Core Java

高级应用程序设计教程

刘甲耀　严桂兰　编著

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

Core Java

高级应用程序设计教程

刘甲耀　严桂兰　编著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内容简介

本书是《Core Java 应用程序设计教程》的续篇，主要阐述高级应用程序设计的方法与技巧（含 Applet 以及 Applet 与 Application 两者的组合应用），其内容取材广泛，由浅入深，涉及：基本 Java Applet；图形、图像与动画设计；事件处理；图形用户界面构件；网络通信与声音文件播放。本书所有示例均在 Core Java 2（使用 TextPad 工具）环境中通过，实用性强，覆盖面广。许多例子采用多种解决方案，充分体现了 Core Java 编程的灵活性与多样性。每章均有小结与习题。书末附录提供了 TextPad 与 JDK 的使用步骤，以及 Core Java 的安装步骤。书中示例、习题与运行结果可通过华信教育资源网（http://www.hxedu.com.cn）免费下载使用。本书可作大专科院校计算机和其他各类专业及培训班的教科书，并可供各行各业从事计算机工作人员使用。

图书在版编目(CIP)数据

Core Java 高级应用程序设计教程/刘甲耀，严桂兰编著. —北京：电子工业出版社，2006.3
ISBN 7-121-02228-1

Ⅰ.C... Ⅱ.①刘... ②严... Ⅲ.JAVA 语言－程序设计－教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2006）第 004053 号

责任编辑：龚立堇
印　　刷：北京市李史山胶印厂
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
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：24.5 字数：625.6 千字
印　　次：2006 年 3 月第 1 次印刷
印　　数：3 000 册　　定价：31.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。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。

前　言

Core Java 是基于网络的纯面向对象编程语言，适用于在各种平台与操作系统上编写各式各样的软件，编译后的代码能在互联网上传递，并确保用户安全运行，因而是当前最富生命力的计算机编程语言之一。

Core Java 除了包含 Java 的所有功能之外，其最大的特点之一是数据的输入与输出（特别是格式化输出）特别简单。就输入一个数据而论，如果使用标准 Java 至少需要四条语句才能实现，而 Core Java 则只要一条语句就能完成。对数据格式化输出来说，使用标准 Java 非常麻烦，而使用 Core Java 则像 C 语言一样容易。

为适应当前 Internet 的迅猛发展及各行各业学习 Core Java 的需要，特别是大专院校为研究生、本科生甚至专科生开设面向对象程序设计课程的需要，我们根据多年对 Java 和 Core Java 教学与科研的实践，以及 Java 版本的升级，并根据 Core Java 能创建应用程序（Application）和小应用程序（Applet）两大特点，以及为了教学叙述的方便，作者特分为《Core Java 应用程序设计教程》和《Core Java 高级应用程序设计教程》两册来阐述。《Core Java 高级应用程序设计教程》主要阐述 Core Java 高级应用程序设计方法与技巧，内容涉及：基本 Java Applet；图形、图像与动画设计；事件处理；图形用户界面构件；网络通信与声音文件播放。本书每章均有小结与习题。本书所有示例均在 Core Java 2 环境下（使用 TestPad 工具）通过，并在附录中提供了 TextPad 与 JDK 的使用步骤及 Core Java 的安装步骤。

本书有以下三大特点：

1．开发工具与语言相结合。本书使用了最新版本 Core Java 2 及 Textpad 工具。

2．取材广泛，由浅入深，重点、难点分明，易学易掌握。

3．编程方法与示例并举。通过一例多解的方式说明 Core Java 编程的灵活性、多样性、实用性与趣味性。

本书中示例、习题与运行结果可通过华信教育资源网（http://www.hxedu.com.cn）免费下载使用。

在本书编写中，承蒙美国某公司 CEO 刘涌博士提供了大量资料，广州私立华联学院庾锦辉、郑小川、陈凤娇、刘桂桥等参与了本书工作，在此表示感谢。

本书不足之处，敬请读者指正。

作者 E-mail 地址：ygl0501@sina.com.cn

作　者

2005 年 12 月

目　录

第 1 章　基本 Java Applet

本章介绍基本 Java Applet，内容涉及：Applet 的创建与组织；在 Web 页中包含 Applet；向 Applet 传递参数。

1.1　Applet 的创建与组织

1.1.1　Applet 的创建

1．Applet 的工作与特点

一般说来，当用户访问 applet（小应用程序）的 Web 页时，applet 就工作。另外，浏览器创建 HTML 文件，该文件是浏览器的拷贝类文件，它由<APPLET>标记所指定的类文件的局部拷贝而来，它也是语法分析文件，如果浏览器找到任何的 import 语句，而且这些类不是在本地站上，它也拷贝这些文件，一旦拷贝了文件，就装入 applet，并且浏览器创建初始类的实例（由<APPLET>标记指定）。注意，浏览器和 applet 执行它创建的类实例的方法，而不执行初始的方法。然后，由浏览器运行类的实例，而且浏览器自动地执行 init() 和 start()方法，类中的其余方法则由 applet 执行，applet 处理过程和 application（独立应用程序）处理过程是不同的。

2．Applet 的创建

为创建一个 applet，根据 java.applet 软件包创建 Applet 类的子类。Applet 类提供 applet 在浏览器内工作，当包含处理鼠标和键盘事件的 UI 组件时，还可用抽象窗口工具（AWT）的能力在屏幕上画图。虽然，applet 可能具有所需的许多 helper 类（包含启动 applet 的主方法），但主 applet 类是触发 applet 执行的类。注意，AWT 是抽象窗口工具和容器，它是一个类组，当程序要求图形用户界面组件（例如，按钮、列表、滚动条、窗口与框架）时就要使用它。

3．创建 Applet 的基本模式

初始的 main applet 类总要有如下的说明：

```
public class 类名 extends java.applet.Applet
{
  ...
}
```

或

```
import java.applet.Applet;
public class 类名 extends Applet
{
```

```
    ...
}
```

或

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class 类名 extends JApplet
{
  ...
}
```

其中，类名由用户指定的合法的标识符构成。

注意：Java 要求将 applet 子类说明为 public，这只有 main applet 类才是正确的，用户创建的任何辅助类可以是 public，也可以是 private。

1.1.2 Applet 主要的组织

由于 applet 需要依靠浏览器来操作，在 applet 的生命周期，许多不同的动作绘画均对应于各种各样的事件。例如，初始化、绘画及鼠标动作，它们的每个动作都有一个对应的方法。因而当事件发生时，applet 就依靠浏览器来调用事件特有的方法。

为提供事件的行为，必须在 applet 的子类中重构（抑制）每个方法，并给出所要求的特定行为。由于 Java 是独立于平台的，不需要重构所有的事件，只需重构所需的事件。所有的 applet 均是 Applet 类的子类。Applet 类具有处理事件的方法，如果需要 Applet 类做任何事，必须创建触发每个事件的成员函数。这样做将重载 applet 中超类的方法，在其方法的每个顶部加上 applet 的方法。有五个主要的 applet 执行方法：初始化、启动、停止、撤销、绘画。

1. 初始化

当 applet 首先装载（或重载）时就发生初始化。初始化可以包括创建必须的对象，设置初始化状态，装载图像或字体，以及设置参数。为提供初始化工作，可重构 init()方法：

```
public void init()
{
  ...
}
```

2. 启动

在 applet 初始化之后就启动 applet。如果 applet 被停止，也可再次启动。例如，当用户连接不同的页面时，applet 会停止，而当用户返回到本页时，applet 就重新启动，即使在 applet 生命周期，也可能多次启动，但初始化只有一次。在 start()方法中可以包含的功能有：启动 applet 控制线程，向 helper 对象发送信息或者仅通知 applet 开始运行。为提供 applet 启动行为，可重构 start()方法：

```
public void start()
{
  ...
}
```

3．停止

停止和启动相互结合进行，当用户触发一个事件信号时就离开包含 applet 当前所在的运行页，转到其他页面时 applet 就停止运行。这时，自动地调用由浏览器安置的 stop()方法，stop()方法就像 MP3 和收录机上的“暂停”按钮，它的作用就是挂起 applet 的执行。当浏览器离开 Web 页时，就启动其他线程的 applet，虽然原有的 applet 将继续进行，但不予完成。通过重构 stop()，可以挂起线程的执行。若用户再次返回 applet 所在的 Web 页时，applet 将要执行 start()方法恢复运行。可通过调用 stop()方法来停止 applet：

```
public void stop()
{
  …
}
```

4．撤销

撤销能使 applet 终止执行，并在 applet 从内存中释放或在浏览器退出之前，进行一些善后处理工作，例如撤销任何终止运行的线程和释放任何其他运行的对象。一般说来，不需要重构 destroy()，除非需要释放特定的资源。例如，创建了线程 applet，为提供 applet 清理行为，可重构 destroy()方法：

```
public void destroy()
{
  …
}
```

destroy()与 finalize()清理的对象是不同的：在一个 applet 的一个类内使用 finalize()，则立即清理所有的对象；而用 destroy()，则在 applet 退出之前清理当前对象。

5．绘画

绘画是指 applet 在屏幕上画一些文本、线、颜色背景或图像。在 applet 生命周期，绘画可以出现多次。例如，绘画可以出现在初始化 applet 之后；在浏览器窗口恢复到前景时；由覆盖窗口放入背景之后；当浏览器窗口移至不同的位置或者重新移至不同位置时；当 Web 页包含一个动画时。paint()方法形式为：

```
public void paint(Graphics g)
{
  …
}
```

与本节的其他方法不同，paint()使用参数：Graphics 类和 context 的一个实例。创建 Graghics 对象并通过浏览器自动地传递给 paint，必须确保 Graphics 类（java.awt 软件包部分）获得输入 applet 代码。通常在 applet 文件头部使用下列 import 语句来实现：

```
import java.awt.Graghics;
```

也可用

```
import java.awt.*;
```

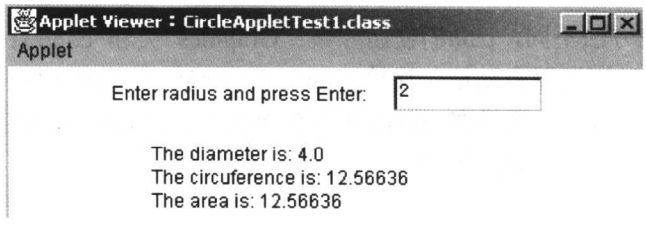
来代替，其中*表示引用 java.awt 包中本程序用到的所有类，Graghics 类当然属于其中。

【例 1.1】读入一个圆的半径，求圆的直径、周长与面积。

方案一：（用 paint，action 方法实现）

```
//CircleAppletTest1.java
import java.awt.Graghics;              //也可用 import java.awt.*;
import java.applet.Applet;
public class CircleAppletTest1 extends Applet
{
  private Label prompt;
  private TextField input;
  private float Radius,Diameter,Circumference,Area;
  private static final float PI=3.14159f;
  public void init()
  {
    prompt=new Label("Enter radius and press Enter:");
    input=new TextField(10);
    add(prompt);
    add(input);
  }
  public void paint(Graphics g)
  {
    try
    {
      String s=input.getText();
      Radius=Float.parseFloat(s);
      //或 Radius=new Float(s).floatValue();
      Diameter=2*Radius;
      Circumference=2*PI*Radius;
      Area=PI*Radius*Radius;
      g.drawString("The diameter is: "+Diameter,90,60);
      g.drawString("The circuference is: "+Circumference,90,75);
      g.drawString("The area is: "+Area,90,90);
    }
    catch(NumberFormatException e)
    {
    }
  }
  public boolean action(Event e,Object o)
  {
    repaint();
    return true;
  }
}
```

运行结果：

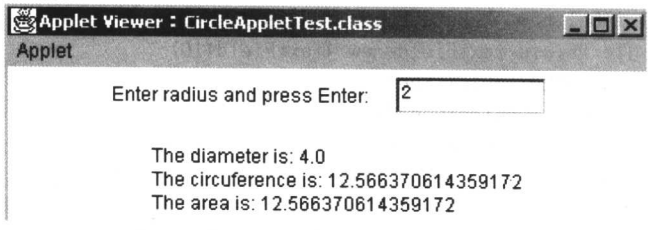


方案二：（用 action 方法实现）

```
//CircleAppletTest.java
```

```
import java.awt.*;
import java.applet.*;
import java.util.*;
public class CircleAppletTest extends Applet
{
  Label prompt;
  TextField input;
  double Radius,Diameter,Circumference,Area;
  public void init()
  {
    prompt=new Label("Enter radius and press Enter:");
    input=new TextField(10);
    add(prompt);
    add(input);
  }
  public boolean action(Event e,Object o)
  {
    Graphics g=getGraphics();
    String s=input.getText();
    Radius=Double.parseDouble(s);
    //或 Radius=new Double(s).doubleValue();
    //或 Radius=Double.valueOf(o.toString()).doubleValue();
    Diameter=2*Radius;
    Circumference=2*Math.PI*Radius;
    Area=Math.PI*Radius*Radius;
    update(g);
    g.drawString("The diameter is: "+Diameter,90,60);
    g.drawString("The circuference is: "+Circumference,90,75);
    g.drawString("The area is: "+Area,90,90);
    return true;
  }
}
```

运行结果：



方案三：（用 implements ActionListener 实现）

```
//CircleAppletTest3.java
import java.awt.*;
import java.applet.*;
import javax.swing.*;
import java.awt.event.*;
public class CircleAppletTest3 extends Applet implements ActionListener
{
  private JLabel prompt;
  private JTextField input;
```

```
    public void init()
    {
      setLayout(new FlowLayout());
      prompt=new JLabel("Enter radius");
      input=new JTextField(10);
      add(prompt);
      add(input);
      input.addActionListener(this);
    }
    public void actionPerformed(ActionEvent evt)
    {
      Graphics g=getGraphics();
      String s=input.getText();
      double radius=Double.parseDouble(s);
      double Diameter=2*radius;
      double Circumference=2*Math.PI*radius;
      double Area=Math.PI*radius*radius;
      update(g);
      g.drawString("The diameter is: "+Diameter,100,60);
      g.drawString("The circuference is: "+Circumference,100,75);
      g.drawString("The area is: "+Area,100,90);
    }
  }
```

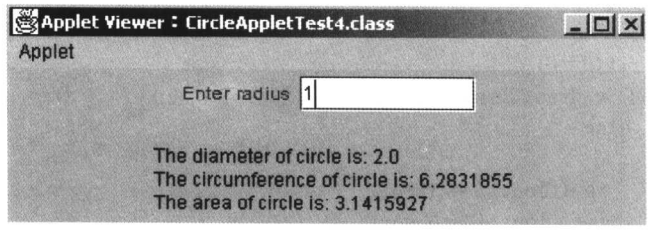
运行结果与方案二同。

方案四：（用 Japplet，actionPerformed 实现）

```
//CircleAppletTest4.java
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class CircleAppletTest4 extends JApplet
{
  public void init()
  {
    JLabel label=new JLabel("Enter radius");
    final JTextField textField=new JTextField(10);
    textField.setEditable(true);
    Container c=getContentPane();
    c.setLayout(new FlowLayout());
    c.add(label);
    c.add(textField);
    textField.addActionListener(new ActionListener()
    {
      public void actionPerformed(ActionEvent e)
      {
        String s=textField.getText();
        double radius=Double.parseDouble(s);
        double Diameter=2*radius;
        double Circumference=2*Math.PI*radius;
        double Area=Math.PI*radius*radius;
        Graphics g=getGraphics();
        update(g);
```

```
            g.drawString("The diameter of circle is:
              "+(float)Diameter,90,60);
            g.drawString("The circumference of circle is:
               "+(float)Circumference,90,75);
            g.drawString("The area of circle is: "+(float)Area,90,90);
          }
        });
    }
}
```

运行结果：



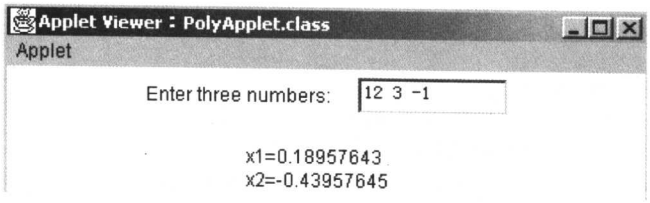
【例 1.2】求解一元二次方程的实根与复根。

方案一：（用 StringTokenizer 实现输入数据）

```
//PolyApplet.java
import java.applet.*;
import java.awt.*;
import java.util.*;
import java.security.*;
public class PolyApplet extends Applet
{
  Label prompt;
  TextField input;
  float a,b,c,d,x1,x2,re,im;
  public void init()
  {
    prompt=new Label("Enter three numbers:");
    input=new TextField(10);
    add(prompt);
    add(input);
  }
  public void paint(Graphics g)
  {
    try
    {
      String s=input.getText();
      StringTokenizer Line=new StringTokenizer(s," ");
      a=Float.parseFloat(Line.nextToken());
      //或 a=new Float(Line.nextToken()).floatValue();
      b=Float.parseFloat(Line.nextToken());
      //或 b=new Float(Line.nextToken()).floatValue();
      c=Float.parseFloat(Line.nextToken());
      //或 c=new Float(Line.nextToken()).floatValue();
      if(a==0)
        System.exit(0);
```

```
        else
        {
          d=b*b-4*a*c;
          if(d>0)
          {
            x1=(float)((-b+Math.sqrt(d))/(2*a));
            x2=(float)((-b-Math.sqrt(d))/(2*a));
            g.drawString("x1="+x1,150,60);
            g.drawString("x2="+x2,150,75);
          }
          else
          {
            if(d==0)
              g.drawString("x1=x2="+-b/(2*a),150,60);
            else
            {
              re=(float)(-b/(2*a));
              im=(float)(Math.sqrt(-d)/(2*a));
              g.drawString("re="+re,150,60);
              g.drawString("im="+im,150,75);
            }
          }
        }
        catch(NoSuchElementException e)
        {
        }
        catch(AccessControlException e1)
        {
          g.drawString("Exit loop!",150,100);
        }
      }
      public boolean action(Event e,Object o)
      {
        repaint();
        return true;
      }
    }
```

运行结果：



方案二：（用 split 实现输入数据）

```
//PolyApplet1.java
import java.applet.*;
import java.awt.*;
import java.util.*;
```

```
import java.security.*;
public class PolyApplet1 extends Applet
{
  TextField input;
  double a,b,c,d,x1,x2,re,im;
  public void init()
  {
    Label prompt=new Label("Enter three numbers:");
    input=new TextField(10);
    add(prompt);
    add(input);
  }
  public void paint(Graphics g)
  {
    try
    {
      String s=input.getText();
      String[] s1=s.split(" ");
      a=Double.parseDouble(s1[0]);
      b=Double.parseDouble(s1[1]);
      c=Double.parseDouble(s1[2]);
      if(a==0)
        System.exit(0);
      else
      {
        d=b*b-4*a*c;
        int i=d>0?1:d==0?2:3;
        switch(i)
        {
          case 1: x1=(-b+Math.sqrt(d))/(2*a);
                  x2=(-b-Math.sqrt(d))/(2*a);
                  g.drawString("x1="+(float)x1,150,60);
                  g.drawString("x2="+(float)x2,150,75);
                  break;
          case 2: g.drawString("x1=x2="+(float)(-b/(2*a)),150,60);
                  break;
          case 3: re=-b/(2*a);
                  im=Math.sqrt(-d)/(2*a);
                  g.drawString("re="+(float)re,150,60);
                  g.drawString("im="+(float)im,150,75);
                  break;
        }
      }
    }
    catch(NoSuchElementException e){}
    catch(ArrayIndexOuOfBoundsExceptio e1){}
    catch(AccessControlException e2)
    {
      g.drawString("Exit loop!",150,70);
    }
  }
```