全国高等职业教育计算机类规划教材
工作过程系统化教程系列

2008年国家精品课程配套教材

过程导向
项目驱动
能力培养
面向就业

- 国家精品课软件编程（Java方向）课程配套教材
- 通过5个完整项目帮助读者领悟Java技术
- 代码经过严格测试，注释详尽，易于阅读
- 项目选择利于学生理解，针对性强

Java项目实战精编

陈显刚　李　季　主　编
张　静　孙凌玲　副主编
姜惠民　主审

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

全国高等职业教育计算机类规划教材·工作过程系统化教程系列
2008年国家精品课程配套教材

Java项目实战精编

陈显刚　李　季　主　编
张　静　孙凌玲　副主编
姜惠民　主　审

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书是为了满足新世纪高等职业学校教学的需要而编写的教材。本书较全面地介绍了 Java 基本设计和应用技术，内容包括面向对象的技术、Java Swing 技术、IO 技术、线程技术和网络技术及 Java 项目开发的过程等。

本书以奠定 Java 编程思维模式、培养 Java 项目开发能力为目标，注重 Java 项目开发技术的实用，通过项目介绍 Java 知识体系，由浅入深、循序渐进，符合认知规律及职业发展规划，并配有项目案例库。

本书可作为高等职业学院计算机专业专科及本科学生的教材，也可供与计算机相关专业的技术人员使用。

图书在版编目（CIP）数据

Java 项目实战精编 / 陈显刚，李季主编. —北京：电子工业出版社，2009.7
全国高等职业教育计算机类规划教材 • 工作过程系统化教程系列
ISBN 978-7-121-08966-4

Ⅰ. J… Ⅱ. ①陈…②李… Ⅲ. JAVA 语言－程序设计－高等学校：技术学校－教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2009）第 086181 号

策划编辑：程超群
责任编辑：王凌燕
印　　刷：
装　　订：北京京师印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：14. 5　字数：368 千字
印　　次：2009 年 7 月第 1 次印刷
印　　数：4 000 册　　定价：23.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

Java 语言是一种新型的网络编程语言，其卓越的特性为无数开发人员所推崇，目前越来越多的应用开发采用了基于 Java 技术的解决方案。Java 是一种简单的，面向对象的，分布式的，解释型的，健壮安全的，结构中立的，可移植的，性能优异、多线程的动态语言。作为一种真正面向对象的编程语言，它提升了应用程序的编程概念和开发思路；作为理想的面向对象的程序设计语言，Java 以自身的简单性和强大功能成为 Internet 编程和跨平台开发中最常用的开发语言。

Java 语言具有面向对象、与平台无关、安全、稳定和多线程等优良特性，是目前软件设计中极为强大的编程语言。Java 语言不仅可以用来开发大型的应用程序，而且特别适合 Internet 的应用开发。尤其是 Java Swing 推出之后，不仅使 Java 的功能更加强大，而且使 Java 具备了“处处可用”的特点，Java 已成为网络时代最重要的语言之一。

本书是工作过程导向系统化课程教材，是行动体系课程开发的成果。以培养能力为主线，按工作过程中不同工作任务的相关性来实现知识和实践技能的整合。按易学、易懂、易掌握的原则，结合 Java 技术，由浅入深，循序渐进地，通过项目介绍 Java 知识体系。

本书聘请启明信息科技股份有限公司高级软件工程师杨平参编。作为 ERP 项目组项目经理，他具有丰富的实践开发经验。他对本书的编写模式、项目设计思想、编码规范等方面给予指导，并根据企业常用的实际知识和技能，设计全书项目，以案例引领知识点，拓宽程序设计思路，通过实训项目提高实践技能。

全书共分 5 章。第 1 章通过掷骰子游戏项目，阐述面向对象的技术；第 2 章通过简单计算器项目，简述 Java Swing 技术；第 3 章通过聊天室项目，简述 IO 技术、线程技术和网络技术等；第 4 章通过二十一点游戏项目，加强 Java Swing 技术和分析问题的能力；第 5 章通过学生信息管理系统综合项目，进一步强化项目开发的能力，同时掌握软件开发过程。通过 5 个项目使学生理解并掌握利用 Java 技术解决实际问题的能力，而不是就 Java 技术而学 Java 技术。

本书的最大特点是通过项目对 Java 的知识点进行精心编排。项目设计顺序符合认知规律及职业规划发展规律，通过对项目的学习，加深读者对所学知识的理解和提升。通过对应的实训项目训练，提高分析问题和解决问题的能力。

本书由陈显刚、李季主编，张静、孙凌玲任副主编；参与本书编写的还有张雨、孙佳帝、金鑫、许春艳、乔丹、杨平；由姜惠民主审。

由于计算机技术发展十分迅速以及作者学识水平所限，加之时间仓促，书中的疏漏和错误在所难免，敬请广大读者不吝批评指正。

编　者

2009 年 4 月

目　录

第1章 掷 骰 子

1.1 项目目标

丢下两个骰子，若数值的总值为 7 点，则赢；否则输。

1.2 项目分析

项目的操作过程如下：3

（1）定义一个表示骰子的类，能掷出数字及获取当前掷出的数。

（2）定义另一个类，在此类中建立骰子类的两个对象，当掷出时判断两个骰子的数值的总值是否为 7，来确定是否成功。

（3）定义第三个类，对以上操作进行测试。

1.3 代码思路及实现

1.3.1 代码思路

1. 定义 Die 类

Die 类表示一个骰子有一个 faceValue 属性，为整型。在 Die 类中有 roll()方法和 getFaceValue()方法。roll()方法功能使 faceValue 属性设置为 1～6 中的一个随机值，没有返回值。getFaceValue()方法功能为取出 faceValue 值。

在本类中使用到了 Math.random()方法，此方法可以实现取随机数的功能，返回值为 double 类型，值的范围为 0.0～1.0。

2. 定义 DieGame 类

提示：DieGame 类有 die1、die2 两个属性，类型分别为 Die 类类型，有一个 play()方法。play()方法返回一个布尔类型，true 表示丢下两个骰子数值的总值为 7 点，否则为 false。

3. 编写一个测试类 DieTest，对上面定义的类进行测试

main()方法中产生 DieGame 对象，执行 play()方法后显示出输赢。

1.3.2 代码实现

1. Die 类

```
import java.util.*;
public class Die {
    private int faceValue;
    public void roll()
    {
        Random random=new Random();

        this.faceValue=Math.abs(random.nextInt()) %6+1;//取 1～6 间任意整数
    }
    public   int getFaceValue()
    {
        return this.faceValue;
    }
}
```

2. DieGame 类

```
public class DiceGame {

    public Die die1=new Die();
    public Die die2=new Die();

    public boolean play()                          //掷骰子，两粒骰子数相加得 7 为 true, 否则为 false
    {
        boolean flag=false;

        die1.roll();
        die2.roll();

        System.out.println(die1.getFaceValue());
        System.out.println(die2.getFaceValue());    //输出本次掷得的骰子数

        int num=die1.getFaceValue()+die2.getFaceValue();
        if(num==7)
        {
            flag=true;
        }
        else
        {
            flag=false;
        }
        return flag;
```

```
    }
}
```

3. 测试类 DieTest

```
public class DieTest {
    public static void main(String[] args) {
        DiceGame dg=new DiceGame();
        if(dg.play())
        {
            System.out.println("您赢了!");
        }
        else
        {
            System.out.println("您输了,请下次努力!");
        }
    }
}
```

1.4 运行与发布

1.4.1 运行

将 Die.java、DieGame.java 和 DieTest.java 3 个文件保存到一个文件夹中，如 e:\Die。在使用 javac 命令进行编译之前，应使用如下命令设置类路径：

```
e:\Die >set classpath= e:\Die
```

然后利用 javac 命令对文件进行编译，使用如下命令：

```
Javac DieTest.java
```

之后，使用 java 执行程序：

```
Java DieTest
```

程序即运行。

1.4.2 发布

使用 jar.exe 将应用程序打包，把应用程序中涉及的类和图片压缩成一个 jar 文件，这样就可以发布程序了。

步骤如下：

（1）编写清单文件，名为 MANIFEST.MF，保存到 e:\Die 文件夹下。其代码如下：

```
Manifest-Version: 1.0
Created-By: 1.5.0_02(Sun Microsystems Inc.)
Main-Class: Die
```

（2）使用如下命令生成 jar 文件：

```
jar cfm Die.jar MANIFEST.MF *.class
```

其中，c 表示要生成一个新的 jar 文件；f 表示要生成的 jar 文件的名字；m 表示清单文件的名字。

（3）为解决解压软件与.jar 文件的关联问题，在发布软件时还应该再编写一个 Die.bat 文件。其中只有如下一条命令：

```
javaw –jar Die.jar
```

以后就可以通过双击 Die.bat 来运行程序了。

1.5 本项目实现中常见问题

可以使用 java.util.Random 类来实现取随机数。取 1～6 中的任意整数时的表达式为：

```
Math.abs(random.nextInt()) %6+1
```

另外可以使用 Math.random()来实现取随机数的操作。

Math.random()的取值范围是 0～1 之间的任意小数，包括 0 但不包括 1，所以在取 1～6 中的任意整数时表达式要写成：

```
(int)(Math.random()*6)+1
```

1.6 项目技术支持

1.6.1 面向对象的基本概念

面向对象是一种新兴的程序设计方法，或者说是一种新的程序设计规范（paradigm），其基本思想是使用对象、类、继承、封装、消息等基本概念来进行程序设计。从现实世界中客观存在的事物（即对象）出发来构造软件系统，并且在系统构造中尽可能运用人类的自然思维方式。开发一个软件是为了解决某些问题，这些问题所涉及的业务范围称做该软件的问题域。其应用领域不仅仅是软件，还有计算机体系结构和人工智能等。

1．对象的基本概念

对象是系统中用来描述客观事物的一个实体，它是构成系统的一个基本单位。一个对象由一组属性和对这组属性进行操作的一组服务组成。从更抽象的角度来说，对象是问题域或实现域中某些事物的一个抽象，它反映该事物在系统中需要保存的信息和发挥的作用；它是一组属性和有权对这些属性进行操作的一组服务的封装体。客观世界是由对象和对象之间的联系组成的。

主动对象是一组属性和一组服务的封装体，其中至少有一个服务不需要接收消息就能主动执行（称做主动服务）。

2．类的基本概念

把众多的事物归纳、划分成一些类是人类在认识客观世界时经常采用的思维方法。分类

的原则是抽象。类是具有相同属性和服务的一组对象的集合，它为属于该类的所有对象提供了统一的抽象描述，其内部包括属性和服务两个主要部分。在面向对象的编程语言中，类是一个独立的程序单位，它应该有一个类名并包括属性说明和服务说明两个主要部分。类与对象的关系就如模具和铸件的关系，类的实例化结果就是对象，而对一类对象的抽象就是类。

3. 消息

消息就是向对象发出的服务请求，它应该包含下述信息：提供服务的对象标识、服务标识、输入信息和回答信息。服务通常被称为方法或函数。

1.6.2 面向对象的基本特征

1. 封装性

封装性就是把对象的属性和服务结合成一个独立的相同单位，并尽可能隐蔽对象的内部细节。它包含以下两个含义：

（1）把对象的全部属性和全部服务结合在一起，形成一个不可分割的独立单位（即对象）。

（2）信息隐蔽，即尽可能隐蔽对象的内部细节，对外形成一个边界（或者说形成一道屏障），只保留有限的对外接口使之与外部发生联系。

封装的原则在软件上的反映是：要求使对象以外的部分不能随意存取对象的内部数据（属性），从而有效地避免了外部错误对它的“交叉感染”，使软件错误能够局部化，大大减少查错和排错的难度。

2. 继承性

特殊类的对象拥有其一般类的全部属性与服务，称做特殊类对一般类的继承。例如，轮船、客轮，人、大人。一个类可以是多个一般类的特殊类，它从多个一般类中继承了属性与服务，这称为多继承。例如，客轮是轮船和客运工具的特殊类。在 java 语言中，通常称一般类为父类（superclass，超类），特殊类为子类（subclass）。

3. 多态性

对象的多态性是指在一般类中定义的属性或服务被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为。这使得同一个属性或服务在一般类及其各个特殊类中具有不同的语义。例如，“几何图形”的“绘图”方法，“椭圆”和“多边形”都是“几何图”的子类，其“绘图”方法功能不同。

1.6.3 类

类是 Java 中的一种重要的复合数据类型，是组成 Java 程序的基本要素。它封装了一类对象的状态和方法，是这一类对象的原形。一个类的实现包括两个部分：类声明和类体。

1. 类声明

```
[public][abstractlfinal] class className [extends superclassName] [implements interfaceNameList]
{...}
```

其中，修饰符 public、abstract、final 说明了类的属性，className 为类名，superclassName 为类的父类的名字，interfaceNameList 为类所实现的接口列表。

2. 类体

类体定义如下：

```
class className
{[public | protected | private ] [static]
  [final] [transient] [volatile] type
  variableName;                    //成员变量
  [public | protected | private ] [static]
  [final | abstract] [native] [synchronized]
  returnType methodName([paramList]) [throws exceptionList]
  {statements}                     //成员方法
}
```

3. 成员变量

成员变量的声明方式如下：

```
[public | protected | private ] [static] [final] [transient] [volatile] type
variableName;              //成员变量
```

其中，

static：静态变量（类变量），相对于实例变量。

final：常量。

transient：暂时性变量，用于对象存档。

volatile：贡献变量，用于并发线程的共享。

4. 成员方法

方法的实现包括两部分内容：方法声明和方法体。

```
[public | protected | private ] [static]
[finall abstract] [native] [synchronized]
returnType methodName([paramList])
[throws exceptionList]       //方法声明
{statements}                 //方法体
```

方法声明中的限定词的含义如下。

static：类方法，可通过类名直接调用。

abstract：抽象方法，没有方法体。

final：方法不能被重写。

native：集成其他语言的代码。

synchronized：控制多个并发线程的访问。

1.6.4 对象

类实例化可生成对象，对象通过消息传递来进行交互。消息传递即激活指定的某个对象的方法以改变其状态或让它产生一定的行为。一个对象的生命周期包括三个阶段：生成、使用和消除。类与对象的关系如图 1-1 所示。

1. 对象的生成

对象的生成包括声明、实例化和初始化。

格式为：

```
type objectName=new type([paramlist]);
```

（1）声明：type objectName。声明并不为对象分配内存空间，而只是分配一个引用空间；对象的引用类似于指针，是 32 位的地址空间，它的值指向一个中间的数据结构，它存储有关数据类型的信息以及当前对象所在的堆的地址，而对于对象所在的实际的内存地址是不可操作的，这就保证了安全性。

（2）实例化：运算符 new 为对象分配内存空间，它调用对象的构造方法，返回引用；一个类的不同对象分别占据不同的内存空间。

（3）生成：执行构造方法，进行初始化；根据参数不同调用相应的构造方法。

```
Student s1=new Student();
```

上面语句按照 Student 类生成一个 s1 对象，或者用下面两条语句。

```
Student s1;
s1=new Student();
```

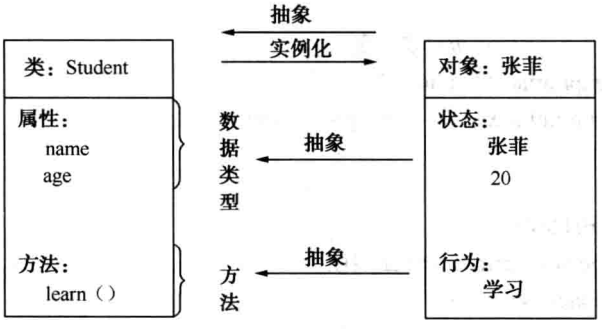

图 1-1 类与对象关系

2. 对象的使用

通过运算符“.”可以实现对变量的访问和方法的调用。变量和方法可以通过设定访问权限来限制其他对象对它的访问。

（1）调用对象的变量。

格式：对象名.变量

对象名是一个已生成的对象，也可以是能生成对象的表达式。

例如：s.name="张三";

（2）调用对象的方法。

格式：对象名.方法名([paramlist]);

例如：s.show( );

3. 对象的清除

当不存在对一个对象的引用时，该对象成为一个无用对象。Java 的垃圾收集器自动扫描对象的动态内存区，把没有引用的对象作为垃圾收集起来并释放。

```
System.gc( );
```

当系统内存用尽或调用 System.gc( )要求垃圾回收时，垃圾回收线程与系统同步运行。

1.6.5 方法

一个类中可以有很多方法，这些方法用来实现类某些功能或动作。

1. 局部变量与类的成员变量同名

方法体是对方法的实现，它包括局部变量的声明以及所有合法的 Java 指令。方法体中声明的局部变量的作用域在该方法内部。若局部变量与类的成员变量同名，则类的成员变量被隐藏。

【例 1-1】 本例说明了局部变量 z 和类成员变量 z 的作用域是不同的。

```
class Variable{
    int x=0,y=0,z=0;                    //类的成员变量
    void init(int x,int y) {
    this.x=x;
    this.y=y;
    int z=5;                            //局部变量
    System.out.println("** in init**");
    System.out.println("x="+x+" y="+y+" z="+z);
  }
}
public class VariableTest{
    public static void main(String args[]){
    Variable v=new Variable();
    System.out.println("**before init**");
    System.out.println("x="+v.x+" y="+ v.y+" z="+v.z);
    v.init(20,30);
    System.out.println("**after init**");
    System.out.println("x="+v.x+ " y="+ v.y+" z="+v.z);
  }
}
```

运行结果为：

```
**before init**
x=0 y=0 z=0
** in init **
```

```
x=20 y=30 z=5
**after init**
x=20 y=30 z=0
```

上例中用到了 this，这是因为 init()方法的参数名与类的成员变量 x,y 的名字相同，而参数名会隐藏成员变量，所以在方法中，为了区别参数和类的成员变量，必须使用 this。

this：用在一个方法中引用当前对象，它的值是调用该方法的对象。返回值须与返回类型一致，或者完全相同，或是其子类。当返回类型是接口时，返回值必须实现该接口。

2. 方法中的参数传递

对于简单数据类型来说，java 实现的是值传递，方法接收参数的值，但不能改变这些参数的值。如果要改变参数的值，则用引用数据类型，因为引用数据类型传递给方法的是数据在内存中的地址，方法中对数据的操作可以改变数据的值。

（1）按值传递方式。

【例 1-2】　按值传递实例演示。

```
class Test{
  static show add(int a){
      a=a+10;
  }
 public static void main(String args[]){
    int a=10;
    System.out.println("调用方法前 a 的值为"+a);
    Test.add(a);
    System.out.println("调用方法后 a 的值为"+a);
   }
}
```

运行结果为：

调用方法前 a 的值为 10

调用方法后 a 的值为 10

Press any key to continue...

（2）按地址传递方式。

【例 1-3】　按地址传递实例演示。

```
class Student{
  String name;
  void change(Student op){
    name=op.name;
 }
}
class TestStudent{
 public static void main(String args[]){
      Student s1=new Student();
      s1.name="Jack";
      Student s2=new Student();
```

```
        s2.name="Tom";
        System.out.println("打印前的 s1.name 的值是"+s1.name);
        s1.change(s2);
        System.out.println("打印后的 s1.name 的值是"+s1.name);
    }
}
```

运行结果为：

打印前的 s1.name 的值是 Jack

打印后的 s1.name 的值是 Tom

Press any key to continue...

3. 方法重载

方法重载是指多个方法享有相同的名字，但是这些方法的参数必须不同，或者是参数的个数不同，或者是参数类型不同。返回类型不能用来区分重载的方法。

参数类型的区分度一定要足够，如不能是同一简单类型的参数，如 int 与 long。

【例 1-4】 方法重载实例演示。

```
class Abs{
  int abs(int a){
    if(a>0)
      return a;
    else
      return -a;
  }
  long abs(long a){
    if(a>0)
      return a;
    else
      return -a;
  }
  double abs(double a){
    if(a>0)
      return a;
    else
      return -a;
  }
}
class AbsTest{
public static void main(String arg[]) {
    Abs ob=new Abs();
    int result1;
    long result2;
    double result3;
    result1=ob.abs(5);
    result2=ob.abs(23456789);
```