HZ BOOKS
华章教育

全美经典
学习指导系列

数据结构
习题与解答
Java语言描述

DATA STRUCTURES WITH JAVA

最佳的复习资料，实用的辅助教材

与国外高校计算机水平保持同步

为考研和出国深造奠定坚实基础

John R.Hubbard 著
阳国贵 周斌 刘亚萍 等译

全球销售超过3000万册

机械工业出版社
China Machine Press
中信出版社
CITIC PUBLISHING HOUSE

全美经典
学习指导系列

数据结构习题与解答

Java语言描述

DATA STRUCTURES WITH JAVA

John R. Hubbard 著
阳国贵 周斌 刘亚萍 等译

机械工业出版社 China Machine Press
中信出版社 CITIC PUBLISHING HOUSE

本书介绍了网络环境下体现软件工程和面向对象理念的JAVA编程语言，讲述了它的面向对象、分布式、可移植性等诸多特点以及如何使用JAVA语言来实现各种主要的数据结构。全书使用了大量的源程序，从一个程序员的视角重新诠释了数据结构以及JAVA实现，是一本不可多得的好书。

本书版权登记号：图字：01-2001-5201

图书在版编目（CIP）数据

数据结构习题与解答：Java语言描述/（美）哈伯德（Hubbard, J. R.）著；杨国贵等译. -北京：机械工业出版社，2002.8

（全美经典学习指导系列）

书名原文：Data Structures with Java

ISBN 7-111-10829-9

Ⅰ.数… Ⅱ.①哈… ②杨… Ⅲ.①语数结构－解题－解题 ②JAVA语言－程序设计－解题 Ⅳ.TP311-44

中国版本图书馆CIP数据核字（2002）第062687号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码100037）

责任编辑：华章

北京忠信诚印刷厂印刷·新华书店北京发行所发行

2002年8月第1版第1次印刷

787mm×1092mm 1/16·27.5印张

印数：0 001-5 000册

定价：39.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

作 者 序

与所有 Schaum 系列丛书中的其他书籍一样，本书主要是为自学者编写的，特别适合与数据结构的课程一起使用，同时也可以作为单独的学习用书或参考书。

书中包括 200 多个例题、习题及习题解答。作者坚信数据结构原理可以通过精心设计并有完整解释的例子开始学习，本书正体现了这一观点。

书中所有例题和习题的源代码均可从http://projectEuclid.net/schaums/dswj/ 或其镜像站点 http://www.richmond.edu/~hubbard/schaums/dswj/进行下载，对本书的任何修正或补充部分也可从这些站点获取。

感谢对书稿提出过改正意见的所有朋友、同事、同学以及 McGraw-Hill 出版社的工作人员。在此，还要特别感谢我的妻子及同事 Anita H.Hubbard，本书得益于她的诸多建议、鼓励和创造性思想，并采用了许多由她提供的习题。

目　录

第1章 Java回顾

本章概述数据结构设计和实现所用的 Java 语言特征，如要了解更为详尽的内容，可以参见附录 D 中[Hubband]条目下所列的参考书。

1.1 面向对象的程序设计

Java 是一种面向对象的程序设计语言，在该语言中：

- 每个数据项均封装在某个对象中。
- 每条可执行的语句均由某个对象来完成。
- 每个对象均是某个类的一个实例或是一个数组。
- 每个类均在一个单继承的层次结构中定义。

Java 中的单继承层次结构是一种树型结构，它的根是 Object 类（见 3.4 节）。这种单继承结构似乎使 Java 成为一种简单的语言，Java 1.3 类库就是这样的一个结构，但该类库却包含 2 130 个类并有 23 901 个成员。

Java 的面向对象的特点使得它特别适合于数据结构的设计和实现，它的集合框架（Collections Framework）（见第 4 章）为程序员构造数据结构提供了一个很好的平台。

1.2 程序设计语言 Java

一个 Java 程序由一个或多个文本文件组成，其中至少有一个文本文件包含了一个 public 类（惟一的一个），在该类中含有惟一的一个 main 方法，其典型用法如下：

```
public static void main(String[ ] args)
```

程序中的每个文件必需包含一个惟一的 public 类或 public 接口定义。对名为 X.java 的文件而言，X 就是该文件中定义为 public 的类名或接口名。每一个这样的文件将被编译为对应的名为 X.class 的类文件。然后，该程序就可通过执行包含 main(String[ ])方法的 X.class 文件来运行，它将执行 main(String[ ])方法，该方法中包含了要执行的语句。程序可以命令行的方式来执行，也可在 IDE（集成开发环境）中运行（参见附录 C）。

1.3 变量和对象

在任何程序设计语言中，数据的存取均是通过变量来进行的。在 Java 中，一个变量要么是对象的一个引用，要么是 8 种基本类型之一，这些类型间的关系如图 1.1 所示。如果变量是一个引用，那么它的值要么为空（null），要么包含一个实例对象的地址。

每个对象属于一个特定的类。类定义了对象的特性和操作，正如一个基本类型定义了它的变量的特性和操作一样。

一个变量是由说明其类型和初值（可选）的声明来定义的，对象由调用类构造器的 new 操作创建。

例 1.1 创建变量和对象

```
public class Ex0101
{ public static void main(String[] args)
  { boolean flag=true;
    char ch='B';
    short m;
    int n=22;
    float x=3.14159F;
    double y;
    String str;
    String nada=null;
    String country = new String("United States");
    System.out.println("flag = " + flag);
    System.out.println("ch = " + ch);
    System.out.println("n = " + n);
    System.out.println("x = " + x);
    System.out.println("nada = " + nada);
    System.out.println("country = " + country);
  }
}
```

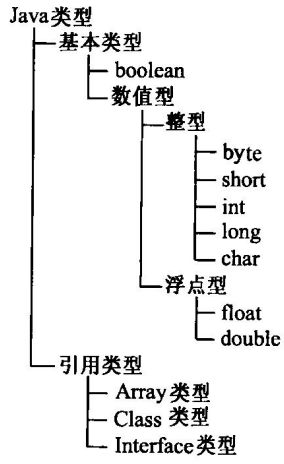

图 1.1 类型间的关系

经编译后运行该程序，其输出将为：

```
flag=true
ch=B
n=22
x=3.14159
nada=null
country=United States
```

在该程序中声明了 9 个变量，随后打印输出了有初值的 6 个变量，这 6 个变量都是基

本类型变量，最后的 3 个变量是指向字符串对象的引用。

程序中仅含有一个对象，即名为 country 的 String（字符型）对象（用更为专业的术语讲，country 是指向 String 对象的变量名）。

图 1.2 显示了 9 个变量和一个对象。每个变量均有一个名字和类型。当类型是一个类时，变量是指向那个类的实例（即一个对象）的一个引用，没有初始值的变量为空，其余的为初值。一个引用变量的惟一可能的值就是一个引用，用黑点表示。当引用不为空时，该点成为指向所引用的对象的箭头。在 Java 中，除非有某个引用变量引用某个对象，否则对象不能够存在。

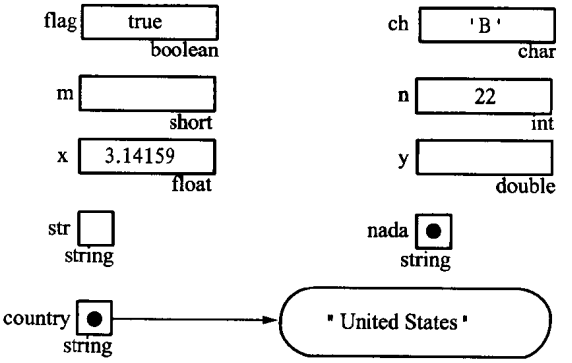

图 1.2　变量与对象

1.4　基本类型

Java 定义了 8 种基本类型：

boolean	布尔型，其值为 false 或 true。
char	16 位统一码，例如，'B' 或 'π'。
byte	8 位整数，取值范围为 -128～127。
short	16 位整数，取值范围为 -32 768～32 767。
int	32 位整数，取值范围为 -2 147 483 648～2 147 483 647。
long	64 位整数，取值范围为-9 223 372 036 854 775 808～9 223 372 036 854 775 807。
float	32 位浮点小数，取值范围为（正或负）1.4E-45F～3.4E38F。
double	64 位浮点小数，取值范围为（正或负）4.9E-324～1.8E308。

注意，浮点数在写法上用 F 作为后缀以便与类型 double 的值相区别。

每一个 Unicode 字符文字可以用'\uxxxx'的方式表示，其中 x 是任意的十六进制数字。例如，'B'的 Unicode 值为'\u0042'，而'π'为'\u03C0'。

除了数字值以外，浮点变量还可以有 3 个特殊的值——NEGATIVE_INFINITY、POSITIVE_INFINITY 和 NaN（表示非数字值）。这些特殊的值由非正常的算术操作所产生。

例 1.2　浮点类型中的特殊值

```
public class Ex0102
{ public static void main(String[] args)
  { double x=1E200;   //表示 1 后面跟 200 个 0
    System.out.println("x = " + x);
```

```
        System.out.println("x*x = " + x*x);
        System.out.println("(x*x)/x = " + (x*x)/x);
        System.out.println("(x*x)/(x*x) = " + (x*x)/(x*x));
    }
}
```

经编译后运行该程序，其输出将为：

```
x=1.0E200
x*x=Infinity
(x*x)/x=Infinity
(x*x)/(x*x)=NaN
```

在上例中，x*x 的值是 Infinity，这是由于 $10^{200}\times10^{200}=10^{400}$ 的原故，该值超出了 double 型值的最大值 1.8E308。

从代数角度上讲，(x*x) /x=x，但当 x 为无穷数时，情况并非如此，因为一个无穷数除以一个非负的有限数时仍然是无穷数，这就是 (x*x) /x=Infinity 的原因（此时，x*x 是一个无穷数，而 x 是一个有穷数）。

最后，一个无穷数除以一个无穷数甚至不是一个无穷数，这点从代数上讲是不确定的，因此，在 Java 语言中，其值为 NaN。

对 8 种基本类型中的每一种而言，Java 均提供了一个包装类 (wrapper class)，它为基本类型的操作提供面向对象的服务机制，例如，对 double 类型的包装类是 Double，类型 int 的包装类是 Integer。

例 1.3 包装类的使用

```
public class Ex0103
{ public static void main(String[] args)
  { String s = "2.7182818284590";
    System.out.println("s = " + s);
    Double x = new Double(s);
    System.out.println("x = " + x);
    double y = x.doubleValue();
    System.out.println("y = " + y);
    s += 45;
    System.out.println("s = " + s);
    y += 45;
    System.out.println("y = " + y);
    int n = x.intValue();
    System.out.println("n = " + n);
    n = Integer.parseInt("3A9",16);
    System.out.println("n = " + n);
  }
}
```

经编译后运行该程序，其输出将为：

```
s=2.7182818284590
x=2.718281828459
y=2.718281828459
s=2.718281828459045
y=47.718281828459
n=2
n=937
```

在上述程序中，构造器 Double(s)将从 String 对象 s 生成 Double 对象 x，将数值 2.718 281 828 459 0 放入对象 x 中。第二个 println()语句将隐含地调用 Double.toString()方法，它将数值转换为字符串用于打印输出。而 x.doubleValue()调用将返回所存储的数值，并把该值赋给 y。+=操作将应用到 String 对象 s 上，对 double 类型变量 y 也进行看似同样的操作，但实际产生的结果是截然不同的。对字符串而言，"+="代表的操作语义是“添加”，即把第二个字符串添加到第一个字符串上，而对数值而言，"+="代表“增加”，即把第二个数加到第一个上面。

接下来，Double.intValue()方法将对存储的数值进行截取，产生整数 2，并用该值对 n 进行初始化。最后， Integer.parseInt("3A9",16)将对十六进制的数值 3A9 进行转换，返回十进制的整数值。因为 $3(16^2)+10(16)+9=937$，所以返回值为 937。

应注意到对字符串和数字值 y 进行+操作时，将自动地把数值 y 转换为该数值的字符串表示，如系统调用 System.out.println("y="+ y)中的+操作就是如此。

1.5 流程控制

Java 支持 if 语句、if…else 语句、switch 语句以及条件表达式运算符"?:"。

例 1.4 条件语句的使用

下面的程序将首先产生 0 到 99 间的随机整数，然后，利用条件语句来对他们进行分类。

```
public class Ex0104
{ public static void main(String[] args)
  { int n = (int)Math.round(100*Math.random());
    System.out.println("n = " + n);
    if (n>25 && n<75) System.out.println(n + " is between 25 and 75");
    else System.out.println(n + " is not between 25 and 75");
    switch ((int)n/20)
    { case 0: System.out.println(n + " < 20");
      case 1: System.out.println(n + " < 40");
      case 2: System.out.println(n + " < 60"); break;
      case 3: System.out.println(n + " < 80"); break;
```

```
        default: System.out.println(n + " >= 80");
      }
      System.out.println(n + (n%2>0 ? " is odd" : " is even"));
    }
  }
```

该程序的输出为：

```
n=19
19 is not between 25 and 75
19<20
19<40
19<60
19 is odd
```

注意，在case为0和case为1的两个条件语句中都省略了break语句，这样，程序将顺序执行，一直到case为2的条件语句为止。在结果中，我们可以看到这3个输出语句的输出结果。对语句n%2 > 0 ? " is odd" : " is even"而言，表达的语义是：当条件部分n%2 > 0为真时（也就是n不能为2所整除），该语句的返回结果为" is odd"，否则返回结果为" is even"。

Java支持while语句、do..while语句以及for循环语句。

例1.5 循环语句的使用

下面的程序将从经验和尝试的角度来验证高斯（Guass）的素数定理，该定理可表述为：如p(n)表示小于n的素数的个数，则比值p(n)(ln n)/n随n的增大而趋向于1.0。

在程序中，奇数n的取值范围从3到1 000 000，对每个n，将计算出p(n)。随着p(n)的增加，我们在它达到5 000的倍数时进行输出，打印出相应的n、p(n)、ln n(自然对数)以及比值p(n)(ln n)/n，可以看到，它趋向于1.0。

```
class Ex0105
{ public static void main(String[] args)
  { System.out.println("n\tp(n)\tln(n)\t\t\tp(n)*ln(n)/n");
    final String DASHES18="\t------------------";
    System.out.println("------\t-----" + DASHES18 + DASHES18);
    int p=1;  // p 为小于 n 的素数的个数
    for (int n=3; n<1000000; n += 2)
    { int d=3;
      while (d<=Math.sqrt(n) && n%d>0)
        d += 2;
      if (n%d==0) continue;
      ++p;
      if (p%5000>0) continue;
```

```
      double ln=Math.log(n);
      System.out.println(n + "\t" + p + "\t" + ln + "\t" + p*ln/n);
    }
    System.out.println("------\t-----" + DASHES18 + DASHES18);
  }
}
```

```
n          p(n)       ln(n)                P(n)*ln(n)/n
-----      ----       -----------------    -----------------
48619      5000       10.79176967999097    1.109830486023054
104743     10000      11.559265009775785   1.1035835339617717
163847     15000      12.006688344529985   1.0991981859170432
224743     20000      12.322712806131365   1.0966048158235286
287137     25000      12.567714732761953   1.0942263390613152
350381     30000      12.766776412829921   1.093105198012728
414991     35000      12.936012112230687   1.091012633835611
479939     40000      13.08141429147497    1.0902564110418174
545749     45000      13.209914442069696   1.0892299388420983
611957     50000      13.324417297588104   1.0886726761511107
679279     55000      13.428787220525182   1.087304770394617
746777     60000      13.52352189210366    1.0865510232990836
814309     65000      13.610095179831822   1.0863888114819662
882389     70000      13.690388280841916   1.0860597533048737
951193     75000      13.765472265206315   1.085384795609801
-----      -----      -----------------    -----------------
```

在程序中，for 循环重复执行了 499 999 次，分别对应 n=3,5,7,9,…,999 999 中的每一个值。循环体对 n 进行测试，看能否被 d=3,5,7,9,…,$\sqrt{n}$ 等数所整除，如果能整除（即 n%d 的余数为零），则第一个 continue 语句将被执行并重复执行下一个 for 循环。当 n 不能为上述各数整除时，表明 n 是一个素数，这样计数器 p 加 1。

Java 支持子程序，这些子程序称为方法。

例 1.6 方法的使用

下面这个程序与例 1.5 的输出结果是一样的，但它使用了一个单独的方法来判断某个整数 n 是否为素数。

```
public class Ex0106
{ public static void main(String[] args)
  { System.out.println("n\tp(n)\tln(n)\t\t\tp(n)*ln(n)/n");
    final String DASHES18="\t-----------------";
    System.out.println("------\t-----" + DASHES18 + DASHES18);
    int p=1;  // p 为小于 n 的素数的个数
    for (int n=3; n<1000000; n += 2)
      if (isPrime(n))
```

```
    { ++p;
      if (p%5000>0) continue;
      double ln=Math.log(n);
      System.out.println(n + "\t" + p + "\t" + ln + "\t" + p*ln/n);
    }
  System.out.println("------\t-----" + DASHES18 + DASHES18);
 }

 private static boolean isPrime(int n)
 { int d=3;
  while (d<=Math.sqrt(n) && n%d>0)
    d += 2;
  if (n%d==0) return false;
  return true;
 }
}
```

程序中的 isPrime( )方法封装了判断 n 是否为素数的代码。当 n 能被某个整数所整除时，就返回 false，表明它不是素数。反之，找不到能整除 n 的整数时，返回 true，表明它是一个素数。

值得指出的是，该方法被声明为 private static，这是有特别的含义的，声明为 private 意味着它将不为该类之外的其他类所使用，而说明为 static 则是由于它将被同样是 static 的 main( )方法所调用。

1.6 类

一个类是一个抽象数据类型的实现。对象是通过对类进行实例化所产生的，在类定义中指明了该类的对象所能保持的数据种类和对象所能完成的操作，这些描述统称为类的成员。

如图1.3 所示，一个类有 4 种成员，分别称为 field、method、inner class 和 interface。field 指存储数据的变量，method 则是完成操作的函数，在类中，可以定义只用于该类的所谓 inner class 和 interface。图 1.3 中还列出了 4 种常用的特定方法 constructor、accessor、mutator 和 utility。

一个 constructor 是用于建立该类对象的方法，这一过程也称为实例化，所产生的对象称为该类的实例，该方法的另一个作用就是对所创建的对象中的各个 field 进行初始化。

一个 accessor 方法是一个只读函数，它返回类中某个 field 的值，这样就避免外部方法对 field 的直接修改。在 Java 语言中，访问某个 field X 的存取方法常取名为 getX()，正因为如此，Java 中的存取方法也常称为“getty”方法。

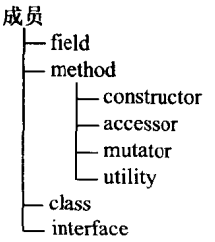

图 1.3 类的成员