全国计算机等级考试二级教程

——Java语言程序设计

(2013年版)

NCRE Examination 全国计算机等级考试

教育部考试中心

高等教育出版社
HIGHER EDUCATION PRESS

全国计算机等级考试二级教程

——Java 语言程序设计

（2013 年版）

Quanguo Jisuanji Dengji Kaoshi Erji Jiaocheng

——Java Yuyan Chengxu Sheji

教育部考试中心

主　编　柳西玲

参　编　许　斌　郎　波　金铁鹰

高等教育出版社·北京

HIGHER EDUCATION PRESS　BEIJING

内容提要

由教育部考试中心推出的计算机等级考试是一种客观、公正、科学的专门测试计算机应用人员的计算机知识与技能的全国性考试，它面向社会，服务于社会。

本书在教育部考试中心组织下，在全国计算机等级考试委员会指导下，由有关专家执笔编写而成。本书按照《全国计算机等级考试二级 Java 语言程序设计考试大纲（2013 年版）》的要求编写，内容包括：Java 语言概论，基本数据类型，运算符与表达式，流程控制，Java 的继承、多态、高级类特性和数组，异常和断言，输入输出及文件操作，线程，编写图形用户界面，Applet 程序设计，集合与泛型，Java 编程风格，应用开发工具与安装使用，等等。

本书是教育部考试中心指定教材，是考生参加全国计算机等级考试二级 Java 语言程序设计的必备参考书，也可作为学习 Java 编程的参考书。

图书在版编目（CIP）数据

全国计算机等级考试二级教程：2013 年版. Java 语言程序设计/教育部考试中心编. --北京：高等教育出版社，2013.5

ISBN 978-7-04-037228-1

Ⅰ.①全… Ⅱ.①教… Ⅲ.①电子计算机-水平考试-教材②JAVA 语言-程序设计-水平考试-教材 Ⅳ.①TP3

中国版本图书馆 CIP 数据核字（2013）第 084138 号

策划编辑 何新权　　责任编辑 柳秀丽　　封面设计 杨立新　　版式设计 杜微言

责任校对 胡晓琪　　责任印制 毛斯璐

出版发行 高等教育出版社
社　　址 北京市西城区德外大街 4 号
邮政编码 100120
印　　刷 国防工业出版社印刷厂
开　　本 787mm×1092mm　1/16
印　　张 20.75
字　　数 520 千字
购书热线 010-58581118

咨询电话 400-810-0598
网　　址 http://www.hep.edu.cn
http://www.hep.com.cn
网上订购 http://www.landraco.com
http://www.landraco.com.cn
版　　次 2013 年 5 月第 1 版
印　　次 2013 年 5 月第 1 次印刷
定　　价 40.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换

版权所有　侵权必究

物 料 号 37228-00

目　录

试读结束：需要全本请在线购买： www.ertongbook.com

第1章 Java语言概论

1995年5月,Sun公司在"Sun World 95"大会上发布了Java,新一代的网络计算机语言由此诞生,1996年发布了开发工具和程序库JDK 1.0,直至1998年发布JDK 1.2,实现了全球信息网络平台。1999年,Sun公司将Java 2平台分为J2SE、J2EE和J2ME三大块,形成了Java技术的基本架构,一直保持至今。这样的基本架构具有很强的开放性,使Java技术能够适应全球不同硬件平台和技术飞速发展所带来的各种变化需求。其中,Java的API(Application Programming Interface)标准成为IT技术的一次重大革命。Java允许三类API:核心API、扩充API、特殊API。2005年,Sun公司将Java 2平台产品改称为Java SE、Java EE和Java ME。从2000年以后,几乎每年对核心Java SE进行改进,至今,已发布了Java SE 6.2版本。Java为Web提供了简便而功能强大的API接口,为Web提供了动态内容的交互页面技术,使Web网页翻开了新的一页,使互联网从通信为主的功能扩展到网络应用系统服务。这种突破性的技术,促使企业的IT系统实现了一次革命性的飞跃,并显示了Java语言的巨大生命力。

1.1 Java语言简介

1.1.1 Java语言的由来

1991年,Sun公司的成员Jame Gosling、Bill Joe等,为电视、控制烤面包机等家用电器开发了一个分布式系统软件Oak(一种橡树名字),它是Java语言的前身。当时,Oak并没有引起人们的注意,直到1994年后期,随着互联网和3W的飞速发展,他们用Java编写了HotJava浏览器,得到Sun公司首席执行官Scott McNealy的支持,并进一步地研制和开发。因为促销和法律的原因,1995年,Oak更名为Java。Java的出现给软件业带来了巨大的冲击,它的独立于平台和安全可靠非常适合网络要求,很快被工业界认可。许多大公司如IBM、Microsoft、DEC等购买了Java的使用权,并被PC Magazine评为1995年十大优秀科技产品。从此,开始了Java应用的新篇章。目前已有15亿台手机内置了Java技术。Java已无处不在,从太空探测、企业电子商务到多层级的网络游戏都已应用Java实现。在现实的人类生活中,将离不开Java技术。

1.1.2 Java语言的目标

创建Java时,其主要目标是:面向对象、简单化、解释型与平台无关、多线程、安全高效、动态性。

1. 面向对象

类思维方法去实现编程,使软件开发人性化、形象化。

2. 简单化

Java 的简单化首先是本身系统的精炼，占内存少，在很普通的计算机上就能运行。其次，它避免了许多其他编程语言的缺点，如取消影响程序代码健壮性的指针运算和编程者对内存管理，使编程很容易入门。

3. 解释型、与平台无关

Java 的虚拟机(Java Virtual Machine，简称 JVM)制定了字节码设计规范，保证了软件体系结构中立，为软件移植建立了良好的基础。与其他解释执行的语言不同，Java 设计的字节码很容易地直接转换成对应于特定 CPU 的机器码，不仅使软件开发周期缩短并且使软件执行时得到较高的性能。

4. 多线程

多线程机制使应用软件能并行执行同步机制，保证了对共享数据的正确操作。编程者分别用不同的线程完成特定的行为，而不需要用全局的事件循环机制，有利于网络上实时交互的生动描述和实现。

5. 安全高效

Java 不允许编程用指针和对内存释放，从根本上避免了非法内存操作。在编译时，对代码进行类型和语法检查；在执行时，校验字节码、代码段格式和规则检查、访问权限和类型转换合法性检查、操作数堆栈的上溢或下溢检查、方法参数合法性检查；在平台安装时，检查配置设定资源域的访问等。

6. 动态性

为适应 Web 应用的快速变化需求，允许程序在运行中下载代码段去动态改变程序的状态。在类库中可自由加入新的方法和实例变量。通过接口支持多重继承，使类的继承更具有扩展性。

1.1.3 Java 语言实现机制

Java 语言为实现其目标，使用了 Java 虚拟机(JVM)、垃圾回收机制和 Java 运行环境(JRE)。

1. JVM

Java 语言的执行模式是编译加解释。编写好的 Java 源程序首先由编译器转换为标准字节代码，然后由 JVM 去解释执行。Java 虚拟机规范对 JVM 的定义为：在计算机中用软件模拟实现的一种虚拟机。JVM 运行的代码存储在. class 文件中，每个文件包含最多一个 public 类的代码。JVM 的代码格式由简洁、高效的字节码构成。JVM 用字节代码程序与各操作系统和硬件分开，保证 Java 程序独立于平台，而 JVM 本身可以用软件或硬件实现。每个 Java 解释器，不管是Java 开发工具还是浏览器，都有一个 JVM 的具体实现来完成下面 3 项任务：

- 加载代码：由类加载器完成。
- 校验代码：由字节码校验器完成。
- 执行代码：由解释器完成。

Java 程序的下载和执行步骤如图 1-1 所示：

(1) 源程序经编译器得字节代码；

(2) 浏览器与服务器连接，要求下载字节码文件；

(3) 服务器将字节代码文件下载到客户机；

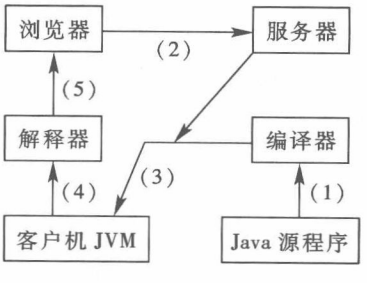


图 1-1 Java 程序的下载和执行

(4) 客户机上的 JVM 执行;

(5) 在浏览器上显示并交互。

2. 垃圾回收机制

在 Java 语言中,所有代码都封装在类中,需要时创建类的实例(对象)来处理,这种动态的实例存储在内存堆中。Java 有一个系统级的线程,对内存的使用进行自动跟踪。该线程能在 JVM 空闲时,对不用的内存进行自动回收。它消除了释放内存的必要,也避免了内存的泄漏,将编程者从琐碎繁忙的内存管理中解脱出来。

3. Java 运行环境(JRE)

任何程序运行都需要一定的软件和硬件环境,这称为平台。Java 语言的运行平台包括 Java 应用程序接口(API)和 JVM,如图 1-2 所示。

<table>
<tr><td colspan="2">Java Application 或 Applet</td></tr>
<tr><td>Java 基本 API</td><td>Java 基本扩展 API</td></tr>
<tr><td>Java 基本类</td><td>Java 标准扩展类</td></tr>
<tr><td colspan="2">JVM</td></tr>
<tr><td colspan="2">移植接口</td></tr>
</table>

图 1-2 Java 运行平台

Java 有 3 种平台:Java SE、Java ME 和 Java EE。它们都立足于核心开发工具包 JDK 的各种版本。本书以 JDK1.6.2 版本为基础,集成开发环境以 NetBeans 6.0 为基础。

1.2 Java 语言面向对象编程

Java 重要的特性之一是它是完全面向对象的编程语言,其核心是用人类解决问题的抽象方法对复杂的客观问题进行分析、组织和解答。对于程序设计而言,其难点当然是寻找尽可能能正确描述问题的抽象。面向对象的编程语言是利用类和对象将问题的数据和操作封装起来,并用标准接口与外界交互,使代表客观世界实体的各种类在程序中能独立和继承。其特性是要求程序具有封装性、继承性、多态性。

1.2.1 面向对象编程的基本概念

学习 Java 语言首先应了解面向对象的一些基本概念:抽象、封装、继承和多态。

1. 抽象

抽象是人类处理复杂事务的提升。**面向对象编程的思路与传统程序设计不同,它强调按照与人类思维方法中的抽象、分类、继承、组合、封装等原则去模拟现实世界的物理存在,**将客观世界事物都抽象定义为各种对象的组合。人们管理抽象的一个有效方法是用层次分类。允许根据物理意义将复杂的系统分解为更多更易处理的部件。这种分层抽象方法也经常被用于程序设计。例如,人们对日常生活中的手机、汽车、电视机等许多设备的使用,都忽略其具体的工作细节,而将它们作为由许多部件组成的一个整体加以利用。面向对象技术也利用这种思路将现实世界的问题抽象成许多对象的类来表示。通过对问题的抽象、分类、封装和组合来更有效地处理问题,也更能清晰地描述客观事物。

面向对象的概念是 Java 的核心。对编程者来讲,重要的是要理解现实系统怎么去抽象转化为软件系统。对任何软件而言,都不可避免地要经历概念抽象的提炼、成长、衰老这样一个生命周期,而面向对象的设计方法,以类为基本单元,封装成包,使软件在生命周期的每一个阶段都有足够的应变能力,去适应千变万化的大千世界。在编程阶段,通过抽象找出各种类,再对各种类

之间的消息进行收集和处理，把问题分解成许多标准接口的构件。当问题有变化时，就能很从容自信地解除或更换现实软件的某些构件代码来适应变化。抽象导致面向对象编程的方法与数据封装，形成不同层次上的构件，使软件构件化，实现了不同层次上的重用。

2. 封装

封装是面向对象技术中隐蔽信息的一种机制。它在程序中将对象的状态和行为封装成为一个完整的、结构高度集中的整体；对外有明确的功能、接口单一通用、可在各种环境下独立运行的单元。其目的是将对象的具体实现细节隐蔽，只通过一个公用接口和消息与其他对象通信。封装使对象的使用者和设计者分开，源代码可独立编写和维护，既保证不受外界干扰，也有利于代码重用。这种封装好的单元是软件标准构件化的基础，可提高软件生产效率、降低成本和缩短开发周期，实现软件工业化。在 Java 语言中，**“类”是程序封装的最小单位。**

3. 继承

继承是描述两个类之间的关系。继承允许一个新类（称为子类）包含另一个已有类（称为父类）的状态和行为。这样，从最一般的类开始，用子类去逐步特殊化，可派生出一系列的子类，使父类和子类的关系层次化，可降低程序的复杂度，并提供了类之间描述共性的方法，减少了类的重复说明。**子类的派生过程称为类的继承。继承是抽象分层管理的实现机制。**

在面向对象的继承概念中，有单继承和多继承两种。单继承是指任何子类都只能从一个父类派生，而多继承是指一个子类可由多个父类派生。因此，单继承的类层次是树状结构，多重继承的类层次是网状结构。在 Java 中，除 Object 类外，任何类都有父类。继承仅支持单继承，多重继承要通过接口实现，这大大降低了继承的复杂度，给编程带来方便。

Java 中，创建了一个类后可用关键字 extends 去创建它的子类。

4. 多态

多态是允许一个类中有多个同名方法，但方法的具体实现却不同的机制。为提高程序的抽象和简洁，许多对象的操作方法名相同，但方法的具体实现细节却不同。如一种排序方法，可以用在不同的数据类型上，对数值类型和字符类型，它们的排序程序就不相同。这种在同一个程序中同名的方法可用不同代码实现的特性称为多态性。可利用子类对父类的方法覆盖来具体实现。多态可以使类的对象响应同名消息（方法）去完成某种特定功能，而具体实现代码却不相同。例如，同样是复制和粘贴操作，在文本处理和图形处理中具体代码是完全不同的。

1.2.2 类与包

类是对具有相同特性对象的封装组合，是 Java 程序的基本单位。编程的过程就是编写类的过程。

1. 类的声明

一个类在使用之前必须先声明。类声明的语法为：

【类修饰符】**class 类名**【extends 父类名】【implements 接口名列表】{类体}

其中：【】内是可选项，{ }内是类体。

类修饰符可以有多个，说明使用该类的权限，包括：

- public：访问控制符指明该类为公共类，可被其他类访问或引用其成员变量和成员方法。注意：Java 语言规定包含 main( )方法的类必须是公共类。
- abstract：抽象类，指明不能实例化的类。为更贴近人的思维和客观世界，定义它常常为改

试读结束：需要全本请在线购买： www.ertongbook.com

变它的子类而设置。

- final:终结类,指明该类不能有子类。

如该类没有修饰符,说明该类具有默认访问权限,即只允许与该类相同包中的类可以访问和调用,其他包中的类不能访问和使用。

class 关键字后是由编程人员定义的一个合法标识符的类名。

extends 关键字用于声明类是从某个父类派生,则该父类名应写在 extends 之后。

implements 关键字用于声明类要实现某些接口,而这些接口名应写在 implements 之后。

类体中声明了该类的所有变量和方法,称为**成员变量**和**成员方法**。下面是类声明的举例。

例 1.1 定义一个日期 Date 类,声明为:

```
public class Date{
    //成员变量
    private int day;
    private int month;
    private int year;
    //成员方法
    public void setDate(d,m,y){
        day = d;
        month = m;
        year = y;
    }
    public void showDate(){
        system. out. println(day+":"+month+":"+year);
    }
}
```

2. 成员变量

类体中的成员变量定义了类的特性,其声明语法为:

【变量修饰符】 类型 变量名【=初值】【,变量名【=初值】】,…;

变量修饰符可以有多个,说明使用该变量的权限和规则。变量修饰符包括:

- public:允许所有的类来访问该变量。
- protected:仅允许相同包的类和该类的子类来访问。
- private:仅允许本类访问的内部变量。
- static:表示该变量是**类变量**,是该类全部对象共享的变量,它独立于该类的任何对象,可直接进行访问。对一些类的对象具有相同的属性值时,可利用 static 来声明,如例 1.2 中的 PI。没有被 static 修饰的变量是**实例变量**(也称**对象变量**),只能通过对象来访问它。

成员变量的类型可以是 Java 中的任意数据类型。具体到某个成员变量其类型应该是唯一的。成员变量名是 Java 的合法标识符。一次可定义多个变量,每个变量可设置自己的初值。在例 1.1 中 day、month、year 就是实例变量的定义。

访问类变量值的语法与实例变量类似,其格式为:

对象名.类变量名

访问 Date 的实例变量值的语法为:

对象名.实例变量名

例 1.2 TestMemberVariable.java //对类变量和实例变量的访问实例

```
class Circle{
    static double PI = 3.14159265;
    int radius;
}
public class TestMemberVariable{
    public static void main(String args[]){
        Circle c = new Circle();              //Circle()是 Circle 类的构造方法
        c.radius = 10;                        //对实例变量赋初值
        System.out.println(Circle.PI);        //访问类变量
        System.out.println(c.radius);         //访问实例变量
        System.out.println(c.PI);             //访问类变量
    }
}
```

运行结果如下：

```
3.14159265
10
3.14159265
```

注意：成员变量区别于方法中声明的局部变量，成员变量的使用范围（作用域）在整个类，而局部变量的作用域只在方法内部。一旦从方法中返回，局部变量就消失。成员变量只存在于类中，不能出现在方法中。

3. 成员方法

类体中的成员方法定义了类的操作，其声明语法为：

【方法修饰符】 **返回值的类型 方法名**（参数表）{方法体}

方法修饰符包括：

- public、protected、private：表示访问权限，与变量修饰符意义相同。
- static：表示是**类方法**，可直接进行访问和调用。不用 static 修饰的方法称为**实例方法**。
- abstract：表示是抽象方法。没有定义方法体。static 方法和 final 方法都不允许是 abstract 方法。
- native：表示是其他语言代码的方法，用它与 Java 代码集成。
- synchronized：表示多个并发线程对共享数据访问的控制。
- final：表示是终结方法，不允许子类方法覆盖。

返回值的类型除 void 关键字说明该方法没有返回值外，编程者必须在方法声明中指定返回变量的数据类型，即在方法体中必须有 return 语句来指定返回值类型。

方法名是合法的 Java 标识符。()内的参数表是调用方法时所需传递消息的列表。

Java 允许参数列表为空。方法体是对方法的具体实现，由局部变量和 Java 语句代码组成。其中的局部变量只在该方法内有效，当方法返回时，局部变量就不存在了。通常局部变量由小写字母组成。

在 Java 程序中要调用方法需要根据方法是否有返回值和调用方法本身所在位置的情况而

定,原则是:

(1) 方法是否有返回值的情况。

- 如有返回值,将调用当做一个数值处理(类型必须与返回值相同)。

如 double x = avg(r,s,t);

System.out.println(avg(16.4,57.8,44.8)); //直接作为一个数值用

- 如没有返回值,可直接调用。

如 public void setDate(d,m,y){

```
    day = d;
    month = m;
    year = y;
}
```

Date birthdate = setDate(01,07,1990);

(2) 调用方法所在位置可能就在本类中,也可能在其他类或超类中。

- 任何情况可用实例变量名.方法名来调用。
- 如在类方法中,可直接调用本身的类方法,但不能直接调用实例方法。
- 如在实例方法中,可直接调用本身类的类方法或实例方法。
- 在实例方法中能用 this.方法名或 super.方法名调用。

例 1.3 TestCircumference.java //实例变量调用实例方法的举例

```
class Circle{
    static double PI = 3.14159265;
    int radius;
    public double circumference(){                    //实例方法
        return (2 * PI * radius);
    }
}
public class TestCircumference{
    public static void main(String args[]){
        Circle c1 = new Circle();
        c1.redius = 20;
        double circumf1 = c1.circumference();         //实例变量调用实例方法
        System.out.println("圆的周长=" + circumf1);
    }
}
```

运行结果如下:

```
圆的周长=125.663706
```

例 1.4 TestCircumference.java //调用类方法示例

```
class Circumference{
    static double PI = 3.14159265;
    public static double circumf(int radius){         //类方法
        return (2 * PI * radius);
    }
```

```
}
public class TestCircumference{
    public static void main(String args[]){
        //对类方法的直接调用
        System.out.println("半径 20 的圆周长 = " + Circumference.circumf(20));
    }
}
```

运行结果如下：

```
半径 20 的圆周长 = 125.663706
```

4. 类的组织——包

包是 Java 语言对一组相关的类、接口和子包进行封装的机制。图 1-3 显示了包的层次。

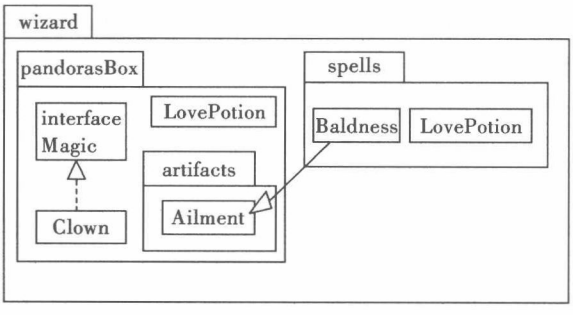


图 1-3 wizard 包

在图 1-3 中是 wizard 包，其中有 pandorasBox 和 spells 子包。在 pandorasBox 子包中有一个 Clown 类，它实现了一个接口 Magic。另外在 pandorasBox 子包中还有一个 LovePotion 类和一个 artifacts 子包，在 artifacts 子包中还有一个 Ailment 类。在 spells 子包中有两个类：Baldness 和 LovePotion，其中 Baldness 类是 Ailment 类的子类。

在包层次中唯一能用“.”符来确定包成员。如 wizard. pandorasBox. LovePotion 与 wizard. spells. LovePotion 表示不同的类。因此，在网络中对软件包都要求用包的层次全名。相同的包加上编程者的网络域名，网络就能区分使用。包可以嵌套包。包一般要求存放在指定的目录下，形成层次结构。Java 利用包来管理类名空间。有利于类和接口的安全、扩展和引用。包是有唯一命名的类的集合。一个 Java 编译单元由包声明（可以省略，用默认包）、import 语句（导入外部类）、类和接口的声明组成。声明一个包要用 package 语句，并将语句放在. java 文件中起始位置。其语法为：

```
package packageName(包的全名);
```

如编译单元没有包语句，编译器默认为该单元在主机当前工作的目录下。包名用小写字母开头。为使用包中的类，要求用 import 语句引入，其语法为：

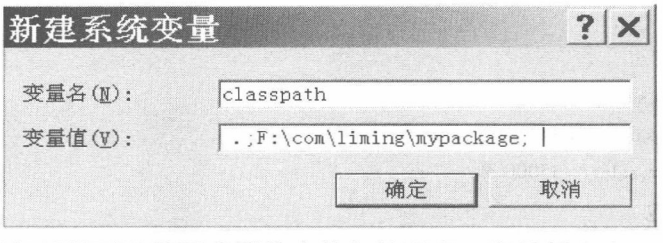
```
import package1【. package2…】. 类名 | *;
```

其中 package1【. package2…】即包的全名，类名是要用的类，也可用 * 表示全部类。

如下面代码段表示对包的引入：

```
package wizard. spells;                                    //包的声明
...
import wizard. pandorasBox. artifacts. * ;                 // 引入子包
...
public class Baldness extends Ailment{                    //从 Ailment 类继承
    wizard. pandorasBox. LovePotion tlcOne;                //类的全名
    LovePotion tlcTwo;                                     //在相同包中的类
    ...
}
...
```

编程者也可以将自编类编辑到一个自编包中。在程序引用包中的自编类时,也用 import 语句。包中的自编类必须用 pakage 语句打头。下面是引入自编包的实例,该程序实现对一个员工数组按照年薪从大到小排序,即用数组排序的 java. util. Arrays 类的 sort(数组对象名)方法。要求数组对象元素实现 java. lang. Comparable 接口的 compareTo( )抽象方法,该方法返回的比较标志是排序的依据。比较返回值,决定是升序还是降序排序。该程序对于年薪的排序用到了 java. lang. Comparable 接口,并引用 Employee 类,被编排进了李明的自编包 com. liming. mypackage 中。首先对自编包所在的文件夹设置 classpath 路径。假设该包 com 文件夹位于 F 盘根目录下:可在 Windows 操作系统中按以下步骤依次进入:开始 → 控制面板 → 系统 → 高级 → 环境变量 → 建立系统变量 classpath。



将 classpath 设置好后就可以编写李明的自编包的程序。参见例 1.5。

例 1.5 SalarySortTest. java

```
import com. liming. mypackage. * ;                         //引入了李明的自编包 com. liming. mypackage
import java. util. * ;                                     // 用 Arrays 类的静态方法 sort( )来排序
public class SalarySortTest{
    public static void main(String[ ] args){
        Employee[ ] staff = new Employee[3];               // Employee 类在自编包中
        staff[0] = new Employee("Harry Hacker", 35000);
        staff[1] = new Employee("Carl Cracker", 75000);
        staff[2] = new Employee("Tony Tester", 38000);
        Arrays. sort(staff);                               // 以对象数组 staff 为参数
        for(Employee e : staff)                            // 依次打印对象数组 staff 的元素 e
            System. out. println("name=" + e. getName( ) + ",salary=" + e. getSalary( ));
    }
}
```

李明自编包中的类:

```
package com.liming.mypackage;
public class Employee                                    // 包中的类都是 public 的
implements Comparable<Employee>{                         // 实现接口
    public Employee(String n, double s){
        name = n;
        salary = s;
    }
    public String getName(){
        return name;
    }
    public double getSalary(){
        return salary;
    }
    public int compareTo(Employee other){                // 实现接口中的抽象方法
        if (salary < other.salary) return 1;             // 提供给 Arrays.sort(staff);
        if (salary > other.salary) return -1;            // 提供给 Arrays.sort(staff);
        return 0;                                        // 提供给 Arrays.sort(staff);
    }
    private String name;
    private double salary;
}
```

运行结果如下：

```
name=Carl Cracker,salary=75000.0
name=Tony Tester,salary=38000.0
name=Harry Hacker,salary=35000.0
```

5. 基本类库的包

在 Java 中，所提供的基本类库 Java API 是由核心 java 包、javax 和 org 扩展包组成：

(1) java 核心包中含有的子包。

- java.lang 包：封装所有应用的基本类，如 Object、Class、String、System、Integer、Thread 等，而 Object 是所有类的根，它所包含的属性和方法被所有类继承。
- java.awt 包：是封装抽象窗口工具包，提供构建和管理用户图形界面功能。
- java.applet 包：为 Applet 提供执行所需的所有类。
- java.io 包：提供程序输入/输出文件操作的类。
- java.net 包：提供程序执行网络通信应用及 URL 处理的类。
- java.rmi 包：提供程序远程方法调用所需的类。
- java.math 包：提供程序常用的整数算术以及十进制算术的基本方法类。
- java.util 包：提供实用程序类和集合类，如系统特性定义和使用、日期函数类、集合 Collection、Map、List、Arrays 等常用工具类。
- java.sql 包：提供访问和处理标准数据源数据的类。
- java.security 包：提供网络安全操作类。
- java.text 包：提供所有处理文本、日期、数字以及非自然语言的消息操作的类。

试读结束：需要全本请在线购买： www.ertongbook.com