Outputs

CHAPTER *10* File Handling

CHAPTER *11* Inheritance

CHAPTER *12* Encapsulation

CHAPTER *13* Polymorphism

CHAPTER *14* Interface

CHAPTER *15* Sorting, Searching, and Recursion

CHAPTER 1

The Java Programming Environment

Java is a programming language[①]. Much like English is a spoken language that we use to communicate with one another, Java is used for communicating our intentions as a programmer to a computer.

A computer is a device that can be configured to understand the instructions written in Java. The set of instructions is commonly known as a *program*. So, when we speak of Java programming, we are involved in writing programs in the Java language that a computer can understand and execute according to what the set of instructions dictates.

The Java programming language[②] is textual and is expressed using English. It has a set of restricted vocabulary that we can use to express our intentions in a program.

Can a computer understand English and how does it know how to execute the program? A computer technically does not understand English as it is. But, it does understand code specially crafted for machines. Such code is typically represented in 0s and 1s which we human being finds it extremely difficult to comprehend.

How do we then bridge the gap between a high-level English-written program and a set of low-level binary-code-based machine instructions? The solution lies in having an interpreter. An interpreter is a software program that takes in a high-level program and translates it into a low-level binary-code-based program that a computer can understand. In the case of Java, the output of the interpretation process is a set of *byte code*.

Byte code, though low-level, is still not at the lowest level. Machine code, or *object*

---

① There are other programming languages available e. g. C, C++, C#, FORTRAN, COBOL, MUMPS, SNOBOL, Pascal, Python, etc.

② Java bears some similarities to the C programming language, as Java was created from C.

code, would be considered as the lowest level of code for a computer. Byte code is not the same as object code because byte code does not conform to the operating instruction set of the target machine. The purpose of generating byte code instead of machine code is to allow for a layer with which a virtual machine (another software program) can take control to interpret the byte code to object code. This virtual machine is provided in the Java programming platform as the Java Virtual Machine (JVM). By "Virtual Machine" we mean that JVM is not a "hard" machine but one that simulates a "hard" machine using software. Thus, byte code is executed by a software (the JVM) whose output is an object code executable by a computer. The same byte code is therefore executable on any computer machines so long as an appropriate JVM is available.

JVM is operating-system-platform dependent. There are thus as many JVM types as there are operating system types. Hence, there is a JVM type for the Microsoft Windows, Linux, UNIX, Solaris, Apple Macintosh, etc. Through this approach, Java has been known as programming language for producing programs that is "*Write Once, Run Anywhere*".

1.1 History of Java

Java was developed by Sun Microsystems in 1995. It was created in response to the need for a programming language that can be operated on multiple operating system platforms. The desire is to have a program written for the Microsoft Windows operating system to be executable also in an Apple Macintosh, UNIX, Linux or Solaris operating system, for example. Sun Microsystems' answer to this requirement is to generate intermediate byte code that is not tied to any operating system architecture. Byte codes are executable by JVMs. Therefore, a Java program written for and executable on a Microsoft Windows operating system needs not be re-compiled for execution on a Linux platform, neither does it need to be re-compiled for one on a UNIX system, so long as an appropriate JVM is available.

1.2 Preparing to Write Java Programs

Before we start to write our first Java program, we need to prepare the environment for programming in the computer. Three items are essential:

1. The Java SE (Standard Edition) Development Kit[①]. The Java SE Development Kit (JDK) includes the Java Runtime Environment (JRE) and command-line development tools that are useful for developing applets and applications.

2. The Java SE 6 Documentation.

3. The JCreator IDE (Interactive Development Environment) Light version.

Refer to Workshop 1.1[②] for instructions on how to download and install the above items.

1.3 A Simple Java Program

How does a Java program look like? Code 1.1 shows a simple Java program. When this program is run, the line "How do you do?" is printed on the Windows DOS prompt[③].

Code 1.1: A Simple Java Program

```
class HowDoYouDo {

  public static void main (String args[]) {
    System.out.println("How do you do?");
  }
}
```

The statement that actually prints the line is: System.out.println("How do you do?");.

The System.out.println statement is enclosed within a procedural block called a *method* in Java. In general, a method is a grouping of data variables and statements. The method referred here is the main() method. When a method is called, all the statements in the method will be executed. The execution is carried out statement by statement until the end of the method is reached. The beginning and ending of a method is denoted by an opening brace ("{") and a closing brace ("}") respectively.

The main() method is enclosed within a *class* HowDoYouDo. A class is defined as a grouping of data variables and methods and has a general structure as shown in Code 1.2.

---

① The current version is JDK 6u6.

② All workshops are included at the end of each chapter.

③ Our discussion will be restricted to the use of Microsoft Windows version of Java. We will be running all Java programs in the Windows DOS prompt.

Code 1.2: A General Class Structure

```
class <classname> {

  public static void main(String argv[]) {
    // put your statements here
  }
}
```

<classname> refers to the name of the class. Replace this with a class name of your choice. Typically, a class name should reflect the purpose of the class. One convention for naming a class is to use upper case for the first letter of the class name. For example, to name the HowDoYouDo class, use HowDoYouDo instead of howDoYouDo.

The keywords "public", "static", and "void" have their significance. "public" indicates the method main( ) can be called by another program outside of this class. "static" indicates that the method main( ) belongs to a class as opposed to belonging to an object. "void" indicates that the method main( ) does not return any value after it has completed its execution. These may not make much sense to you now but do not worry, we will discuss them further later in the book. For now, just make sure that these three keywords are present in front of the method name. The sequence in which they appear is not important.

String argv[ ] is a String array parameter definition. It is through this array that inputs entered via the Windows DOS prompt are captured and conveyed to the program. You may alternatively use String args[ ] in place of String argv[ ]. No other names for the String array are allowed.

The statement "// put your statements here" is a comment. We use comment to explain to other fellow programmers our intention. A comment is not executable. Java will ignore it during program execution.

1.4 How to Run a Java Program?

The best way to explain how to run a Java program is for you to try it out yourself. Refer to Workshop 1.2 for instructions on how to run a Java program.

1.5 Commonly Encountered Problems

Programmers new to Java programming need to be aware of the following characteristics of Java to avoid errors in coding:

1. Java is case-sensitive. A class named HowDoYouDo. java is different from one named howdoyoudo. java. A variable named Total is different from total. Also, the primitive type int (to be discussed in Chapter 2) when spelt as Int is not recognized as such.

2. javac acts on a file of ". java" extension and not on a ". class" extension. The ". java" extension must be included when javac is called, e. g. D:\java > javac HowDoYouDo. java.

3. java acts on a file of ". class" extension. However, execution of a ". class" file does not require the ". class" extension, e. g. D:\java > java HowDoYouDo.

4. Only classes with the main( ) method are executable e. g. HowDoYouDo class.

5. Ensure the current directory is properly set in the classpath variable. Start > Settings > Control Panel > System > Advanced > Environment Variables > System Variables > Classpath > Variable Value. Check the Variable Value of Classpath. If the current directory denoted by "." (i. e. period) is not included in the Classpath Variable Value, add it in.

Workshops

Workshop 1.1: Preparing the Environment for Java Programming

We need to download:

1. The Java SE (Standard Edition) Development Kit[①]. The Java SE Development Kit (JDK) includes the Java Runtime Environment (JRE) and command-line development tools that are useful for developing applets and applications.

2. The Java SE 6 Documentation.

3. The JCreator IDE (Interactive Development Environment) Light version.

Go to the JCreator download website: http://www. jcreator. com/download. htm(See Figure

① The current version is JDK 6u6.