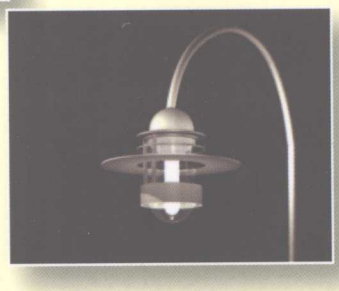
普通高等教育“十一五”国家级规划教材

高职高专计算机系列

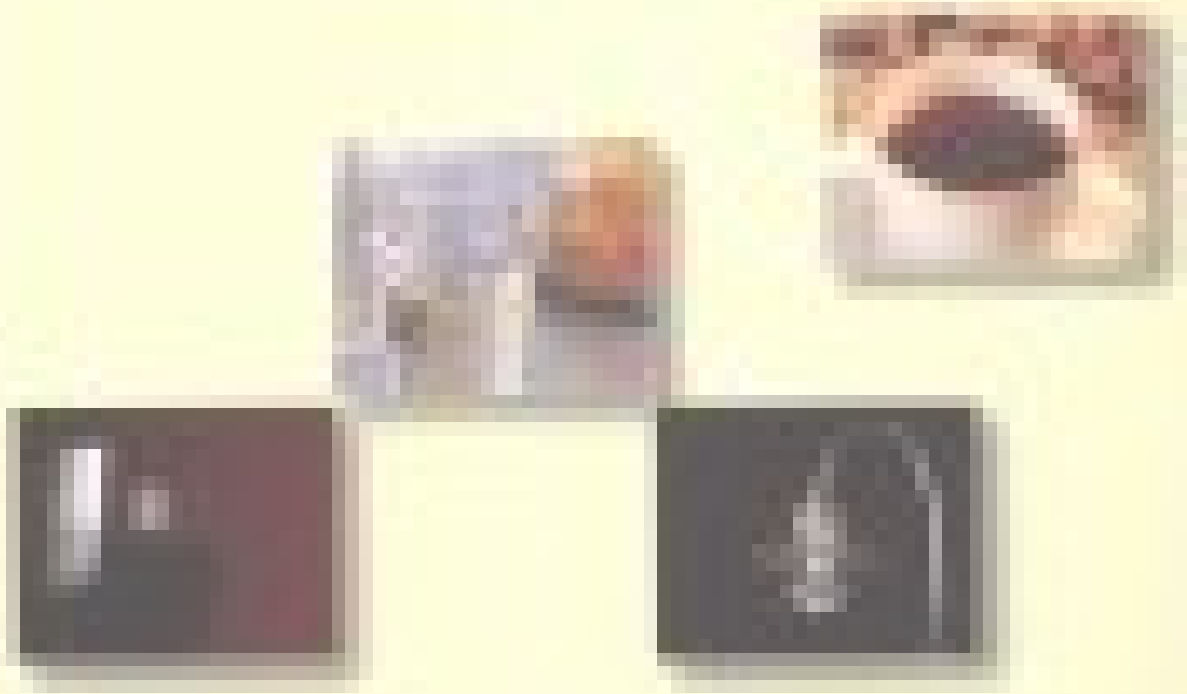
Java开发技术

魏勇 编著



人民邮电出版社
POSTS & TELECOM PRESS

Java开发技术



普通高等教育“十一五”国家级规划教材

高职高专计算机系列

Java 开发技术

魏 勇 编著

人民邮电出版社
POSTS & TELECOM PRESS
北 京

图书在版编目（CIP）数据

Java 开发技术 / 魏勇编著. —北京：人民邮电出版社，2008.5
普通高等教育“十一五”国家级规划教材. 高职高专计算机系列
ISBN 978-7-115-17627-1

Ⅰ. J… Ⅱ. 魏… 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（2008）第 018425 号

内 容 提 要

在应用 Java 开发的实际过程中，程序员很少碰到只涉及 Java 语言本身的问题。本教材主要针对中级 Java 程序员，所以要求读者已经掌握 Java 语言的基本知识。

本教材从 Java 开发经常涉及的几个主要技术展开讨论，内容包括 Java 的流技术、线程、网络通信、JDBC 技术、Web 编程、Struts 框架、分布式编程、EJB、Java 数据结构等。本教材配有大量实例，实例中的程序都通过调试，因而读者在进一步验证时，不会出现不必要的困惑。为便于理解，本教材有 3 个典型的实例贯穿在各章节，它们是 Hello World、用户登录、生产者和消费者实例。

本教材可作为高职院校计算机专业相关课程的教材，也可供各类社会培训机构选用，还可以供软件开发人员自学参考。

普通高等教育“十一五”国家级规划教材
高职高专计算机系列

Java 开发技术

◆ 编　　著　魏　勇
策划编辑　潘春燕
责任编辑　李　凯

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：18.5
字数：446 千字　　2008 年 5 月第 1 版
印数：1－3 000 册　　2008 年 5 月北京第 1 次印刷

ISBN 978-7-115-17627-1/TP

定价：29.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

Java不依赖任何操作系统，而且Java的类库提供了很完善的功能。在应用Java进行实际开发过程中，程序员很少碰到只涉及Java语言本身的问题。本教材主要针对中级Java程序员，所以要求读者已经掌握Java语言的基本知识。

相比其他语言来说，我们可以归纳出Java的很多优点。Java问世十几年来，已经得到广泛应用。Java技术发展很快，关于Java的技术资料也很多，但国内许多Java技术书籍大多面向资深开发人员，不适合作为高职院校的教材。本教材将学习目标从“知道不知道”提升到“会不会应用”。本教材重视实践环节，安排了大量实例。

高等职业院校计算机相关专业大都开设Java课程，但仅把Java作为一门语言来教学是远远不够的。为了适应各种商业应用和分布式处理的需要，Sun Microsystem公司在Java 1.0的基础上推出了J2EE技术（亦即Java 2.0 Enterprise Edition），提供了Java企业应用编程接口（Java Enterprise API），为企业计算以及电子商务应用系统提供了有关的技术和强大的类库支持。Java企业应用编程接口包括JDBC（Java Database Connectivity）、EJB（Enterprise Java Beans）、Java RMI（Java Remotie Method Invocation）、Java IDL（Java Interface Definition Language）、JNDI（Java Naming and Directory Interface）、JMAPI（Java Management API）、JMS（Java Message Service）、JTS（Java Transaction Service）、Java Servlets and Java Server Pages（JSPs）、JavaMail、Connectors、XML（eXtensible Markup Language）等。

Java涉及的技术很广，教材不能像技术手册一样面面俱到。本教材从Java开发经常涉及的几个主要技术展开讨论，内容包括Java的流技术、线程、网络通信、JDBC技术、Web编程、Struts框架、分布式编程、EJB、Java数据结构等。

本教材具有以下一些特点。

❑ 丰富的实例　对于读者、特别是初学者而言，如果教材仅对技术进行陈述，往往容易使读者感到迷惑。为便于理解，在教材中给出实例并加以说明是最好的途径之一。本教材有诸多实例贯穿于各个章节，并且每一个实例程序都通过调试，因

而读者在进一步验证时，不会出现不必要的困惑。

关于 Java 的各项技术相对较独立，为便于读者理解，本教材用 3 个典型的实例贯穿在各章节，它们是 Hello World、用户登录、生产者和消费者实例。

❑ 合理的结构　本教材符合读者循序渐进、由浅入深的学习习惯，上手应用快，学习效果好。

❑ 简练流畅的语言　本教材语言简练，重点突出。

❑ 典型的习题　本教材大部分章节都附有代表性较强的习题，以帮助读者巩固知识。

❑ 涉及工具多　本教材讲述 Java 的多项技术，除 JDK 外，还包括以下一些工具。

- eclipse-SDK-3.2.1-win32.zip
- hibernate-3.1.3.zip
- jakarta-struts-current.zip
- jakarta-tomcat-5.0.12.zip
- jboss-3.2.6.zip
- jdk-1_5_0_08-windows-i586-p.exe
- MyEclipse_5.5.1GA_E3.2.2_installer.exe
- mysql-4.0.14b-win.zip
- mysql.jar
- mysql-connector-java-3.0.16-ga-bin.jar

为便于教学，本教材配有丰富的教学资源，包括 PDF 电子教案、edu.mdb、一个 Hibernate 配置目录、习题分析解答、各章的源程序清单、本地方法和 JavaMail 技术讲义等，读者可到人民邮电出版社网站 www.ptpress.com.cn 下载。

本教材的编写工作得到了深圳信息职业技术学院软件工程系的大力支持，在此表示衷心的感谢！书中难免有不妥之处，恳请各位专家和读者批评指正。同时也欢迎读者与作者交流，作者的联系方式如下。

E-mail：weiuser@hotmail.com

电话：0755-25859280

编著者

2008 年 3 月

目　录

第 1 章 线 程

- 线程的概念
- 线程的实现
- 线程组
- 线程的同步与交互

同进程一样，线程的引入也是为了实现并行处理，从而提高系统的效率。线程是一个程序内部的一个单一的顺序控制流。本章首先介绍线程的基本概念，然后介绍如何在 Java 中编写线程程序，以及多线程的程序设计等。

1.1 线程的概念

在传统上，并发多任务的实现采用的是在操作系统（OS）级运行多个进程。进程是在计算机上运行的可执行文件针对特定的输入数据的一个实例，同一个可执行程序文件如果操作不同的输入数据就是两个不同的进程。

一般说来，线程指程序中的一个单一的顺序控制流。线程是进程的一条执行路径，它包含独立的堆栈和 CPU 寄存器状态，每个线程共享其所附属的进程的所有的资源，包括打开的文件、页表（因此也就共享整个用户态地址空间）、信号标识及动态分配的内存等。多线程意味着一个程序的多行语句同时执行。但多线程并不等价于多次启动一个程序，操作系统也不把每个线程当作独立的进程来对待。

虽然线程与进程都是顺序执行的指令序列，但线程与进程不同。对于进程来说，由于各个进程，包括子进程与父进程之间都拥有自己独立的代码和数据空间，进程间的耦合关系差，并发实现也不太容易；多个线程则共享数据空间，同时每个线程都有自己的执行堆栈和程序计数器，可以避免进行无谓的数据复制。

所谓多线程程序设计，是指使单个程序中包含并发执行的多个线程。当多线程程序执行时，该程序对应的进程中就有多个控制流在同时运行，即具有并发执行的多个线程。在一个进程中包含并发执行的多个控制流，而不是把多个控制流一一分散在多个进程中，这是多线程程序设计与多进程程序设计的截然不同之处。

正是由于线程与进程之间的这些差别，决定了多线程技术比起用进程来实现并行处理，有切换速度快、通信容易实现等优越性。

Java 通过线程有效地实现了多个任务的并发执行。在一个 Java 程序的执行过程中，通常总是有许多线程在运行。

1.2 线程的实现

1.2.1 线程体

Java 中的线程实现非常简单，可以用两种方式来创建线程，一种是实现 Runnable 接口，另一种是继承 Thread 类重写 run()方法。两种方式唯一的不同就是前者保留了继承一个类的可能（因为 Java 只支持类的单继承，但接口没有此限制）。

线程类中的 run()可以被直接调用，但决不是启动一个线程，两者有着本质的区别。每个线程都是通过特定的方法 run()来完成其操作的，方法 run()称为线程体。线程的启动需要用 start()方法来来实现，下面是代码示例。

```
:
public class Mythread extends Thread{
    public void run(){
    //here is where you do something
    }
}

public class Mythread implements Runnable{
    public void run(){
    //here is where you do something
    }
}
:
```

这两种方法的区别是，如果某类需要继承其他的类，如 Applet，那么只能选择实现 Runnable 接口，因为 Java 只允许单继承。

启动一个线程只需要调用 start()方法，针对两种实现线程的方法也有两种启动线程的方法。

下面的程序产生两个线程，start()方法只是让线程处于就绪状态，系统会给每一个处于就绪状态的线程一个时间片来运行，所以这两个线程交替运行，而且每次启动程序，交替运行的顺序也不一定一致，完全取决于当时系统的调度情况。

例 1-1 该实例由两个类 MyThread 和 TestThread 构成，MyThread 继承 Thread，并覆盖了它的 run()方法。该方法是线程真正运行的部分。在该方法中，循环 10 次，不断地判断当前正在运行的线程，并输出相应结果。

下面是类 MyThread，它继承了类 Thread，Thread 类的构造方法 Thread（String str）可为线程命名。

MyThread.java

```
import java.awt.*;
public class MyThread extends Thread
{
  public MyThread(String str) {
    super(str);
```

```
    }
    public void run() {
      for(int i=0;i<10;i++)
      {
        try{
          sleep(1);
        }catch(InterruptedException e){}
        if(getName().equals("thread1")) {
          System.out.println("thread1"+"is runing");
        }
        else {System.out.println("thread2"+"is runing");}
      }
    }
}
```

下面的类 TestThread 通过 main()方法产生两个名为 thread1 和 thread2 的线程，并启动它们，使其处于就绪状态，交由系统调度运行。

TestThread.java

```
public class TestThread {
  public static void main(String args[]){
    MyThread t1=new MyThread("thread1");
    MyThread t2=new MyThread("thread2");
    t1.start();
    t2.start();
    System.out.println("*****the programe is running*******");
  }
}
```

下面是两组程序实际的运行结果：

```
C:\J2SDK1～1.2\bin>javaTestThread
*****theprogrameisrunning*******
thread1 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread2 is running
thread1 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread2 is running
```

```
C:\J2SDK1～1.2\bin>javaTestThread
*****theprogrameisrunning*******
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
thread1 is running
thread2 is running
```

我们注意到，两次启动程序后，线程交替运行的顺序是不一样的，这取决于当时系统的调度情况。

另外，类 TestThread 中的语句 *System.out.println("*****the programe is running *******");* 尽管在两个线程启动后，但却比这两个线程先执行。这说明用 start()启动线程只是让线程处于就绪状态，是否真正运行则通过系统用时间片来调度。

1.2.2 线程的状态

线程是程序中单一的顺序控制流，具有生命周期，即它由创建而产生，由撤销而消亡。在线程的生命周期中，它总是从一种状态变迁到另一种状态。状态表示线程正在进行的活动以及在这段时间内线程能完成的任务。

图 1-1 所示为一个 Java 线程所具有的不同状态及各状态间进行变迁所需调用的方法。

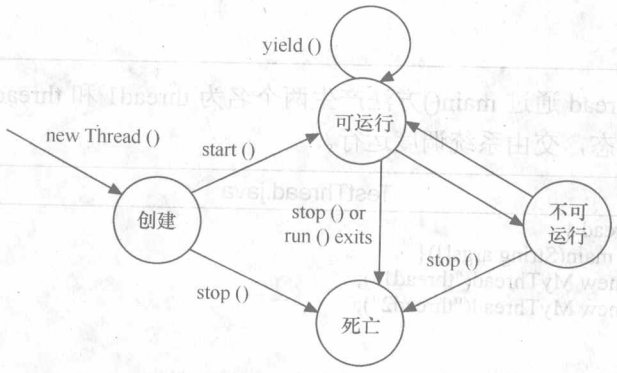


图 1-1 线程的状态及其转换

1. 创建（newThread）状态

当生成一个线程对象并对它进行了初始化之后，该线程处于创建状态，它仅仅是一个空的线程对象，系统没有为它分配 CPU 资源。

处于这种状态的线程只能被启动或被终止，调用除此以外的其他方法都会失败并且会引起非法状态处理。

2. 可运行（Runnable）状态

线程通过方法 start()来启动，以进入可运行状态，等待被调度执行。

start()方法产生了运行这个线程所需的系统资源，安排其运行，并调用线程体 run()方法，这样就使得该线程处于可运行(Runnable)状态。

这一状态不是运行中（Running）状态，因为线程也许实际上并未真正运行。特别对于单处理器的计算机，要在同一时刻运行所有的处于可运行状态的线程是不可能的，Java 的运行系统必须实现调度来保证这些线程共享处理器。但是在大多数情况下，可运行状态也就是运行中，当一个线程正在运行时，它是可运行的，并且也是当前正运行的线程。

yield()方法使运行从当前的线程切换到下一个可运行的线程。这是一种使低优先级线程不至于空等的方法。

3. 不可运行（NotRunnable）状态

当以下 4 种情况发生时，线程就进入不可运行状态。

（1）sleep()方法：当线程被调度执行进入运行状态时，可以调用其方法 sleep()暂停它的执行，方法 sleep()每次只睡眠一段确定的时间。

（2）suspend()方法：方法 suspend()暂时停止线程的工作，然后用方法 resume()来使它重新工作。

（3）为等候一个条件变量，线程调用 wait()方法。

（4）输入输出流中发生线程阻塞。

从不可运行状态返回可运行状态也分以下 4 种情况。

（1）如果线程处于睡眠状态中，sleep()方法中的参数为休息时间，当这个时间过去后，线程即为可运行的。

（2）如果一个线程被挂起，必须调用 resume()方法才能返回。

（3）如果线程在等待条件变量，那么要停止等待的话，需要该条件变量所在的对象调用 notifyAll()方法。

（4）如果在 I/O 流中发生线程阻塞，则特定的 I/O 指令将结束这种不可运行状态。

4. 死亡（Dead）状态

（1）从线程的 run()方法正常退出，进入死亡状态。

（2）调用 stop()方法也可停止当前线程。

在 Thread 类的程序接口中提供了 isAlive()方法，如果线程已被启动并且未被终止，那么 isAlive()返回 true。若返回 true，则该线程是可运行或是不可运行的；如果返回 false，则该线程是新创建或是已被终止的。

1.2.3 线程的调度

多个线程同时处于可执行状态并等待获得 CPU 时间时，线程调度系统根据各个线程的优先级来决定给谁分配 CPU 时间。具有高优先级的线程会在较低优先级的线程之前得到执行。线程的调度是抢先式的，如果在当前线程的执行过程中，一个具有更高优先级的线程进入就绪状态，则这个高优先级的线程立即被调度执行。对优先级相同的线程来说，调度将采用轮转法。

线程的优先级用 1～10 这 10 个数字表示，数字越大表明线程的级别越高。1 和 10 可分别用 Thread.MIN_PRIORITY，Thread.MAX_PRIORITY 来表示。线程的默认优先级是 5，即 Thread.NORM_PRIORITY。可以用 Thread 类的方法 getPriority()和 setPriority()来存取线程的优先级。例如：

```
int getPriority();
void setPriority(intnewPriority);
```

例 1-2　生成两个线程，第一线程先启动，但优先级为 Thread.MIN_PRIORITY；第二个线程后启动，但优先级为 Thread.MAX_PRIORITY。我们会发现优先级高的线程完成后再运行优先级低的线程。

MyThread 继承 Thread，run()方法是一个循环，输出当前线程名及线程的优先级别。

MyThread.java

```
class MyThread extends Thread{
  MyThread(String str){
    super(str);
  }
  public void run(){
    for(int i=0;i<3;i++)
    System.out.println(getName()+""+getPriority());
  }
}
```

TestThread 的 main()方法生成两个线程，并设置它们的优先级别。

TestThread.java

```
class TestThread{
  public static void main(Stringargs[]){
    Thread first=new MyThread("thread1");
    first.setPriority(Thread.MIN_PRIORITY);
    first.start();
    Thread second=new MyThread("thread2");
    second.setPriority(Thread.MAX_PRIORITY);
    second.start();
  }
}
```

运行结果为

```
C:\j2sdk1.4.2\bin>javaTestThread
thread2 10
thread2 10
thread2 10
thread1 1
thread1 1
thread1 1
```

从运行结果来看，线程 first 尽管先启动，但由于优先级比 second 低，所以线程 second 执行结束后，再开始运行线程 thread2。

1.2.4 Daemon 线程

Daemon 线程是一类特殊的线程，通常在一个较低的优先级上运行。Daemon 线程一般被用于为系统中的其他对象和线程提供服务。典型的 Daemon 线程是 JVM 中的系统资源自动回收线程，它始终在低级别的状态中运行，用于实时监控和管理系统中的可回收资源。通常 Daemon 线程体是一个无限循环以等待服务请求。当系统中只剩下 Daemon 线程在运行时，Java 解释器将退出，因为这时没有其他线程需要提供服务。

可以通过调用方法 isDaemon()来判断一个线程是否是 Daemon，也可以调用方法 setDaemon()来将一个线程设为 Daemon 线程。

下面是一个 Daemon 线程的框架。

```
class DaemonThread extendsThread{
  DaemonThread(){
    setDaemon(true);
    start();
  }

  public void run(){
    while(true){
    //wait for service request and process
    }
  }
}
```

在上面的例子中 DaemonThread 线程在创建时设置了 Daemon 标志。如果程序结束时还有 Daemon 线程存在，Java 将会消灭那些线程，因此不必担心它们是否消亡。