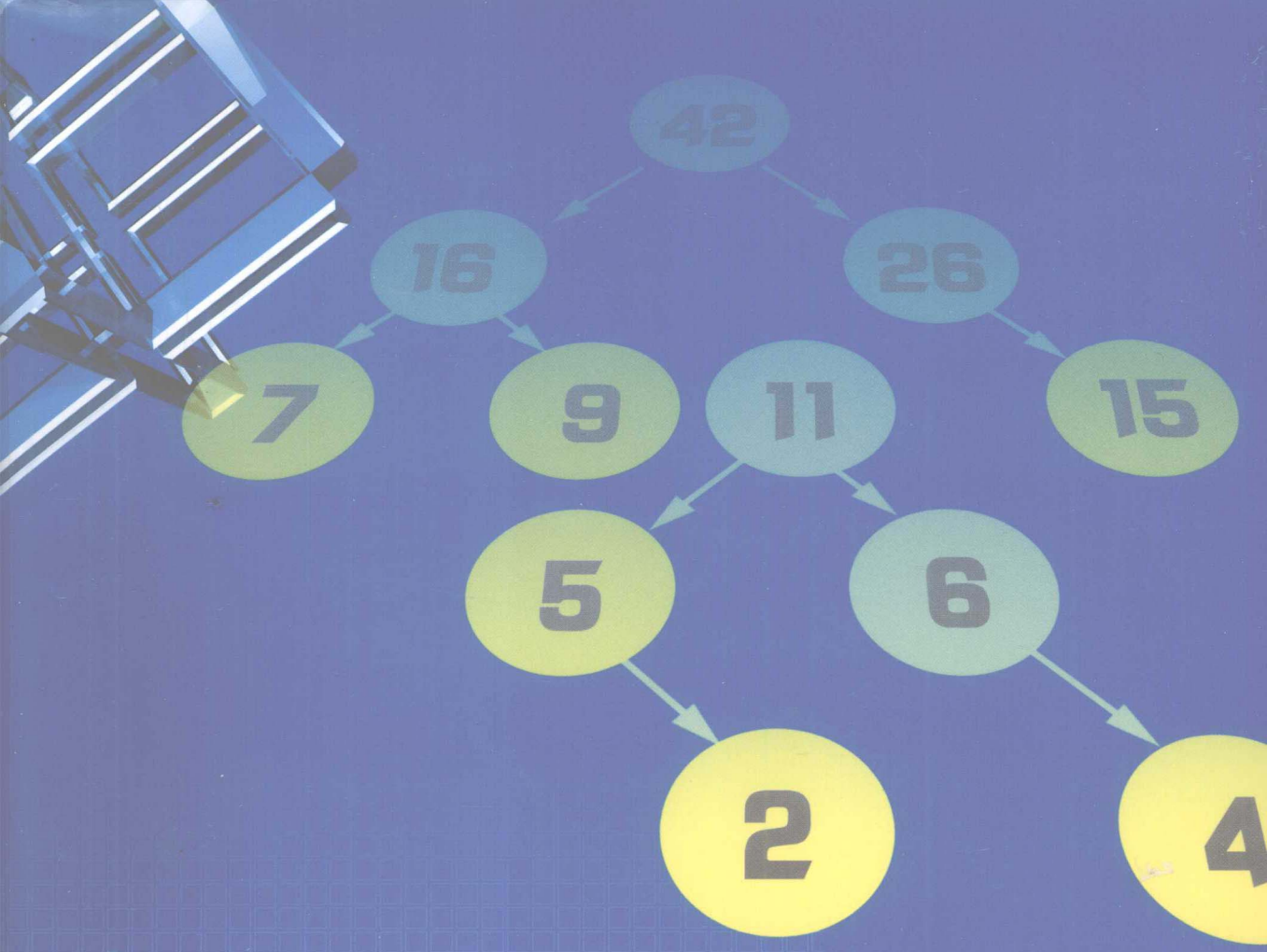

数据结构与算法（JAVA语言版）

SHUJU JIEGOU YU SUANFA JAVA YUYANBAN

周鹏 雷国洪 谢从满 邓小炼 编著

湖北科学技术出版社

数据结构与算法（JAVA语言版）

SHUJU JIEGOU YU SUANFA JAVA YUYANBAN

周鹏 雷国洪 谢从满 邓小炼 编著

湖北科学技术出版社

图书在版编目(CIP)数据

数据结构与算法：JAVA语言版／周鹏等编著. —武汉：湖北科学技术出版社，2008.12

ISBN 978-7-5352-4254-9

Ⅰ. 数… Ⅱ. 周… Ⅲ. ①数据结构②算法分析③JAVA语言—程序设计 Ⅳ. TP311.12 TP312

中国版本图书馆CIP数据核字（2008）第186846号

责任编辑：高诚毅　　封面设计：喻　杨

出版发行：湖北科学技术出版社　　电话：027-87679468

地　　址：武汉市雄楚大街268号　　邮编：430070

（湖北出版文化城B座12-13层）

网　　址：http: //www.hbstp.com.cn

印　　刷：荆州市翔羚印刷有限公司　　邮编：434000

787 × 1092　1/16　　15.25印张　　337千字

2008年12月第1版　　2008年12月第1次印刷

定价：28.00元

内容提要

本书根据抽象数据类型的实现方法与 Java 语言的面向对象特性，抽象、封装了线性表、堆栈、队列、二叉树、图等基本数据结构，较好地体现了面向对象的思想。在算法部分，介绍了基于归纳的递归、分治法、贪心法等基本的算法与设计技巧，以及均摊分析、Master method 等高级分析方法，并给出了相应的实现代码，其中许多代码可以直接运用到工程实践中。本书可作为普通高等院校学生的教材，也可作为具有一定 Java 基础的读者学习数据结构的参考书。

前　言

数据结构是在非数值计算的程序设计问题中研究计算机的操作对象及其相互关系和运算等的学科，是计算机应用相关专业的基础课。

在了解编程语言的基本语法后，通过对数据结构课程的学习，能够让读者进一步熟悉编程语言，熟练使用该语言来表达自己的思想，毕竟任何语言（包括自然语言和机器语言）都是用来表达思想的。教学中，采用常用的程序设计语言来描述数据结构与算法，让实例代码可被直接使用，课程会更有实用价值。

习惯上，大多数理工科学生的程序设计入门教程是 C 语言程序设计，数据结构课程采用 C/C++语言描述。C 语言是面向过程的程序设计语言的优秀代表，C++属于面向对象的程序设计语言，当今的主流程序设计语言是面向对象的，但在现有数据结构教程中涉及 C++面向对象的思想相对较少。

Java 是目前流行的纯面向对象的程序设计语言之一，使用 Java 语言，可以充分运用其面向对象的思想来有效组织和应用集合、线性结构、树形结构、图状结构等数据对象。虽然 Java 本身已经基本实现了这些数据结构，但学习其实现原理，能够快速掌握许多解决问题的最优方法。

为此，可以在一年级下学期就直接开设 Java 语言课程，继而用 Java 语言讲授数据结构，进一步阐述 Java 面向对象的编程思想，最后介绍以 Java 相关技术为基础的 MVC 框架及其 WEB 应用，取得了较好的实践效果。

本书要求学生具备一定的面向对象程序设计基础，难度不大，比较适合普通本科院校计算机类和信息类相关专业作为教材使用。考虑到不同专业的培养计划以及教学课时的差别，书中的部分内容仅给出了实现程序，没有将相应的面向对象的设计方法与数据结构的抽象数据类型结合起来展开阐述。

编　者

2008 年 6 月

目 录

第 1 章 Java 与面向对象程序设计

本章简要介绍 Java 的基本知识。熟悉 Java 的读者可以直接阅读第二章。

Java 语言是一种具有许多良好特性(如面向对象、可移植性、健壮性等)并广泛使用的计算机高级程序设计语言,这里只阐述理解书中 Java 代码所必需的相关内容。

1.1 Java 语言基础知识

1.1.1 基本数据类型及运算

在 Java 中每个变量使用前均必须声明它的类型。Java 共有八种基本数据类型:四种整型,两种浮点型,一种字符型以及用于表示真假的布尔类型。各种数据类型的细节如表 1 -1 所示。

表 1 -1 Java 数据类型

类型	存储空间	范围
int	32bit	[-2147483648,2147483647]
short	16bit	[-32768,32767]
long	64bit	[9223372036854775808,9223372036854775807]
byte	8bit	[-128,127]
float	32bit	[-3.4E38,3.4E38]
double	64bit	[-1.7E308,1.7E308]
char	16bit	[Unitcode 字符]
boolean	8bit	[true,false]

声明一个变量时,应先给出此变量的类型,随后写上变量名。一行中可以声明多个变量,声明变量的同时可以对变量进行初始化。例如:

```
int i;
double x, y = 1.2;
char c = 'z';
boolean flag;
```

程序设计常常需要在不同的数值数据类型间进行转换。图 1 -1 给出了数值数据类型间的合法转换。

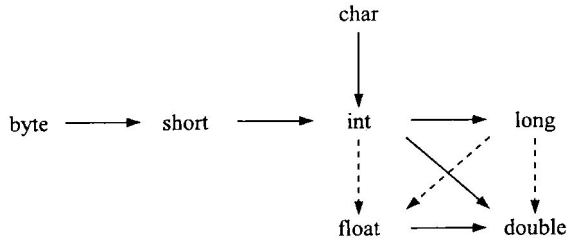

图 1-1 数值数据类型间的合法转换

图 1-1 中六个实箭头表示无信息损失的转换,而三个虚箭头表示的转换则可能会丢失精度。

进行其他非合法的转换可能会丢失信息。Java 通过强制类型转换来完成可能会丢失信息的转换。其语法是在需要进行强制类型转换的表达式前使用圆括号,圆括号中是需要转换的目标类型。例如:

```
double x = 7.8;
int n = (int)x;          //n 等于 7
```

Java 使用常见的算术运算符 + - * /进行加、减、乘、除运算。当除法运算符/作用于两个整数时,运算结果是整数。整数的模(即求余)运算使用 % 运算符。对整型变量还有一种常见操作:递增与递减运算。与 C/C++一样,Java 也支持递增和递减运算符。例如:

```
int n = 7, m = 2;
double d = 7;
n = n/m;                 //n 等于 3
d/ = m;                  //d 等于 3.5
n - -;                   //n 等于 2
int a = 2 * n + +;       //a 等于 4
int b = 2 * + +m;        //b 等于 6
```

此外,Java 还具有完备的关系运算符,如 = =(是否相等),<(小于),>(大于),<=(小于等于),>=(大于等于),! =(不等于);并且 Java 使用 && 表示逻辑与,‖表示逻辑或,! 表示逻辑非;以及七种位运算符 &(与)、|(或)、^(异或)、~(非)、>>(右移)、<<(左移)、>>>(高位填充 0 的右移)。

最后,Java 还支持一种三元运算符 ?:,这个运算符有时很有用。它的形式为:

condition ?e1 :e2

这是一个表达式,在 condition 为 true 时返回值为 e1,否则为 e2。例如:

```
min = x < y ? x : y;
```

这时 min 变量中将保存 x 和 y 二者间的较小值。

1.1.2 流程控制语句

计算机高级语言程序设计中共有三种流程结构,分别是:顺序、分支、循环。其中分

支与循环流程结构需要使用固定的语法。

Java 中有两种语句可用于分支结构，一种是 if 条件语句，另一种是 switch 多选择语句。

条件语句的形式如下：

```
if (condition )statement1 else statement2
```

当 if 后的条件 condition 的值为 true 时执行 statement1 中的语句，否则执行 statement2 中的语句。

多选择语句的形式为：

```
switch (integer expression ) {
    case value1 : block1 ; break ;
    case value2 : block2 ; break ;
     …
    case valueN : blockN ; break ;
    default : default block ;
}
```

switch 语句从与选择值相匹配的 case 标签处开始执行，一直执行到下一个 break 处或者 switch 的末尾。如果没有相匹配的 case 标签，而且存在 default 子句，那么执行 default 子句。如果没有相匹配的 case 标签，并且没有 default 子句，则结束 switch 语句的执行，执行 switch 后面的语句。

Java 中的循环语句主要有三种，分别是 for 循环、while 循环、do…while 循环。

for 循环的形式为：

```
for (initialization ; condition ; increment )statement ;
```

for 语句循环控制的第一部分通常是对循环变量的初始化，第二部分给出进行循环的测试条件（在循环条件满足的情况下才执行循环体），第三部分则是对循环变量的更新。

while 循环的形式为：

```
while (condition )statement ;
```

while 循环首先对循环条件进行测试，只有在循环条件满足的情况下才执行循环体。

do…while 循环的形式为：

```
do statement while (condition ) ;
```

与 while 循环不同的是，do…while 循环首先执行一次循环体，当循环条件满足时则继续进行下一次循环。

1.1.3　字符串

字符串指一个字符序列。Java 没有内置的字符串数据类型，而是在标准 Java 库中包含一个名为 String 的预定义类。每个被一对双引号括起来的字符序列均是 String 类的一个实例。字符串可以使用如下方式定义：

```
String s1 = null;          //s1 指向 NULL
String s2 = "";            //s2 是一个不包含字符的空字符串
```

String s3 = "Hello";

Java 允许使用符号 + 把两个字符串连接在一起。当连接一个字符串和一个非字符串时,后者将被转换成字符串,然后进行连接。例如:

```
s3 = s3 + "World!";          //s3 为"HelloWorld!"
String s4 = "abc" + 123;     //s4 为"abc123"
```

Java 的 String 类包含许多方法,其中多数均非常有用,表 1 - 2 给出了最常用的一些方法。

表 1 - 2　　Java String 类常用方法

char	charAt(int index) Returns the char value at the specified index.
int	compareTo(String anotherString) Compares two strings lexicographically.
int	compareToIgnoreCase(String str) Compares two strings lexicographically, ignoring case differences.
boolean	endsWith(String suffix) Tests if this string ends with the specified suffix.
boolean	equals(Object anObject) Compares this string to the specified object.
boolean	equalsIgnoreCase(String anotherString) Compares this String to another String, ignoring case considerations.
int	indexOf(String str) Returns the index within this string of the first occurrence of the specified substring.
int	lastIndexOf(String str) Returns the index within this string of the rightmost occurrence of the specified substring.
int	length() Returns the length of this string.
boolean	startsWith(String prefix) Tests if this string starts with the specified prefix.
String	substring(int beginIndex) Returns a new string that is a substring of this string.
String	substring(int beginIndex, int endIndex) Returns a new string that is a substring of this string.
char[]	toCharArray() Converts this string to a new character array.

续表

String	toLowerCase( ) Converts all of the characters in this String to lower case using the rules of the default locale.
String	toString( ) This object(which is already a string!) is itself returned.
String	toUpperCase( ) Converts all of the characters in this String to upper case using the rules of the default locale.
String	trim( ) Returns a copy of the string, with leading and trailing whitespace omitted.

如果读者需要进一步了解有关 String 提供的其他方法及方法完成的功能，可以通过在线 API(应用程序接口)文档了解相关信息，从中可以查到标准库中所有的类及方法。API 文档是 Java SDK 的一部分，以 HTML 格式显示。JDK1. 5. 0 的 API 文档地址为：http://java. sun. com/j2se/1. 5. 0/docs/api/index. html。

1.1.4　数　组

数组是用来存放一组具有相同类型数据的数据结构。可以通过整型下标来访问数组中的每一个值。数组是可以通过在某种数据类型后面加上[ ]来定义，在此之后跟上变量名就可以定义相应类型的数组变量。例如：

```
int[ ]a;
```

还可以使用另一种方法定义数组，例如：

```
int a[ ];
```

以上这两种方法的定义是等价的。在这里只定义了一个整型数组变量 a，但是还没有将 a 真正初始化为一个数组。将一个数组初始化可以使用 new 关键字，也可以使用赋值语句。数组一旦被创建，就不能改变它的大小。

例如：

```
a = new int[10];      //将 a 初始化为大小为 10 的整型数组。
int[ ]b = {0,1,2,3}   //将 b 初始化为大小为 4 的整型数组，
                      //并且 4 个分量的值分别等于 0,1,2,3
```

数组的下标从 0 开始计数，到数组大小减 1 结束。在 Java 中不能越过数组的下标范围去访问数组中的数据。例如：

```
a[10] = 10;
```

如果越过数组的下标范围访问，则会产生一个名为 ArrayIndexOutOfBoundsException 的运行时错误。要避免产生这种错误，应在访问某个下标对应的数组元素前检查数组的大小。数组大小可以通过数组的成员变量 length 返回。例如：

```
for( int i = 0;i < a. length;i + + )
    a[i] = i;
```

在 Java 中数组实际上是一个类,因此两个数组变量可以指向同一个数组。

请读者预测以下这段代码的运行结果:

```
int[ ]a = {1,1,1};
int[ ]b = a;
for(int i =0;i < b. length;i + + )
    b[i] + + ;
for(int i =0;i < a. length;i + + )
    System. out. print(a[i]);
```

这段代码对数组 b 的每个分量加 1,输出数组 a 的每个分量时,可以发现 a 的每个分量都发生了变化。其原因与赋值语句 int[ ]b = a 有关。该语句将对数组 a 的引用传递给了变量 b,此时数组变量 a、b 实际上是指向同一个数组空间。图 1 - 2 说明了这段代码运行时的情况。

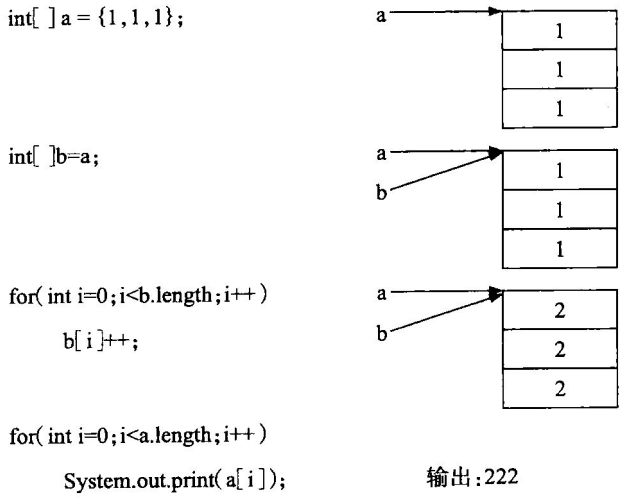

图 1-2　两个数组变量指向同一数组空间

为了得到两个分量值均相等但空间不同的数组,可以使用循环语句或 System 类中的 arraycopy 方法来完成。

同样,当数组作为方法的参数传递时,传递的是对数组的引用,因此在方法中操作数组会影响到原来的数组。例如:

```
public void changeArray(int[ ]a){
    for(int i =0;i < a. length;i + + )
        a[i] = a[i] * 2;
}
```

那么如下代码

```
int c = {2,2,2};
changeArray(c);
for(int i =0;i < c. length;i + + )
```

```
System. out. print( c[ i ] ) ;
```

的运行结果为:444。

1.2　Java 的面向对象特性

面向对象程序设计(Object Oriented Programming, OOP) 是目前主流的程序设计方法,它已经取代了以前基于过程的程序设计技术。面向对象程序设计主要是指在程序设计中采用抽象、封装、继承等设计方法。

1.2.1　类与对象

在面向对象思想中,抽象决定了对象的对外形象、内部结构以及处理对象的外部接口,其关键是处理对象的可见外部特征。抽象主要是从现实世界中抽象出合理的对象结构。

封装性是保证软件部件具有优良模块性的基础。在 Java 中,最基本的封装单元是类,一个类定义了一组对象所共享的行为(数据和代码)。一个类的每个对象均包含类所定义的结构与行为,这些对象就像是一个模子铸造出来的。所以对象也叫做类的实例。

1.2.1.1　类的定义

在定义一个类时,需要指定构成该类的代码与数据,类所定义的数据叫做成员变量,操作数据的代码叫做成员方法。方法定义怎样使用成员变量,这意味着类的行为和接口要由操作数据的方法来定义。

Java 提供公有和私有的访问模式,类的公有接口代表外部用户应该知道或可以知道的每件东西。类内部复杂的私有方法和数据只能被该类的内部代码访问。这样,类的内部行为实现被有效封装,外部不能也不需要知道。

类的定义通过关键字 class 实现。例如:

```
public class People {
    private String name;
    private String id;
    //Constructor
    public People( ) {
        this("","");
    }
    public People(String name,String _id) {
        this. name = name;
        id = _id;
    }
    public void sayHello( ) {
        System. out. println( "Hello! My name is " + name) ;
    }
```

```
    public void sayHello(String name){
        System.out.println("Hello, " +name +"! My name is " +this.name);
    }
    //get & set methods
    public void setName(String name){
        this.name = name;
    }
    public void setId(String id){
        this.id = id;
    }
    public String getName(){
        return this.name;
    }
    public String getId(){
        return this.id;
    }
}
```

代码中使用 class 关键字定义了一个名为 People 的类,class 前面的 public 关键字表示这个类是一个公有类,可被任何其他类访问。People 类中使用了 this 关键字,this 关键字主要有两个作用,一是引用隐式参数,二是调用类中的其他构造方法。

在类的内部,通过 private 关键字定义了两个私有的成员变量,因为类的成员变量一般不可以让外部随意访问和修改。作为示例,代码随后定义了四个 get、set 方法,使得外部能够获取或修改这些信息,实际应用应该根据需要取舍。

People 类定义了两个构造方法,构造方法是一种特殊的方法,其作用是构造并初始化对象,在一个类中构造方法可以定义多个。要构造一个新的对象只需要通过 new 操作符调用构造方法。例如:

```
People jack = new People("Jack","0001");
```

此外在类中还定义了两个 sayHello 方法实现与外界的互操作。

1.2.1.2 使用现有类

Java 提供了大量的预定义类可供使用,同时程序中可能还会使用第三方提供的或者自己编写的属于其他包的类,使用这些类会给编写程序带来巨大的便利。

如果在某个类中需要引用其他包中的类,可以使用关键字 import 将需要使用的类引入。

例如在 People 类中又定义了一个新的成员变量 birthday,数据类型是日期,而日期可以使用 Java 提供的预定义类 Calendar 实现。以下代码给出了实现方法。

```
import java.util.Calendar;
public class People {
    private String name;
```