Core Java

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 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

由于 Core Java 能创建应用程序（Applications）和小应用程序（Applets），同时也为了教学上的系统性与诱导性，作者特将 Core Java 分成为《Core Java 应用程序设计教程》和《Core Java 小应用程序设计教程》两册来阐述。本书为《Core Java 应用程序设计教程》，主要阐述 Core Java 应用程序设计的方法与技巧，其内容取材广泛，由浅入深，它涉及：基本 Core Java（含 Core Java 的特点与基本编程模式，基本数据类型，Core Java 的基本输入与输出，基本运算符，条件与循环语句，方法）；引用（含引用的含义与操作，对象与引用的基础，字符串，数组，异常处理，使用流类实现的输入与输出）；对象与类（含面向对象程序设计的含义，Javadoc；基本方法，软件包，附加的构造）；继承（含继承的含义，继承的基本语法，多重继承，接口，通用组件的实现）。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 的使用步骤和习题参考答案，以及 Core Java 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 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，2005. 2
ISBN 7-121-00897-1

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（2004）第 006199 号

责任编辑：龚立堇
印　　刷：北京东光印刷厂
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：787×1 092　1/16　印张：25.5　字数：650 千字
印　　次：2005 年 2 月第 1 次印刷
印　　数：5 000 册　　定价：32.00 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，适用于编写各式各样的软件，适用于各种平台与操作系统，编译后的代码能在互联网上传递，并确保用户安全运行，因而是当前最富有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，起码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和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（Applications）和小应用程序（Applets）两大特点，以及为了教学叙述的方便，特分为《Core Java 应用程序设计教程》和《Core Java 小应用程序设计教程》两册来阐述。本书为《Core Java 应用程序设计教程》，主要阐述 Core Java 应用程序设计方法与技巧，内容涉及：基本 Core Java；引用；对象与类；继承。《Core Java 小应用程序设计教程》阐述基本 Applet；图形程序设计；事件处理；图形用户界面构件；网络通信与声像播放。本书每章均有小结、重点对象和习题，并在附录中提供了习题参考答案。本书共有 310 条示例，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，以及 Core Java 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。

在本书编写中，承蒙美国某公司副总裁刘涌博士提供了大量资料，广州私立华联学院教师与学生林汶彬、陈亮、杨红柱、陈小强、王少莉等参与本书工作，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 1 月

目　录

第 1 章　基本 Core Java

本章介绍基本 Core Java，内容涉及：Core Java 的特点与基本编程模式、基本数据类型、Core Java 的基本输入与输出、基本运算符、条件与循环语句、方法。

1.1　Core Java 的特点与基本编程模式

1.1.1　Core Java 的特点

1. Core Java 的含义与作用

（1）Core Java 是独立于平台的通用型面向对象编程语言

Java 是美国 Sun 公司于 1995 年 5 月公布的一种通用的面向对象编程语言，而 Core Java 是 Java 的最新版本（Java 2）。它除了包含 Java 的所有功能之外，还增添一些新的特点，从而成为更为简单易学的一种基于网络的、通用型面向对象编程语言，它能编写各式各样软件。与 Java 一样，Core Java 包含了两层意思：第一，它是 Sun 公司研制的一种新型简化的基于对象的开放系统语言，允许软件开发人员将应用程序发布到万维网（World Wide Web，简称 WWW）或由分布信息系统厂商所开发的任一前端设备上；第二，它是一种虚拟机（或称 Java 虚拟机），其最终能使基于 Core Java（Java 2）的应用程序变成一般的程序，即不管硬件或操作系统环境如何，Core Java 应用程序都可以运行。对于 Java 虚拟机来说，同样的 Core Java 应用程序，完全可以在无任何改变或重新编译的情况下，在任何平台上运行。

（2）Core Java 能编写独立应用程序和小应用程序，完成各式各样的任务

Core Java 与其他大部分程序设计语言不同，它除了能创建独立应用程序（Applications）之外，还能创建小应用程序（Applets）。小应用程序的基本功能与独立应用程序一样，但它能在 Java-Capable 浏览器内运行。小应用程序出现在网页中，很像图像，但又不同于图像，它是动态的和交互的；小应用程序能用来创建动画、图形或者响应用户输入的区域、游戏，以及能在同一网页上，完成文本和图形之间的其他交互作用。

为创建一个小应用程序，必须用 Java（或 Core Java）编写程序，用 Java 编译器编译，并在 HTML Web 页上嵌入小应用程序，使用制作普通 HTML 和图像文件一样的方式，把结果的 HTML 和 Java 文件放到网址上。

（3）小应用程序与独立应用程序的区别

Core Java 程序分为小应用程序和独立应用程序两类。

- 小应用程序是在万维网上下载，并由用户机上的网页浏览器执行的 Core Java 程序。小应用程序不能直接执行和使用，要执行小应用程序，必须通过支持 Core Java 兼容的浏览器。因此，可以把 Core Java 小应用程序看做是在网络浏览器中执行的应用程序。

• 独立应用程序是用 Java（或 Core Java）语言编写的更通用的程序，它不需要浏览器来运行，而只需要 Java 虚拟机来运行，即使用 Java 解释器来执行应用程序。

除了上述的基本浏览器和非浏览器差别之外，小应用程序和独立应用程序之间并没有什么不同，Java（或 Core Java）程序可以是小应用程序、独立应用程序、或者两者的组合，取决于如何编写和使用程序。

2. Core Java 语言的特点

概括地说，Core Java 具有以下的特点。

（1）Core Java 独立于平台

所谓独立于平台就是把程序从某一计算机系统移到另一计算机系统的能力。平台独立性是 Core Java 语言拥有的并超过其他编程语言的最主要优点之一。与其他一些独立于平台的语言（如 ANSI-C）不同，Core Java 在源级和二进制级上都是独立于平台的。

在源级上，Core Java 的基本数据类型具有和所有开发平台相适应的规格。Core Java 的基类库使得程序容易编写，能把代码从一个平台移到另一个平台，而不需要专为在该平台工作而重新编写代码。Core Java 二进制文件也是独立于平台的，可以在多种平台上运行，而不需要重新编译源文件。实际上，Core Java 二进制文件在形式上称为字节码。字节码是一组指令，看起来很像机器码，但又不依赖于任何一种处理器。

当用 Core Java 编写程序时，Core Java 开发环境有 Java 编译器和解释器两部分。Java 编译器用 Java 程序代替从源文件产生的机器代码，它产生字节码。为运行 Java 程序，应运行字节码的解释器程序，这个解释器同样地执行 Java 程序。也可以单独地运行解释器或者（对小应用程序而言）字节码解释器（称为虚拟机），装入 Java 兼容的浏览器，可以运行小应用程序。

用字节码表示的 Core Java 程序，意味着不依赖于一个系统，程序可以在任何一个平台上和任何一个操作系统上运行。而且，使用字节码的优点是执行速度快。

（2）Core Java 面向对象

Core Java 具有面向对象方法的所有优点，以及创建灵活的模块化程序和重用代码的能力。Core Java 许多面向对象的概念都是从 C++继承的，它包含一组类库，该类库提供基本数据类型，系统输入与输出的能力，以及其他实用函数。

（3）Java 简单易学

Java 设计的目标是简单，易编写程序，易编译和调试。Java 是由类和方法所组成，用户可以编写类和方法，并由此创建一个 Java 程序，但在编写程序时应尽量充分利用 Java 类库中已存在的丰富的类与方法，因而，学习 Java 语言实际上应包括两方面：一方面是学习 Java 语言编写自己所需的类和方法，另一方面是学习如何用 Java 类库中的类和方法。

Java 是仿照 C 和 C++的，而大部分语法和面向对象结构都是借用 C++的，如果熟悉 C++，学习 Java 就特别容易。

虽然 Java 与 C 和 C++类似，但 Java 排除了 C 和 C++语句较复杂的部分,从而使得 Java 语句更简单。在 Java 中既没有指针，也没有指针运算，而字符串和数组是真正的对象，存储管理是自动的。

（4）Core Java 更简单易学

Core Java（Java 2）是 Sun 公司的最新版本，特别是基本数据输入/输出方面特别简单。

对 Java 来说，要实现输入数据至少需要写四条语句，而使用 Core Java，只要一条语句就可以了。另外，要实现格式化数据输出，使用 Java 来实现，也很麻烦，而用 Core Java 只要一条语句即可。

1.1.2 Core Java 的基本编程模式

1. 注解

Core Java 有三种注解形式。

第一种形式是从 C 继承的多行注解，以“/*”开始，“*/”结束。例如：

```
/* This is a two line comment */
```

这种形式注解不能嵌套，即不能在一个注解中含有另一个注解。

第二种形式是从 C++继承的，以“//”开始，没有结束标志，而是注解扩展到行结束为止。例如：

```
// This is a line comment
```

第三种形式是以“/**”开始，而不是以“/*”开始。这种形式的注解可用于提供 javadoc 实用程序的信息，其将根据注解产生文档。实际上，javadoc 是一个能自动生成文字的工具，它从程序中抽取方法原型及特定格式的注解，生成优质的超文本文档。例如：

```
/** Java documentation comment */
```

编程时，加上适当的注解，能提高程序的可读性，不仅使需要使用或修改时易读，而且编程人员本身也易读。注解被 Java 编译器所忽略，不会使计算机产生任何操作。

2. 基本编程模式

Core Java 有应用程序（Applications）和小应用程序（Applets）两种编程模式。本书重点介绍 Applications，而 Applets 放在《Core Java 小应用程序教程》一书中介绍。

（1）应用程序基本编程模式

Core Java 独立应用程序的基本编程模式为

```
import corejava.*;                          //引入 corejava 包中本程序用到的类
public class 用户定义的类名                  //定义类
{
  public static void main(String[ ] args)   //主方法
  {
    方法体
  }
}
```

注意：

关键字 class 前面的 public 可以省略。用户定义的类名用做文件名，并以 java 作为扩展名。

从表面上看，Core Java 和 Java 的编程模式基本相同，而实际上，Core Java 比 Java 更为简单易用。

下面，我们通过两个简单的例子看 Java 与 Core Java 在编写独立应用程序上的差别，以

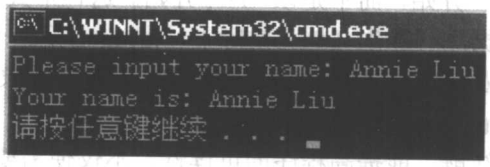
及 Core Java 的编程风格。

【例 1.1.1】输入您的姓名，并在屏幕上显示所输入的姓名。

方法一：（用 Java 编程）

```
//NameJava.java
import java.io.*;
public class NameJava
{
  public static void main(String[] args)throws IOException
  {
    DataInputStream dis=new DataInputStream(System.in);
    System.out.print("Please input your name: ");
    String name=dis.readLine();
    System.out.println("Your name is: "+name);
  }
}
```

运行结果：



说明：

- 由于 DataInputStream 类是在 java.io 软件包中，故在程序开头要写上引入语句：import java.io.*;
- main 为主方法，当程序运行时，就调用 main 方法。紧接 main()后的 throws IOException 表示抛出输入输出异常，即异常处理（以后介绍）。Java 和 Core Java 程序都是由类和方法组成。在 Java 和 Core Java 中，方法等价于函数或过程，是静态（static）方法。当运行任何程序时，调用专用的静态方法 main，或者用命令行参数调用。要求写出 main 的参数类型和 void 返回类型。此外，在 main()后要加上 throws IOException 来抛出异常。
- DataInputStream 是在 java.io 包中的数据输入流类，语句：

  DataInputStream dis=new DataInputStream(System.io);

是用于创建一个 DataInputStream 类的对象 dis。

- System.out.print()是通过 System.out 类调用 print()方法来实现输出的，这里是输出“Please input your name:”，提示用户输入您的姓名。
- String name=dis.readLine();是一个输入语句，dis 对象通过点号运算符“.”调用 readLine()方法，以实现输入一个字符串（姓名）并存入字符串变量 name 中。
- println()同样是用于执行输出，通过使用 println 方法，把字符串 name 放到标准输出流 System.out 中（输入/输出将在后面介绍）。现在只需记住上述语法，它可用于执行输出任何实体，不管实体是一个整数、浮点数、字符串，或者其他类型，都用这种语法。

注意：System.out.println()方法输出并显示其参数，且自动加上一个换行符，而 System.out.print 则不换行。System.print 方法输出字符串一般放入一个缓冲区，并不真正显示在屏幕上，System.out.flush()可强迫缓冲区的字符显示于屏幕。

方法二：（用 Core Java 编程）

方案一

```
//NameCoreJava.java
import corejava.*;
public class NameCoreJava
{
  public static void main(String[] args)
  {
    String name=Console.readLine("Please input your name: ");
    System.out.println("Your name is: "+name);
  }
}
```

运行结果与方法一同。

方案二

```
//NameCoreJava1.java
import corejava.*;
public class NameCoreJava1
{
  public static void main(String[] args)
  {
    String name=Console.readLine("Please input your name: ");
    Format.printf("Your name is: %s\n",name);
  }
}
```

运行结果与方法一同。

比较上述用两种方法编写的程序，可以看出 Core Java 在输入输出方面做了较大的改进：

- 在程序开头的 import 语句，Core Java 是使用 import corejava.*;，而 Java 是使用 import java.io.*;。
- 在提示和输入字符串方面，Core Java 只需使用一条语句，即使用 Console 类调用 readLine 方法来实现：

```
String name =Corsole.readLine("Please input your name: ");
```

而 Java 则要使用三条语句，第一条创建输入流类对象，第二条提示输入字符串（姓名），第三条输入字符串（姓名）：

```
DataInputStream dis=new DataInputStream(System is);
System.out.print("Please input your name: ");
String name=dis.readLine();
```

- 在输出方面，Core Java 使用 Format 类调用 printf 方法来实现，其形式类似于 C 语言的 printf，要在 printf 参数表中指定格式串：

```
Format.printf("Your name is: %s\n",name);
```

也可使用与 Java 一样的 System.out.print()形式输出：

```
System.out.println("Your name is: "+name);
```

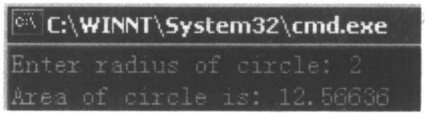
但是，使用 Java 的这种输出方式，难以实现格式化输出。

【例 1.1.2】输入圆的半径，求圆的面积（使用 Application 编程）。

方案一：（用 Java 编程）

```
//CircleAreaJavaApp.java
import java.io.*;                                          //引入 java.io 包中本程序要用到的类
class CircleAreaJavaApp                                    //定义类
{
  public static void main(String[] args)throws IOException  //调用主方法并抛出 IO 异常
  {
    DataInputStream dis=new DataInputStream(System.in);    //创建数据输入流对象
    System.out.print("Enter radius of circle: ");
    String s=dis.readLine();                               /*通过对象 dis 调用 readLine 方法输入
                                                             数字字符串，并存入变量 s 中*/
    double radius=Double.parseDouble(s);                   /*通过 Double 类调用 parseDouble 方
                                                             法，将数字字符串转换为 double 数，
                                                             并赋给 double 型变量*/
    //或 double radius=new Double(s.trim()).doubleValue();
    //或 double radius=new Double(s).doubleValue();
    //或 double radius=Double.valueOf(s).doubleValue();
    double Area=3.14159*radius*radius;
    System.out.println("Area of circle is: "+Area);
  }
}
```

运行结果：



方案二：（用 Core Java 编程）

```
//CircleAreaCoreJavaApp.java
import corejava.*;
class CircleAreaCoreJavaApp
{
  public static void main(String[] args)
  {
    double radius=Console.readDouble("Enter radius of circle: ");
    double Area=3.14159*radius*radius;
    Format.printf("Area of circle is: %f\n",Area);
    System.out.println("Area of circle is: "+Area);
  }
}
```

运行结果与方案一同。

说明：

- 从两种方法的编程看出，Core Java 的输入/输出比 Java 简单得多。
- 对输入数据来说，Java 要实现提示和输入一个数据，需要使用四条语句：第一条创建数据输入流类对象；第二条提示输入数据；第三条使用输入流类对象通过点号运算符调用 readLine()方法来输入数据，并把输入的数据作为一个字符串存入字符串变量中；第四条通过封套类（Double）调用相应的方法，把该字符串转换成所输入的数据：

```
double radius=Double.pareDouble(s);
```

或 double radius=new Double(s).doubleValue();

或 double radius=Double.valueOf(s).doubleValue();

而 Core Java 实现输入一个数据，只要使用一条语句（比 Java 少三条语句），即使用 Console 类调用 readDouble 方法即可完成数据的输入。

- 对输出数据来说，可用 Format.printf()来实现，也可用 System.out.println()来实现，前者既简单，还可实现格式化输出。而后者是 Java 的基本输出机制，可用于执行任何实体的输出，不管实体是整数、浮点数、字符串或其他类型都用这种语法，但要实现格式化输出很麻烦。

（2）小应用程序基本编程模式

小应用程序的基本编程模式为

```
import java.awt.Graphics;                    //引入 java.awt 包中的 Graphics 类
import java.applet.Applet;                   //引入 java.applet 包中的 Applet 类
public class 用户定义的类名 extends Applet    //定义类
{
  public void paint(Graphics g)              //调用 paint 方法
  {
    方法体
  }
}
```

注意:

① import java.awt.Graphics;

import java.applet.Applet;

可写成

mport java.awt.*;

import java.applet.*;

星号（*）说明 java.awt 和 java.applet 包中所有的类，均可提供给编译器进行编译，但整个包被引入时，编译器将只装入该程序中用到的类，即“*”表示只引入该程序中用到的所有类。

② import java.awt.Graphics;

import java.applet.Applet;

public class 用户定义的类名 extends Applet

{

…

}

可写成

import java.awt.Graplics;

public class 用户定义的类名 extends java.applet.Applet

{

…

}

说明：

- import 语句用于装载 Java 程序编译时所需的类。
- 关键字 class 用于类的定义，关键字 extends 使新类可以从已存在的类定义中继承，所有的 Java 小应用程序 applet 必须从 Applet 类中继承。类用于将对象实例化在内存中以便程序使用。一个对象是计算机内存中的一个区域，专用于存储程序所需的信息。从 Applet 继承的类，一般是一个 public 类。左花括号标志着每个类定义的开始，对应的右花括号用于类定义的结束。
- applet 的类名用做文件名，并以.java 作为扩展名。
- paint 方法自动被调用来显示信息。关键字 void 表示一个方法完成其任务后不返回任何信息。每个方法定义的内容由左花括号开始，以对应的右花括号结束。
- 每个类都编译成独立的文件，该文件以类名作为文件名，以扩展名.class 结束。
- 坐标是按像素点从 applet 的左上角即 X 轴 0 点和 Y 轴 0 点开始计算。Graphics 类中的大部分绘图方法都要求至少指定一个坐标系，以确定 applet 在何处画图。

③ 必须创建一个 HTML（超文本标记语言）文件，以便将 applet 装入浏览器内执行，即要使用 HTML 文件嵌入 applet，HTML 文件格式为

```
<HTML>
<HEAD>
<TITLE> 标题 </TITLE>
</HEAD>
<BODY>
<P> 分段说明 </P>
<APPLET CODE=类名.class WIDTH=宽度  HEIGHT=高度>
</APPLET>
</BODY>
</HTML>
```

也可以写成：

```
<APPLET CODE=类名.class WIDTH=宽度  HEIGHT=高度>
</APPLET>
```

说明：

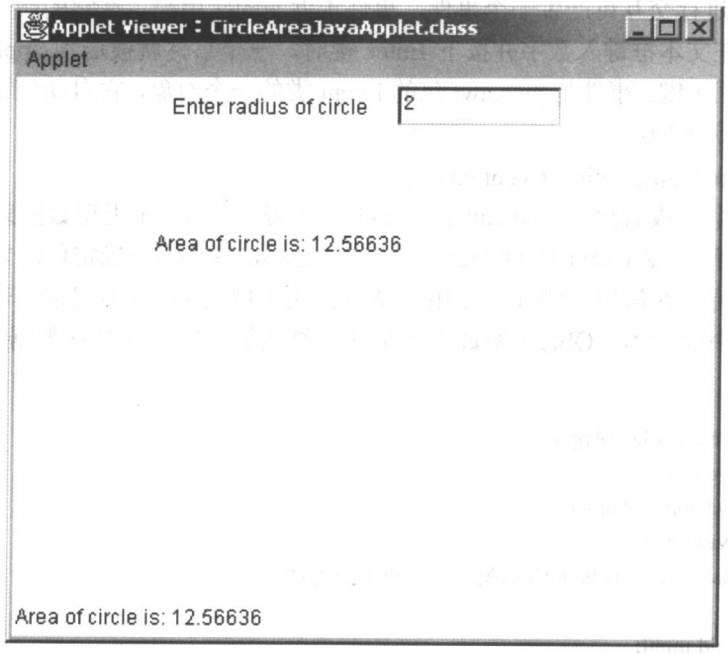
- APPLET 标记激活一个 Java 小应用程序，即以 APPLET 开头的行使得启动 Java-capable Web 浏览器，装入和使用“类名.class”所有被编译的 Java 小应用程序.class 文件扩展。
- 启动 Java-enadble Web 浏览器才能显示小应用程序，其他的浏览器将是显示各种结果，而不是显示小应用程序，某些浏览器将什么也不显示。
- 使用 CODE 属性表示包含用户的小应用程序类的 Java-enable Web 浏览器名。
- 使用 WIDTH 和 HEIGHT 属性，表示小应用程序的浏览器尺寸（以像素为单位），浏览器使用这些值。

【例 1.1.3】输入圆的半径，计算圆的面积（使用 Applet 编程）。

方案一：

```
//CircleAreaJavaApplet.java
import java.awt.*;
import java.applet.Applet;
public class CircleAreaJavaApplet extends Applet
{
  TextField input;
  public void init()
  {
    Label prompt=new Label("Enter radius of circle");
    input=new TextField(10);
    add(prompt);
    add(input);
  }
  public boolean action(Event e,Object o)
  {
    Graphics g=getGraphics();
    String s=input.getText();
    double radius=new Double(s).doubleValue();
    double Area=3.14159*radius*radius;
    g.drawString("Area of circle is: "+Area,80,100);
    showStatus("Area of circle is: "+Area);
    return true;
  }
}
```

运行结果：



说明：

- intit 方法用于初始化 applet 所需的变量和引用。intit 在 applet 开始执行时自动地被调用，并保证是第一个被调用。
- Java 程序中的数据，除了由基本数据类型定义的变量外，都是对象。变量类似于一