

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）潘祥春



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。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

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. TP312.14

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

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话:010-62770177 转 3103 产品编号:033347-01

C ontents

CHAPTER 1 The Java Programming Environment

- 1.1 History of Java 2
- 1.2 Preparing to Write Java Programs 2
- 1.3 A Simple Java Program 3
- 1.4 How to Run a Java Program? 4
- 1.5 Commonly Encountered Problems 5
- Workshops 5
 - Workshop 1.1: Preparing the Environment for Java Programming 5
 - Workshop 1.2: How to Run a Java Program? 10
 - Workshop 1.3: How to Compile and Run a Java Program in DOS Prompt? 12
- Exercises 13

CHAPTER 2 The Java Language Components

- 2.1 Print2Numbers Java Program 17
- 2.2 The Java Vocabulary and Character Sets 19
- 2.3 Primitive Data Types 19
 - 2.3.1 Boolean 20
 - 2.3.2 Characters 20
 - 2.3.3 Integers 21

2.3.4	Floating Point	22
2.3.5	Object References	22
2.3.6	String	22
2.4	Identifiers	22
2.5	Reserved Words	23
2.6	Comments	24
2.7	Basic Program Structure	25
Workshops		26
Workshop 2.1	: Understanding the Sequence of Program Execution	26
Exercises		28

CHAPTER 3 Expressions, Statements and Operators

3.1	Expression Statements	31
3.1.1	Types of Expressions	31
3.1.2	Assignment Expression Statements	31
3.1.3	Prefix or Postfix Forms of “++” and “--” Statements	32
3.1.4	Method Call Statements	32
3.1.5	Object Creation Statements	32
3.2	Declaration Statements	33
3.3	Operators	33
3.3.1	Arithmetic Operators	34
3.3.2	Auto-Increment and Auto-Decrement Operators	35
3.3.3	Logical Operators	35
3.3.4	Relational Operators	37
3.3.5	Bitwise Operators	39
3.3.6	The Conditional Operator “?:”	40
3.3.7	Assignment Operators	41
3.3.8	“+” Operator	43
3.3.9	“.” Operator	44
3.3.10	Precedence and Associativity	44
Workshops		47
Workshop 3.1	: Entering Data for Program Execution	47
Exercises		53

CHAPTER 4 Program Flow Controls

4.1	Sequence	56
4.2	Selection	57
4.2.1	Block	57
4.2.2	Types of Selection Statements	59
4.3	Iteration	64
4.3.1	The while Statement	64
4.3.2	The do-while Statement	66
4.3.3	The for Statement	67
4.3.4	The Enhanced 'for' Statement	70
4.4	Labels	70
4.5	The break Statement	71
4.6	The continue Statement	71
	Exercises	71

CHAPTER 5 Arrays

5.1	Array	75
5.1.1	Declaring and Creating an Array	75
5.1.2	Initializing an Array	77
5.1.3	Using Arrays	77
5.2	Two-dimensional Arrays	79
5.2.1	One-dimensional Array Approach	79
5.2.2	Two-dimensional Array Approach	80
5.2.3	Populating Two-dimensional Arrays	82
5.3	Applying the Enhanced 'for' Statement in Arrays	83
5.4	An Application; Printing Numbers Divisible by 3	84
5.4.1	Using Label and break Statement	85
5.4.2	Using continue Statement	88
	Workshops	90
Workshop 5.1; Copying Arrays		90
	Exercises	96

CHAPTER 6 Methods

6.1	Defining a Problem	98
6.2	A Problem Solving Approach	99
6.3	Improving the Problem-Solving Approach	103
6.3.1	Advantage of Using Methods	107
6.3.2	Walking Through readInputValues() Method	107
6.3.3	Walking Through convertMarksToGrades() Method	107
6.3.4	Walking Through printDetails() Method	107
6.4	Block Structure and Scope	108
6.4.1	Local Variables	108
6.4.2	Global Variables	109
6.4.3	Determining Scope of Variables across Methods	110
6.4.4	Distinguishing Local Variables from Global Variables	111
6.4.5	Scope of Identifier Declaration	112
6.5	Parameters	113
6.5.1	Actual and Formal Parameters	113
6.5.2	Value Parameters	117
6.6	Methods that Return Values	119
6.6.1	Returning Values	119
6.6.2	The return Statement	121
	Workshops	121
	Workshop 6.1: Using Methods	121
	Exercises	125

CHAPTER 7 Class and Objects

7.1	Class and Objects	128
7.2	Constructing Objects	128
7.2.1	Constructors	129
7.2.2	Multiple Constructor Method Definition	131
7.2.3	Constructor Method Invocation	133
7.3	Instance and Class Variables	134

7.4	Instance and Class Methods	135
7.4.1	Instance Methods	135
7.4.2	Class Methods	136
7.5	Constants	138
7.6	The this Keyword	139
7.6.1	Using this Keyword in Instance Method	140
7.6.2	Using this Keyword in Constructor	141
7.7	Inner Class	141
7.7.1	Compiling an Inner Class	142
7.7.2	Static Inner Class	143
7.7.3	Creating Inner Class Objects	143
7.8	Class Hierarchy	143
7.8.1	Superclass and Subclass	144
7.8.2	Inheritance	146
	Workshops	148
	Workshop 7.1: Implementing Class and Objects	148
	Exercises	153

CHAPTER 8 The Java Application Programming Interface (API)

8.1	Java Package	157
8.1.1	The 'package' Keyword	158
8.1.2	The 'import' Keyword	159
8.1.3	File Name of a Public Class	161
8.2	The Java™ Platform Standard Edition	162
8.3	The Java API	162
8.3.1	The Java API Documentation	163
8.3.2	The Java API Packages	164
8.3.3	Directory Structure of Java API Packages	167
8.3.4	The java.lang, java.io, and java.util Packages	168
8.3.5	Reading the Java API Documentation	168
8.3.6	Using the Java API	169
8.4	The Ubiquitous System.out.println() Method	170
8.4.1	The System Class	170

8.4.2	The PrintStream Class	171
8.5	String Tokenizers	173
8.5.1	The java.util.StringTokenizer Class	173
8.5.2	Delimiter Characters	175
	Exercises	176

CHAPTER 9 Inputs and Outputs

9.1	Input and Output Streams	178
9.1.1	Screen Outputs	178
9.1.2	Keyboard Inputs	179
9.1.3	Reading and Displaying Texts	180
9.2	Exception Handling	181
9.2.1	Java Exception Handling	181
9.2.2	Explicit Exception Handling	182
9.3	The Scanner Class	185
9.3.1	Creating a Scanner Object	186
9.3.2	Handling Numerical Data Types	186
9.3.3	Handling String Values	186
9.3.4	Handling Boolean Values	186
9.3.5	Exceptions and Delimiters	187
9.3.6	A Scanner Class Application	188
	Exercises	190

CHAPTER 10 File Handling

10.1	Text Files	193
10.1.1	Writing to a Text File	193
10.1.2	Appending Texts to a File	195
10.1.3	Reading from a File	195
10.2	Binary Files	196
	Exercises	198

CHAPTER 11 Inheritance

11.1	The Inheritance Mechanism	199
11.1.1	Subclass and Superclass	199
11.1.2	java.lang.Object Class	200
11.1.3	Downward Property Propagation	200
11.2	Demonstrating Inheritance	200
11.3	The super Keyword	202
11.3.1	Syntax	202
11.3.2	Constructor Chaining	203
11.3.3	Calling Superclass Methods	205
11.4	Method Overriding	205
	Exercises	207

CHAPTER 12 Encapsulation

12.1	Access Modifiers: public, protected, private	216
12.1.1	Using Access Modifiers	216
12.1.2	Accessibility Effects	217
12.2	Data Field Encapsulation	218
12.3	Class Abstraction	219
12.4	Class Encapsulation	220
12.4.1	Encapsulating a Class	220
12.4.2	Enhanced Maintainability	222
12.4.3	Bundling and Information Hiding	225
	Exercises	226

CHAPTER 13 Polymorphism

13.1	Illustrating Polymorphism with Geometric Shapes	229
13.1.1	The Triangle Class	230
13.1.2	The Rectangle Class	231
13.1.3	The GeometricShape Class	233

13.1.4	The User Class; GeometricShapeMain Class	234
13.2	Abstract Class	235
13.3	Dynamic Binding	236
	Exercises	237

CHAPTER 14 Interface

14.1	The Interface Construct	239
14.2	Interface Definition	239
14.2.1	Interface Declaration and Interface Body	240
14.2.2	Compilation of Interface	241
14.2.3	Implementing Interface	241
14.3	Understanding the Use of Interface	243
14.4	What and How in the Use of Interface	244
14.5	Application of Interface	245
14.5.1	Sales Person Application	245
14.5.2	SalesPerson and Employee Class	247
14.5.3	Sort by Age; The main() Method 1	248
14.5.4	Sort by Name; The main() Method 2	250
14.5.5	Sort by Wage; The main() Method 3	251
14.5.6	The Output	251
14.6	The Serializable Interface	252
14.7	Interface and Abstract Class	257
14.8	Changes in Interface	257
14.9	Uses of Interface	258
	Exercises	258

CHAPTER 15 Sorting, Searching, and Recursion

15.1	Sorting	264
15.1.1	Selection Sort	264
15.1.2	Bubble Sort	267
15.1.3	A Sorting Application	269

15.2	Searching	273
15.2.1	Linear Search	273
15.2.2	Binary Search	274
15.2.3	A Searching Application	276
15.3	Recursion	280
15.3.1	Recursive Method	281
15.3.2	Writing a Recursive Method	284
15.3.3	The Ubiquitous Factorial	284
15.3.4	Applying Recursion	285

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.