

第1章

Java 语言概述

本章读者可以学到如下实例：

- ▶ 实例 001 输出 “Hello World”
- ▶ 实例 002 输出控制台传递的参数
- ▶ 实例 003 输出由 “*” 组成的三角形
- ▶ 实例 004 输出符号表情



实例说明

学习 Java 编程首先要下载和配置 JDK。完成下载和配置之后，可以编写一个简单的程序来检验配置的效果。本实例将在 DOS 控制台上输出“Hello World”，其运行效果如图 1.1 所示。

实现过程

(1) 打开文本编辑软件，如 Windows 系统的记事本，并在其中输入如下代码：

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

(2) 保存文件到 D 盘，并将其命名为 Test，扩展名是.java。对于记事本，可以按 Ctrl+S 键保存文件，并将文件名写成“Test.java”。

脚下留神：

记事本默认会对文件增加.txt 扩展名，因此“Test.java”中的双引号并不能省略。

(3) 打开 DOS 控制台并切换路径到 D 盘，输入“javac Test.java”命令编译源代码，输入“java Test”命令运行 class 文件。

脚下留神：

javac 和 java 命令后面都需要一个空格。

技术要点

本实例主要使用了两个 Java 命令，即 javac 和 java。其中，javac 命令用于编译源文件，后面的文件要带扩展名.java，此时将生成一个 class 文件；java 命令用于运行 class 文件，并在 DOS 控制台上显示运行的结果。

实例 002 输出控制台传递的参数

(实例位置：配套资源\SL\01\002)

实例说明

在使用 java 命令运行程序时，可以同时传递多个参数。本实例将输出用户传递的参数，其

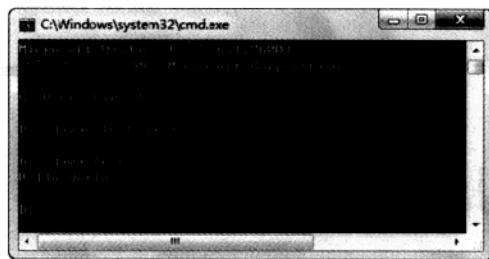


图 1.1 输出“Hello World”



运行效果如图 1.2 所示。

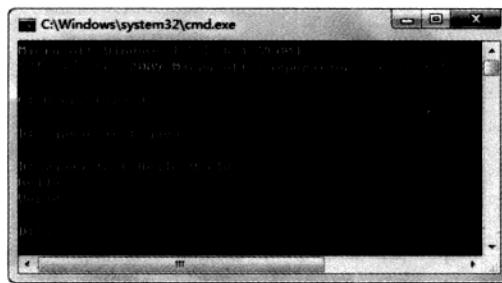


图 1.2 输出用户传递的参数

实现过程

(1) 打开文本编辑软件, 如 Windows 系统的记事本, 并在其中输入如下代码:

```
public class Test {
    public static void main(String[] args) {
        System.out.println(args[0]);
        System.out.println(args[1]);
    }
}
```

(2) 保存文件到 D 盘, 并将其命名为 Test, 扩展名是.java。对于记事本, 可以按 Ctrl+S 键保存文件, 并将文件名写成 “Test.java”。

脚下留神:

记事本默认会对文件增加.txt 扩展名, 因此 “Test.java” 中的双引号并不能省略。

(3) 打开 DOS 控制台并切换路径到 D 盘, 输入 “javac Test.java” 命令编译源代码, 输入 “java Test Hello World” 命令运行 class 文件。

脚下留神:

javac 和 java 命令后面都需要一个空格。

技术要点

使用 java 命令时, 如果传递多个参数, 它们之间需要使用空格进行分隔。传递的参数保存在一个 String 类型的数组中并传递给 main()方法。在 main()方法中, 可以使用其方法参数调用传递的值。

实例 003 输出由 “*” 组成的三角形

(实例位置: 配套资源\SL\01\003)

实例说明

在学习了基本的输出语句后, 就可以使用它来输出一些简单的图形。本实例将输出一个由





“*”组成的三角形，其运行效果如图 1.3 所示。



Note

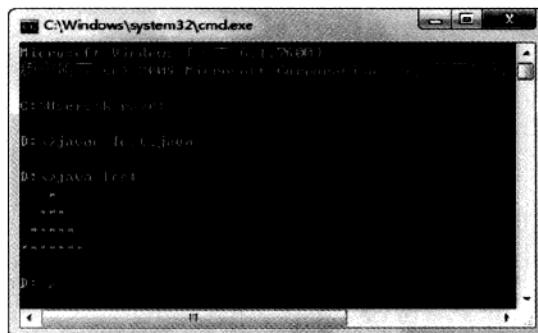


图 1.3 输出三角形

实现过程

(1) 打开文本编辑软件，如 Windows 系统的记事本，并在其中输入如下代码：

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println(" *");  
        System.out.println(" ***");  
        System.out.println(" *****");  
        System.out.println("*****");  
    }  
}
```

多学两招：

也可以使用循环语句来输出三角形，这样编写的代码更加简洁。

(2) 保存文件到 D 盘，并将其命名为 Test，扩展名是.java。对于记事本，可以按 Ctrl+S 键保存文件，并将文件名写成“Test.java”。

脚下留神：

记事本默认会对文件增加.txt 扩展名，因此“Test.java”中的双引号并不能省略。

(3) 打开 DOS 控制台并切换路径到 D 盘，输入“javac Test.java”命令编译源代码，输入“java Test”命令运行 class 文件。

脚下留神：

javac 和 java 命令后面都需要一个空格。

技术要点

使用 javac 命令在编译源代码时，需要指明代码的类型，即需包含.java 扩展名。使用 java 命令在运行 class 文件时，不需要指明扩展名，即不能包括 class。



实例 004 输出符号表情

(实例位置: 配套资源\SL\01\004)

实例说明

除了可以输出字符串和简单的几何图形外,还可以通过字符串的适当组合,输出符号表情。本实例将在控制台上输出一个猪头,其运行效果如图 1.4 所示。

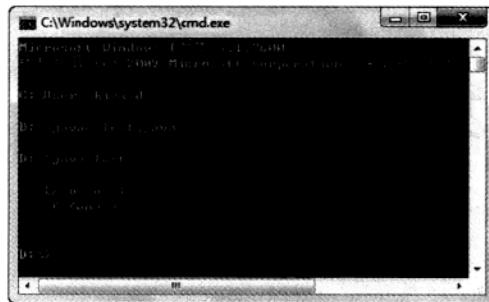


图 1.4 输出猪头

实现过程

(1) 打开文本编辑软件,如 Windows 系统的记事本,并在其中输入如下代码:

```
public class Test {
    public static void main(String[] args) {
        System.out.println("  ^__^ ");
        System.out.println("  { o   o }");
        System.out.println("  ((oo))");
        System.out.println("  ");
    }
}
```

(2) 保存文件到 D 盘,并将其命名为 Test,扩展名是.java。对于记事本,可以按 Ctrl+S 键保存文件,并将文件名写成“Test.java”。

脚下留神:

记事本默认会对文件增加.txt 扩展名,因此“Test.java”中的双引号并不能省略。

(3) 打开 DOS 控制台并切换路径到 D 盘,输入“javac Test.java”命令编译源代码,输入“java Test”命令运行 class 文件。

脚下留神:

javac 和 java 命令后面都需要一个空格。

技术要点

Java 中使用 System.out.println();语句完成在控制台上的输出,该语句在输出完毕后会自动换行。类似地还有 System.out.print();语句,它在输出完毕后不会换行。



第 2 章

Eclipse 开发工具

本章读者可以学到如下实例：

- 实例 005 下载并运行 Eclipse 工具
- 实例 006 为 Eclipse 安装汉化包
- 实例 007 使用 Eclipse 注释代码
- 实例 008 使用 Eclipse 格式化代码
- 实例 009 安装 WindowBuilder 插件
- 实例 010 开发计算器界面



实例 005 下载并运行 Eclipse 工具

实例说明

在开发 Java 程序时，首先要编写源代码。虽然可以使用简单的文本编辑器完成该工作，但是效率非常低。专业的开发人员会选择一款适合自己风格的集成开发环境。本实例讲述如何下载和运行 Eclipse 工具，其运行效果如图 2.1 所示。

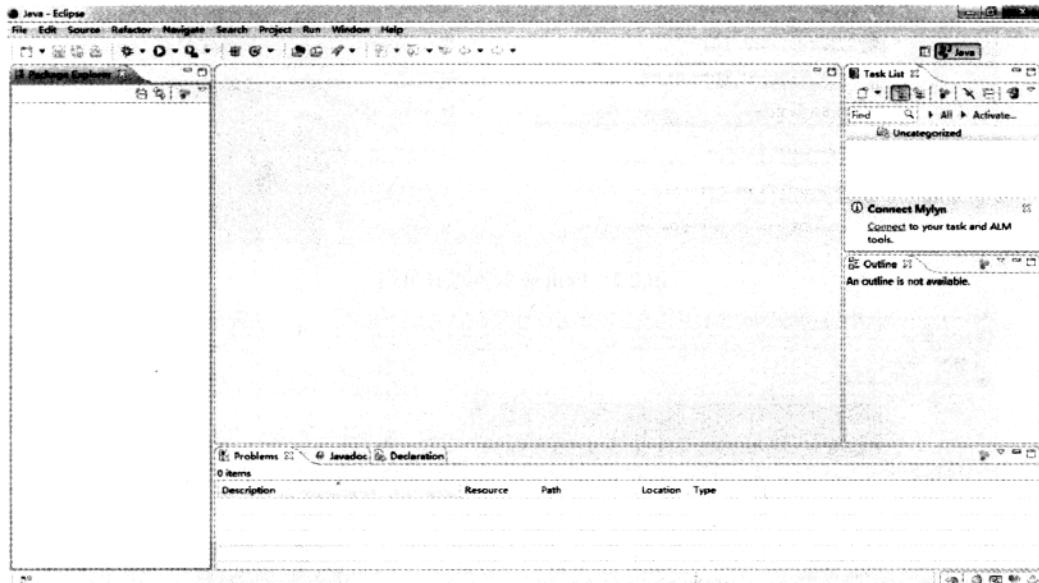


图 2.1 Eclipse 开发环境

实现过程

(1) 打开浏览器，在地址栏中输入“<http://www.eclipse.org/downloads/>”，按 Enter 键将打开 Eclipse 的主页。单击页面中的 Eclipse Downloads 超链接，如图 2.2 所示。

(2) 选择 Eclipse IDE for Java Developers 中的 Windows 32 Bit 下载。如果读者使用的是 64 位计算机，可以选择 Windows 64 Bit 下载。然后跳转到如图 2.3 所示的页面。

(3) 单击[China] Actuate Shanghai (<http://www.eclipse.org/downloads/>)超链接开始下载。保存压缩包到磁盘，并解压缩。运行其中的 eclipse.exe 文件，得到的效果如图 2.1 所示。

技术要点

在图 2.2 中，提供了多个 Eclipse 版本，如适合进行 Java EE 开发的版本、适合进行 PHP 开发的版本等。读者可以根据自己的需求选择合适的版本。





Note

The screenshot shows the Eclipse Downloads page. At the top, there's a banner for 'eclipseCON™ 2011' and a 'Register Now' button. Below the banner, there's a navigation bar with links to Home, Downloads, Users, Members, Committers, Resources, Projects, and About Us. A search bar is also present. The main content area is titled 'Eclipse Downloads' and lists several packages:

- Eclipse IDE for Java Developers (99 MB)
- Eclipse IDE for Java EE Developers (205 MB)
- Eclipse IDE for C/C++ Developers (86 MB)
- JRebel for Eclipse (Promoted Download!)
- Eclipse Classic 3.8.1 (170 MB)
- Eclipse for PHP Developers (14 MB)
- Eclipse IDE for JavaScript Web Developers (106 MB)

Each package entry includes a 'Downloaded' count, a 'Details' link, and a 'Windows' section showing 32-bit and 64-bit options. To the right, there's a 'Hint' box with instructions about Java runtime requirements and a 'Master BIRT' sidebar.

图 2.2 Eclipse 版本选择页面

The screenshot shows the 'Eclipse downloads - mirror selection' page. It features a 'Register Now' banner and a 'Downloads' sidebar with options for Bit Torrents, Source code, and More Packages. The main content area is titled 'Eclipse downloads - mirror selection' and contains the following information:

- A note stating that all downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.
- A download link for 'eclipse-java-helios-SR1-win32.zip' with a 'SHA1' checksum.
- An option to 'pick a mirror site below'.
- A 'Give Back to Eclipse' section with donation buttons for \$5, \$15, and \$25, and a 'Become a friend of Eclipse' link.
- A 'Get It Faster Here' section with links to Google, GetEclipse, and Youcan.
- A 'GIVE BACK TO THE ECLIPSE COMMUNITY' sidebar with a 'BECOME AN ECLIPSE FOUNDATION MEMBER' button.
- Links for 'HTTP mirrors only (xml)', 'FTP mirrors only (xml)', and 'All mirrors (xml)'.
- A 'Related Links' section.

图 2.3 选择 Eclipse 下载链接

实例 006 为 Eclipse 安装汉化包

实例说明

为了方便中国用户的使用, Eclipse 提供了汉化包。本实例将演示如何下载并安装汉化包, 汉化后的 Eclipse 如图 2.4 所示。

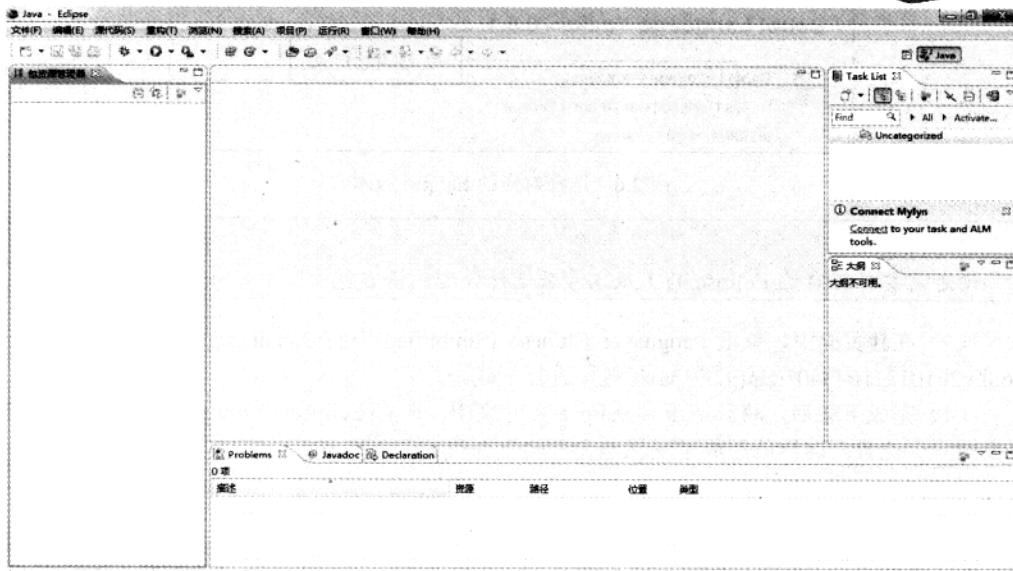
**Note**

图 2.4 汉化后的 Java 开发环境界面

实现过程

(1) 在浏览器的地址栏中输入“<http://www.eclipse.org/babel/>”，按 Enter 键将打开 Babel 项目的网页，如图 2.5 所示。单击 Downloads 超链接。

图 2.5 Eclipse Babel 项目主页

(2) 在打开的 Eclipse Babel 下载页面中单击 Helios 超链接，如图 2.6 所示。



Note

指点迷津：

读者需要根据自己 Eclipse 的主版本号来选择合适语言包。

Babel Language Packs - 0.8.1



[Babel Language Pack Zips](#)

Helios | Galileo | Ganymede | Europa

[Installation instructions](#)

单击该超链接

图 2.6 选择对应的 Eclipse 版本

技术要点

Eclipse Babel 项目中提供了多种语言包，如图 2.7 所示。读者可以根据自己的需求进行选择。

The following language packs are based on the community translations entered into the Babel Translation Tool, and may not be complete or entirely accurate. If you find missing or incorrect translations, please use the Babel Translation Tool to update them. All downloads are provided under the terms and conditions of the Eclipse Foundation Software Use Agreement unless otherwise specified.

Go to Arabic Bulgarian Catalan Chinese (Simplified) Chinese (Traditional) Czech Danish Dutch English (Australian) Estonian Finnish French German Greek Hebrew Hindi Hungarian Indonesian Italian Japanese Klingon Korean Mongolian Norwegian Persian Polish Portuguese (Brazilian) Romanian Russian Spanish Swedish Turkish Ukrainian Welsh Translations

Language: Arabic

- BabelLanguagePack.alit.ar_3.6.0.v20101211043401.zip (69.48%)
- BabelLanguagePack.datatools.ar_3.6.0.v20101211043401.zip (22.55%)
- BabelLanguagePack.gdp_mig_ar_3.6.0.v20101211043401.zip (11.23%)
- BabelLanguagePack.gdp_mn_ar_3.6.0.v20101211043401.zip (21%)
- BabelLanguagePack.gdp_mm_ar_3.6.0.v20101211043401.zip (72.92%)
- BabelLanguagePack.eclipse_ar_3.6.0.v20101211043401.zip (64.96%)
- BabelLanguagePack.modelling_emt_validation_ar_3.6.0.v20101211043401.zip (16.65%)
- BabelLanguagePack.modelling_emt_cst_ar_3.6.0.v20101211043401.zip (16.65%)
- BabelLanguagePack.modelling_gmf_prf_ar_3.6.0.v20101211043401.zip (36.29%)
- BabelLanguagePack.modelling_gmf_gmf_toolbag_ar_3.6.0.v20101211043401.zip (28.46%)
- BabelLanguagePack.modelling_gmf_mft_ar_3.6.0.v20101211043401.zip (27.79%)
- BabelLanguagePack.t_latinino_ar_3.6.0.v20101211043401.zip (17.78%)
- BabelLanguagePack.t_latinino_ar_3.6.0.v20101211043401.zip (17.78%)
- BabelLanguagePack.t_latinino_ar_3.6.0.v20101211043401.zip (17.11%)
- BabelLanguagePack.t_turkmen_ar_3.6.0.v20101211043401.zip (66.34%)
- BabelLanguagePack.technology_eqlt_ar_3.6.0.v20101211043401.zip (16.65%)
- BabelLanguagePack.technology_eqlt_ar_3.6.0.v20101211043401.zip (9.65%)
- BabelLanguagePack.technology_jgit_ar_3.6.0.v20101211043401.zip (0.51%)

图 2.7 语言选择界面

实例 007 使用 Eclipse 注释代码

(实例位置：配套资源\SL\02\007)

实例说明

在编写代码的过程中，需要经过反复的调试以达到最佳的效果。通过注释代码可以直观地看到这段代码所起的作用。为了方便用户使用，Eclipse 提供了很多快捷键。本实例将演示注释



Note

代码的快捷键的使用，其运行效果如图 2.8 所示。

```
*Test.java
1 package com.mingrisoft;
2
3 public class Test {
4     public static void main(String[] args) {
5         // System.out.println("《Java 编程词典》真好用！");
6     }
7 }
```

图 2.8 注释代码的效果

实现过程

- (1) 在 Eclipse 中创建项目 007，并在该项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建类文件，名称为 Test。在类中输入如下代码：

```
public class Test {
    public static void main(String[] args) {
        System.out.println("《Java 编程词典》真好用！");
    }
}
```

(3) 将光标停留在“《Java 编程词典》真好用！”一行上，按 Ctrl+/ 键，可以看到该行代码已经被注释了，效果如图 2.8 所示。

技术要点

Java 中支持 3 种注释格式，即单行注释、多行注释和文档注释。使用 Eclipse 时，可以同时选择多行代码，然后使用 Ctrl+/ 快捷键来同时注释。

实例 008 使用 Eclipse 格式化代码

(实例位置：配套资源\SL\02\008)

实例说明

在编写代码的过程中可能要反复地修改以达到最佳效果，此时可能造成代码格式混乱。Eclipse 提供了格式化代码的快捷键，可以帮助程序员完成单调的格式化任务。本实例将演示该快捷键的使用，格式化之前的效果如图 2.9 所示，格式化之后的效果如图 2.10 所示。

```
*Test.java
1 package com.mingrisoft;
2
3 public class Test {
4     public
5     static
6     void
7     main(String[]
8     args) {
9         System.out.println("《Java 编程词典》真好用！");
10    }
11 }
```

图 2.9 格式化代码前的效果

```
*Test.java
1 package com.mingrisoft;
2
3 public class Test {
4     public static void main(String[] args) {
5         System.out.println("《Java 编程词典》真好用！");
6     }
7 }
```

图 2.10 格式化代码后的效果

实现过程

- (1) 在 Eclipse 中创建项目 008，并在该项目中创建 com.mingrisoft 包。

**Note**

(2) 在 com.mingrisoft 包中创建类文件，名称为 Test。在类中输入如下代码：

```
public class Test {  
    public  
    static  
    void  
    main(String[]  
          args) {  
        System.out.println("《Java 编程词典》真好用！");  
    }  
}
```

(3) 将光标放在编辑器窗体中，按 Shift+Ctrl+F 键，可以看到代码已经被格式化了。

技术要点

除了上面介绍的快捷键外，Eclipse 还提供了大量的快捷键。在菜单栏中选择“窗口”/“首选项”/“常规”/“键”命令，可以看到各种预定义的快捷键的功能。读者可以根据自己的习惯进行修改。

实例 009 安装 WindowBuilder 插件

实例说明

默认的 Eclipse 工具并没有提供图形化的 Swing 程序开发工具。本实例将演示如何安装 WindowBuilder 插件，它是 Google 公司提供的图形化 Swing 程序开发插件。使用它来开发 Swing 程序的界面如图 2.11 所示。

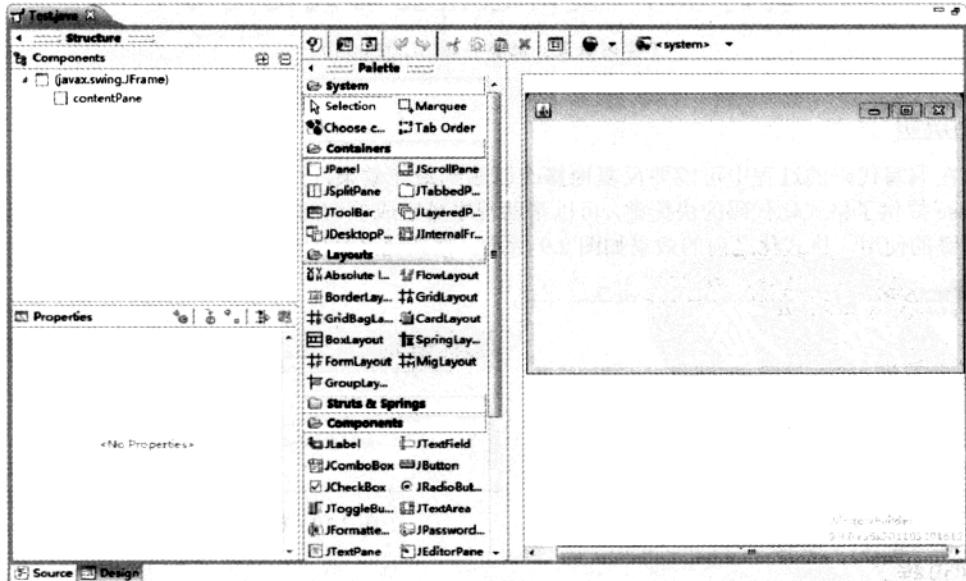


图 2.11 使用 WindowBuilder 插件开发 Swing 程序



实现过程

- (1) 在浏览器的地址栏中输入网址“<http://code.google.com/intl/zh-CN/javadevtools/download-wbpro.html>”，按 Enter 键进入 WindowBuilder 插件的下载页面，根据 Eclipse 的版本，选择对应的更新网址。
- (2) 打开 Eclipse，选择“帮助”/Install New Software 命令，如图 2.12 所示。
- (3) 在打开的增加更新仓库对话框的 Name 文本框中输入“WindowBuilder”，在 Location 文本框中输入在第(1)步中输入的网址，最后单击“确定”按钮，如图 2.13 所示。

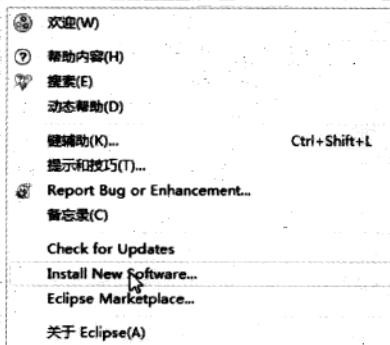


图 2.12 “帮助”菜单项

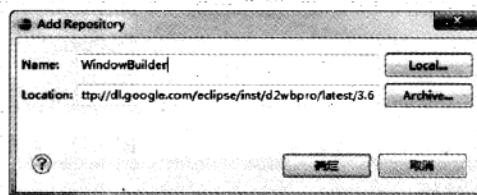


图 2.13 增加更新仓库对话框

- (4) 开始进行安装，中间需要重启一次。在安装完毕后，可以在菜单栏中选择“文件”/“新建”/“其他”命令，会显示如图 2.14 所示的对话框。其中包括了 WindowBuilder 文件夹。

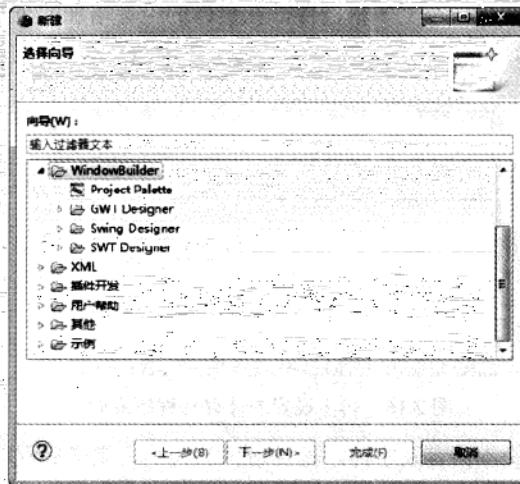


图 2.14 “新建”对话框

技术要点

对于其他的 Eclipse 插件，也可以通过类似的方式完成安装。例如，安装开发 Perl 语言的插件，可以在 www.epic-ide.org 上找到相关的说明。



Note



实例 010 开发计算器界面

(实例位置: 配套资源\SL\02\010)

实例说明

计算器是各种操作系统提供的经典程序之一，在安装完 WindowBuilder 插件后，将演示如何使用它来开发计算器界面。实例的运行效果如图 2.15 所示。

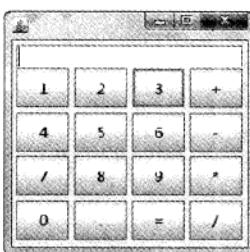


图 2.15 计算器程序界面

实现过程

- (1) 在 Eclipse 中创建项目 010，并在该项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建 JFrame 类型的文件，名称为 Calculator，并在屏幕下方选择 Design 选项卡切换到设计界面。
- (3) 在窗体的顶部增加一个文本框控件，在窗体的中央增加一个面板，然后将该面板设置成 4 行 4 列的网格布局，并增加 16 个按钮，效果如图 2.16 所示。

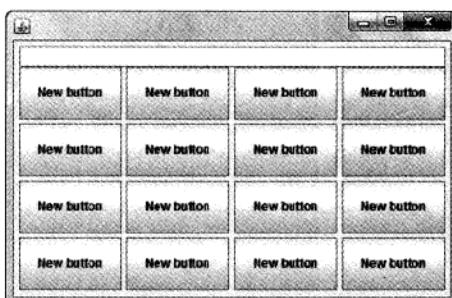


图 2.16 初步设置的计算器程序界面

- (4) 修改按钮上的文本内容及字体，调整窗体的大小，完毕后程序的运行效果如图 2.15 所示。

技术要点

Swing 中控件上的文本非常灵活，不仅能够设置字体，还能够设置颜色。此外，还可以在按钮控件中使用 HTML 代码实现复杂的样式。

第3章

Java 语言基础

本章读者可以学到如下实例：

- ▶ 实例 011 输出错误信息与调试信息
- ▶ 实例 012 从控制台接收输入字符
- ▶ 实例 013 重定向输出流实现程序日志
- ▶ 实例 014 自动类型转换与强制类型转换
- ▶ 实例 015 加密可以这样简单（位运算）
- ▶ 实例 016 用三元运算符判断奇数和偶数
- ▶ 实例 017 不用乘法运算符实现 2×16
- ▶ 实例 018 实现两个变量的互换（不借助第 3 个变量）



实例 011 输出错误信息与调试信息

(实例位置: 配套资源\SL\03\011)



实例说明

Note

程序开发中对于业务代码的部分功能需要配合调试信息以确定代码执行流程和数据的正确性,当程序出现严重问题时还要输出警告信息,这样可以在调试中完成程序开发,本实例将介绍如何输出调试信息与错误提示信息。实例的运行效果如图 3.1 所示。

实现过程

(1) 在 Eclipse 中新建项目 011, 在项目中创建 com.mingrisoft 包。

(2) 在 com.mingrisoft 包中创建 PrintErrorAndDebug 类, 并完成该类的 main() 主方法, 在该方法中分别输出调试信息与错误信息。关键代码如下:

```
public class PrintErrorAndDebug {  
    public static void main(String[] args) {  
        System.out.println("main()方法开始运行了。");  
        //输出错误信息  
        System.err.println("在运行期间手动输出一个错误信息: ");  
        System.err.println("t 该软件没有买保险, 请注意安全");  
        System.out.println("PrintErrorAndDebug.main()");  
        System.out.println("main()方法运行结束。");  
    }  
}
```

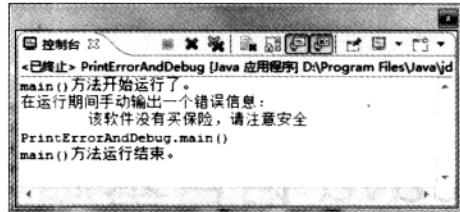


图 3.1 输出错误信息与调试信息

技术要点

本实例使用 System 类的 out 和 err 两个成员变量来完成调试信息与错误信息的输出, 它们两个都是 System 的类变量, 也就是说是使用 static 关键字修饰的。out 是标准调试信息的输出流, err 是标准错误信息的输出流。实例中调用了两个输出流通用的 println() 方法来输出一行数据。该方法的声明如下:

```
public void println(String x)
```

参数说明

x: 被输出到控制台的字符串。

实例 012 从控制台接收输入字符

(实例位置: 配套资源\SL\03\012)

实例说明

System 类除了包含 out 和 err 两个输出流之外, 还有 in 输入流的实例对象作为类成员, 它



可以接收用户的输入。本实例通过该输入流实现从控制台接收用户输入文本，并提示该文本的长度信息。实例的运行效果如图 3.2 所示。

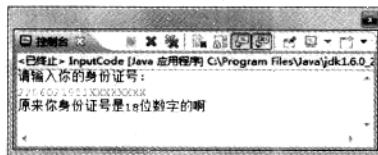


图 3.2 从控制台接收输入字符

实现过程

- (1) 在 Eclipse 中新建项目 012，在项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建 InputCode 类，在该类的主方法中创建 Scanner 扫描器来封装 System 类的 in 输入流，然后提示用户输入身份证号码，并输出用户身份证号码的位数。关键代码如下：

```
import java.util.Scanner;
public class InputCode {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // 创建输入流扫描器
        System.out.println("请输入你的身份证号："); // 提示用户输入
        String line = scanner.nextLine(); // 获取用户输入的一行文本
        // 打印对输入文本的描述
        System.out.println("原来你身份证号是" + line.length() + "位数字的啊");
    }
}
```

技术要点

本实例的关键技术就是用到了 System 类的输入流也就是类变量 in，它可以接收用户的输入信息，并且是标准的输入流实例对象。另外，Scanner 类是 Java 的扫描器类，它可以从输入流中读取指定类型的数据或字符串。本实例使用 Scanner 类封装了输入流对象，并使用 nextLine() 方法从输入流中获取用户输入的整行文本字符串。该方法的声明如下：

```
public String nextLine()
```

返回值：从扫描器封装的输入流中获取一行文本字符串作为方法的返回值。

实例 013 重定向输出流实现程序日志

(实例位置：配套资源\SL\03\013)

实例说明

System 类中的 out 成员变量是 Java 的标准输出流，程序常用它来输出调试信息。out 成员变量被定义为 final 类型的，无法直接重新复制，但是可以通过 setOut() 方法来设置新的输出流。本实例利用该方法实现了输出流的重定向，把它指向一个文件输出流，从而实现了日志功能。程序运行后控制台提示运行结束信息，如图 3.3 所示，但是在运行过程中的步骤都保存到了日志文件中，如图 3.4 所示。





图 3.3 在控制台提示程序运行完毕

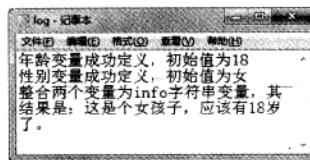


图 3.4 重定向输出流实现程序日志的效果

Note

实现过程

(1) 在 Eclipse 中新建项目 013，在项目中创建 com.mingrisoft 包。

(2) 在 com.mingrisoft 包中创建 RedirectOutputStream 类，编写该类的 main() 主方法，在该方法中保存 System 类的 out 成员变量为临时变量，然后创建一个新的文件输出流，并把这个输出流设置为 System 类新的输出流。在程序关键位置输出调试信息，这些调试信息将通过新输出流保存到日志文件中。最后恢复原有输出流并输出程序运行结束信息。关键代码如下：

```
import java.io.FileNotFoundException;
import java.io.PrintStream;
public class RedirectOutputStream {
    public static void main(String[] args) {
        try {
            PrintStream out = System.out; //保存原输出流
            PrintStream ps=new PrintStream("./log.txt"); //创建文件输出流
            System.setOut(ps); //设置使用新的输出流
            int age=18; //定义整型变量
            System.out.println("年龄变量成功定义，初始值为 18");
            String sex="女"; //定义字符串变量
            System.out.println("性别变量成功定义，初始值为女");
            //整合两个变量
            String info="这是个"+sex+"孩子，应该有"+age+"岁了。";
            System.out.println("整合两个变量为 info 字符串变量，其结果是：" +info);
            System.setOut(out); //恢复原有输出流
            System.out.println("程序运行完毕，请查看日志文件。");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

技术要点

本实例的关键技术是调用了 System 类的 setOut() 方法改变了输出流，System 类的 out、err 和 in 成员变量是 final 型的，不能直接赋值，要通过相应的方法来改变流。下面分别介绍改变这 3 个成员变量的方法。

1. setOut()方法

该方法用于重新分配 System 类的标准输出流。方法的声明如下：

```
public static void setOut(PrintStream out)
```

参数说明

out：新的 PrintStream 输出流对象。



2. setErr()方法

该方法将重新分配 System 类的标准错误输出流。方法的声明如下：

```
public static void setErr(PrintStream err)
```

参数说明

err：新的 PrintStream 输出流对象。

3. setIn()方法

该方法将重新设置 System 类的 in 成员变量，即标准输入流。方法的声明如下：

```
public static void setIn(InputStream in)
```

参数说明

in：新的 InputStream 输入流对象。



Note

实例 014 自动类型转换与强制类型转换

(实例位置：配套资源\SL\03\014)

实例说明

Java 基本数据类型之间存在自动类型转换与强制类型转换两种转换方法。本实例将演示这两种类型转换的用法，实例的运行效果如图 3.5 所示。注意 long 类型向低类型 short 转换时发生的数据丢失。

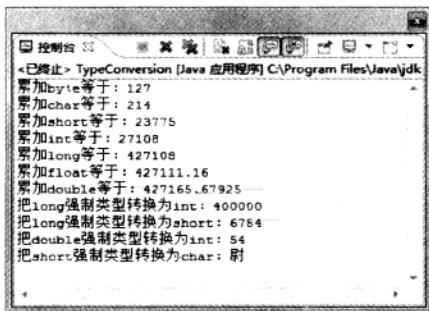


图 3.5 自动类型转换与强制类型转换的输出结果

实现过程

- (1) 在 Eclipse 中新建项目 014，在项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建 TypeConversion 类，在该类的主方法中创建各种基本类型的变量，在输出语句中分别输出所有变量累加值。注意每次累加值的数据类型，所有整数运算都被自动转换为 int 类型再进行运算，所有浮点数值都被自动转换为 double 类型进行运算。最后把高类型数据向低类型数据进行强制类型转换，并注意运算结果是否丢失数据。关键代码如下：

```
public class TypeConversion {
    public static void main(String[] args) {
        byte b = 127;
```



Note

```
char c = 'W';
short s = 23561;
int i = 3333;
long l = 400000L;
float f = 3.14159F;
double d = 54.523;
//低类型向高类型自动转换
System.out.println("累加 byte 等于: " + b);
System.out.println("累加 char 等于: " + (b + c));
System.out.println("累加 short 等于: " + (b + c + s));
System.out.println("累加 int 等于: " + (b + c + s + i));
System.out.println("累加 long 等于: " + (b + c + s + i + l));
System.out.println("累加 float 等于: " + (b + c + s + i + l + f));
System.out.println("累加 double 等于: " + (b + c + s + i + l + f + d));
//高类型到低类型的强制转换
System.out.println("把 long 强制类型转换为 int: " + (int) l);
//高类型到低类型转换会丢失数据
System.out.println("把 long 强制类型转换为 short: " + (short) l);
//实数到整数转换将舍弃小数部分
System.out.println("把 double 强制类型转换为 int: " + (int) d);
//整数到字符类型的转换将获取对应编码的字符
System.out.println("把 short 强制类型转换为 char: " + (char) s);
}
}
```

技术要点

本实例的关键技术是强制类型转换。强制类型转换的实现方法是使用一对小括号，将其他类型转换为指定的类型，语法如下：

转换后的类型 变量 = (转换后的类型)被转换的变量

例如：

```
long value = 100L;
byte btValue = (byte)value;
```

指点迷津：

上面代码将 long 型变量的值强制转换为 byte 类型。

实例 015 加密可以这样简单（位运算）

（实例位置：配套资源\SL\03\015）

实例说明

本实例通过位运算的异或运算符“^”把字符串与一个指定的值进行异或运算，从而改变字符串中每个字符的值，这样就可以得到一个加密后的字符串，如图 3.6 所示。当把加密后的字符串作为程序输入内容，异或运算会把加密后的字符串还原为原有字符串的值，如图 3.7 所示。

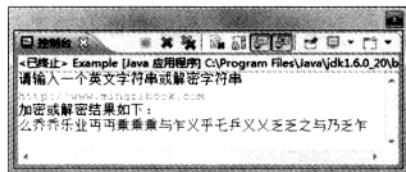


图 3.6 使用异或加密字符串

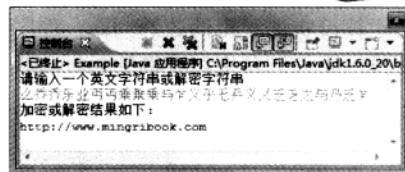


图 3.7 使用异或解密字符串

实现过程

- (1) 在 Eclipse 中新建项目 015，在项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建 Example 类，在该类的主方法中创建 System 类的标准输入流的扫描器对象，提示用户输入一个英文的字符串或者要解密的字符串，然后通过扫描器获取用户输入的字符串，经过加密或解密后，把字符串通过错误流输出到控制台。关键代码如下：

```
import java.util.Scanner;
public class Example {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("请输入一个英文字符串或解密字符串");
        String password = scan.nextLine(); // 获取用户输入
        char[] array = password.toCharArray(); // 获取字符数组
        for (int i = 0; i < array.length; i++) { // 遍历字符数组
            array[i] = (char) (array[i] ^ 20000); // 对每个数组元素进行异或运算
        }
        System.out.println("加密或解密结果如下: ");
        System.out.println(new String(array)); // 输出密钥
    }
}
```

指点迷津：

程序最后使用的标准错误输出流不是用于输出错误信息，而是利用了其在 Eclipse 控制台以红色显示的特性来突出显示。

技术要点

本实例的关键技术是异或运算。如果某个字符（或数值）x 与一个数值 m 进行异或运算得到 y，则再用 y 与 m 进行异或运算就可以还原为 x，因此应用这个原理可以实现加密和解密功能。

实例 016 用三元运算符判断奇数和偶数

（实例位置：配套资源\SL\03\016）

实例说明

三元运算符是 if…else 条件语句的简写格式，它可以完成简单的条件判断。本实例利用这





个三元运算符实现了奇偶数的判断，程序要求用户输入一个整数，然后程序判断是奇数还是偶数，并输出到控制台中。实例的运行效果如图 3.8 所示。



Note

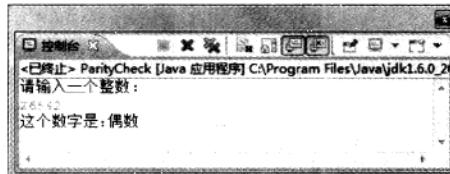


图 3.8 判断奇数和偶数的结果

实现过程

- (1) 在 Eclipse 中新建项目 016，在项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建 ParityCheck 类，在该类的主方法中创建标准输入流的扫描器对象，提示用户输入一个整数，并通过扫描器的方法来接收一个整数，通过三元运算符判断该数字与 2 的余数，如果余数为 0 说明其是偶数，否则是奇数。关键代码如下：

```
import java.util.Scanner;
public class ParityCheck {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in); // 创建输入流扫描器
        System.out.println("请输入一个整数：");
        long number = scan.nextLong(); // 获取用户输入的整数
        String check = (number % 2 == 0) ? "这个数字是：偶数" : "这个数字是：奇数";
        System.out.println(check);
    }
}
```

技术要点

本实例的关键技术就是以三元运算符实现简单的条件判断。其语法格式如下：

条件运算？运算结果 1：运算结果 2；

如果条件运算结果为 true，返回值就是运算结果 1，否则返回运算结果 2。

另外，本实例使用了扫描器的 nextLong() 方法直接获取了整型数据，避免了类型转换等业务代码。该方法的声明如下：

public long nextLong()

返回值：该方法返回一个 long 类型的数值，这个数是从扫描器封装的输入流中获取的。

实例 017 不用乘法运算符实现 2×16

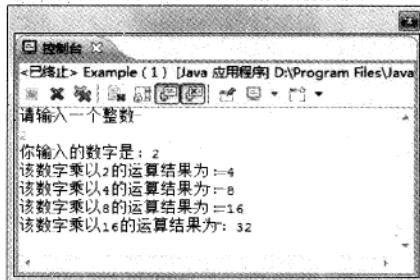
(实例位置：配套资源\SL\03\017)

实例说明

程序开发中常用的乘法运算是通过“*”运算符或者 BigDecimal 类的 multiply() 方法实现的。但是本实例将介绍在这两种方法之外如何实现乘法，而且实现的运算效率非常高。实例的运行效果如图 3.9 所示。



Note

图 3.9 不用乘法运算实现 2×16

实现过程

- (1) 在 Eclipse 中新建项目 017，在项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建 Example 类，在该类的主方法中接收用户输入的一个整数，然后对该整数执行位运算中的左移操作，并输出运算结果。关键代码如下：

```

import java.util.Scanner;
public class Example {
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);           //创建扫描器
        System.out.println("请输入一个整数");
        long number = scan.nextLong();                  //获取输入的整数
        System.out.println("你输入的数字是: "+number);
        System.out.println("该数字乘以 2 的运算结果为: "+(number<<1));
        System.out.println("该数字乘以 4 的运算结果为: "+(number<<2));
        System.out.println("该数字乘以 8 的运算结果为: "+(number<<3));
        System.out.println("该数字乘以 16 的运算结果为: "+(number<<4));
    }
}
  
```

技术要点

本实例的关键技术就是左移运算。如果一个整数左移运算 n 位，就相当于这个整数乘以 2 的 n 次方。

例如， $2 \ll 4$ （即 2 左移 4 位）相当于 2^4 ，也就是 16。

实例 018 实现两个变量的互换（不借助第 3 个变量）

（实例位置：配套资源\SL\03\018）

实例说明

变量的互换常见于数组排序算法中，当判断两个数组元素需要互换时，将创建一个临时变量来共同完成互换，临时变量的创建增加了系统资源的消耗。如果需要交换的是两个整数类型的变量，那么可以使用更高效的方法。本实例演示了如何不借助临时变量（第 3 个变量）实现两个整数类型变量的高效互换。实例的运行效果如图 3.10 所示。

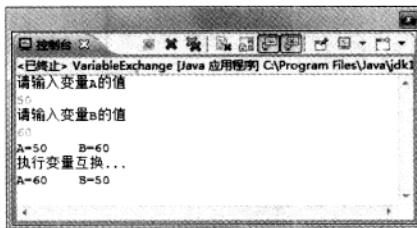


图 3.10 不借助第 3 个变量实现两个变量互换

实现过程

- (1) 在 Eclipse 中新建项目 018，在项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建 VariableExchange 类，在该类的主方法中创建扫描器对象接收用户输入两个变量值，然后通过位运算中的异或运算符“^”实现两个变量的互换。关键代码如下：

```
import java.util.Scanner;
public class VariableExchange {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in); // 创建扫描器
        System.out.println("请输入变量 A 的值"); // 接收第 1 个变量值
        long A = scan.nextLong();
        System.out.println("请输入变量 B 的值"); // 接收第 2 个变量值
        long B = scan.nextLong();
        System.out.println("A=" + A + "\tB=" + B);
        System.out.println("执行变量互换...");
        A = A ^ B; // 执行变量互换
        B = B ^ A; // 执行变量互换
        A = A ^ B; // 执行变量互换
        System.out.println("A=" + A + "\tB=" + B);
    }
}
```

技术要点

本实例的关键技术是异或运算，可以参考实例 015。

第 4 章

流程控制

本章读者可以学到如下实例：

- ▶ 实例 019 判断某一年是否为闰年
- ▶ 实例 020 验证登录信息的合法性
- ▶ 实例 021 为新员工分配部门
- ▶ 实例 022 用 switch 语句根据消费金额计算折扣
- ▶ 实例 023 判断用户输入月份的季节
- ▶ 实例 024 使用 while 循环语句与自增运算符循环遍历数组
- ▶ 实例 025 使用 for 循环输出杨辉三角形
- ▶ 实例 026 使用嵌套循环在控制台上输出九九乘法表
- ▶ 实例 027 使用 while 循环计算 $1+1/2!+1/3!+\dots+1/20!$
- ▶ 实例 028 使用 for 循环输出空心的菱形
- ▶ 实例 029 终止循环体
- ▶ 实例 030 循环体的过滤器



实例 019 判断某一年是否为闰年

(实例位置: 配套资源\SL\04\019)



实例说明

Note

地球绕太阳一圈称之为一年，所用时间是 365 天 5 小时 48 分 46 秒，取 365 天为一年，4 年将多出 23 小时 15 分 6 秒，将近一天，所以 4 年设一个闰日（2 月 29 日），这年称为闰年。本实例将要求用户输入年份，并判断输入的年份是否为闰年。实例的运行效果如图 4.1 所示。

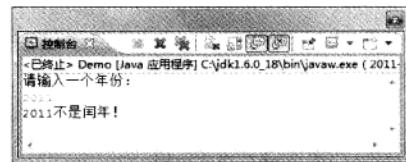


图 4.1 判断某一年是否为闰年

实现过程

(1) 在 Eclipse 中创建项目 019，并在该项目中创建 com.mingrisoft 包。

(2) 在 com.mingrisoft 包中创建类文件，名称为 Demo。在该类的主方法中接收用户输入的一个整数年份，然后通过闰年计算公式，判断这个年份是否为闰年，并在控制台输出判断结果。关键代码如下：

```
package com.mingrisoft;
import java.util.Scanner;

public class Demo {
    /**
     * @param args
     */
    public static void main(String[] args) { //主方法
        Scanner scan = new Scanner(System.in);
        System.out.println("请输入一个年份："); //向控制台输出一个提示信息
        long year;
        try {
            year = scan.nextLong();
            if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) { //是闰年
                System.out.print(year + "是闰年！");
            } else { //不是闰年
                System.out.print(year + "不是闰年！");
            }
        } catch (Exception e) {
            System.out.println("您输入的不是有效的年份！");
        }
    }
}
```

指点迷津：

java.util 包的 Scanner 类是一个用于扫描输入文本的简单文本扫描器，可以用这个类从控制台写入数据。该类的 nextLong() 方法可以将输入信息扫描为一个 long 型的数据，如果输入的信息不能被成功转换为 long 型，将抛出 java.util.InputMismatchException 异常。



多学两招：

三元运算符（? :）是 if…else 语句的一个简洁写法，可以根据需求来决定使用哪种。前者常用于赋值判断，后者常用于业务流程。

技术要点

本实例主要的技术就是应用 if 语句判断闰年。判断一个年份是否为闰年，要满足两个条件：一个是能被 4 整除但不能被 100 整除，另一个是能被 400 整除。

判断闰年的公式用 Java 语法实现的格式如下：

```
year % 4 == 0 && year % 100 != 0 || year % 400 == 0
```



Note

实例 020 验证登录信息的合法性

(实例位置：配套资源\SL\04\020)

实例说明

大多系统的登录模块都会接收用户通过键盘输入的登录信息，这些登录信息将会被登录模块验证，如果使用的是指定的用户名与密码，则允许用户登录；否则将用户拒之门外。本实例将通过 if…else 语句进行多条件判断实现登录信息验证。实例的运行效果如图 4.2 所示。

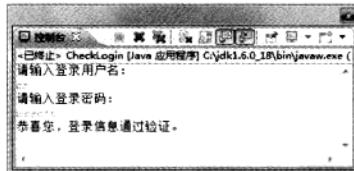


图 4.2 输入合法登录信息的效果

实现过程

- (1) 在 Eclipse 中创建项目 020，并在该项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建类文件，名称为 CheckLogin，并在该类的主方法中接收用户输入的登录用户名与登录密码，然后通过 if 条件语句分别判断用户名与密码，并输出登录验证结果。关键代码如下：

```
import java.util.Scanner;
public class CheckLogin {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in); // 创建扫描器
        System.out.println("请输入登录用户名：");
        String username = scan.nextLine(); // 接收用户输入登录名
        System.out.println("请输入登录密码：");
        String password = scan.nextLine(); // 接收用户输入登录密码
        if (!username.equals("mr")) { // 判断用户名合法性
            System.out.println("用户名非法。");
        } else if (!password.equals("mrsoft")) { // 判断密码合法性
            System.out.println("登录密码错误。");
        } else { // 通过以上两个条件判断则默认通过登录验证
            System.out.println("恭喜您，登录信息通过验证。");
        }
    }
}
```



多学两招：

字符串属于对象而非基本数据类型，不能使用“==”来判断两个字符串是否相当，所以需要通过 equals()方法来判断两个字符串内容是否相同，正如本实例对用户名和密码的判断那样。使用“==”判断的将是两个字符串对象的内存地址，而非字符串内容。



Note

技术要点

本实例应用的主要技术点就是 if…else 语句。if…else 语句的语法格式比较灵活，可以是简单的 if…else 格式，也可以是以下格式：

```
if(表达式){  
    若干语句  
}else if{  
    若干语句  
}else{  
    若干语句  
}
```

本实例中应用的就是这种格式。

实例 021 为新员工分配部门

(实例位置：配套资源\SL\04\021)

实例说明

本实例根据用户输入的信息确定员工应该分配到哪个部门。实例中需要根据用户输入的信息进行多条件判断，所以采用了 switch 语句。实例的运行效果如图 4.3 所示。

实现过程

(1) 在 Eclipse 中创建项目 021，并在该项目中创建 com.mingrisoft 包。

(2) 在 com.mingrisoft 包中创建类文件，名称为 Example，并在该类的主方法中创建标准输入流的扫描器，通过扫描器获取人事部门输入的姓名与应聘编程语言，然后根据每个语言对应的哈希码来判断分配部门。关键代码如下：

```
import java.util.Scanner;  
public class Example {  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in); // 创建扫描器  
        System.out.println("请输入新员工的姓名：");  
        String name = scan.nextLine(); // 接收员工名称  
        System.out.println("请输入新员工应聘的编程语言：");  
        String language = scan.nextLine(); // 接收员工应聘的编程语言  
        // 根据编程语言确定员工分配的部门  
        Switch (language.hashCode()) {  
            case 3254818: // Java 的哈希码  
            case 2301506: // Java 的哈希码
```

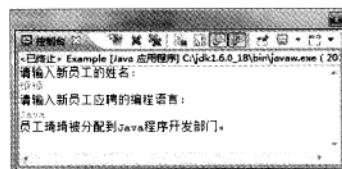


图 4.3 为新员工分配部门



Note

```
case 2269730: //JAVA 的哈希码
    System.out.println("员工"+name+"被分配到 Java 程序开发部门。");
    break;
case 3104: //c# 的哈希码
case 2112: //C# 的哈希码
    System.out.println("员工"+name+"被分配到 C# 项目维护组。");
    break;
case -709190099: //asp.net 的哈希码
case 955463181: //Asp.net 的哈希码
case 9745901: // ASP.NET 的哈希码
    System.out.println("员工"+name+"被分配到 Asp.net 程序测试部门。");
    break;
default:
    System.out.println("本公司不需要" + language + "语言的程序开发人员。");
}
```

多学两招：

在 switch 语法中每个 case 关键字可以作为一个条件分支，但是对于多个条件采取相同业务处理的情况，可以把多个 case 分支关联在一起，省略它们之间的 break 语句，而在最后一个相同的 case 分支中实现业务处理并执行 break 语句，就像本实例中应用的那样。

技术要点

本实例的技术要点在于 switch 多分支语句的使用。该语句只支持对常量的判断，而常量又只能是 Java 的基本数据类型，虽然在以后的 JDK 版本中可能支持对 String 类的字符串对象进行判断，但是就目前项目的需求也有很多需要对字符串进行多条件判断的情况。本实例采取的是对字符串的哈希码进行判断，也就是把 String 类的 hashCode()方法返回值作为 switch 语法的表达式，case 关键字之后跟随的是各种字符串常量的哈希码整数值。

实例 022 用 switch 语句根据消费金额计算折扣

(实例位置: 配套资源\SL\04\022)

实例说明

俗话说“商场如战场”，各大商家为了笼络有限的顾客，经常会打出各种各样的促销手段。例如，会员折扣制度，即对会员的消费金额进行累加，当超过一定数额时，可以享受相应的折扣。累计消费金额越高，享受的折扣越多。本实例将应用 switch 语句计算累计消费金额达到一定数额时，享受不同的折扣价格。实例的运行效果如图 4.4 所示。

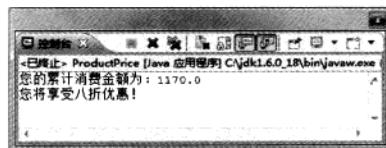


图 4.4 根据消费金额计算折扣

实现过程

- (1) 在 Eclipse 中创建项目 022，并在该项目中创建 com.mingrisoft 包。



(2) 在 com.mingrisoft 包中创建类文件，名称为 ProductPrice，并在该类的主方法中实现本实例的业务代码。在该方法中，首先假设一个用户消费总额的变量 money，并初始化一个折扣变量 rebate，然后经过运算来获得用户等级，对不同的等级给予不同的折扣优惠。关键代码如下：

```
public class ProductPrice {  
    public static void main(String[] args) {  
        float money = 1170; //金额  
        String rebate = ""; //折扣  
        if (money > 200) {  
            int grade = (int) money / 200; //等级  
            switch (grade) { //根据等级计算折扣比例  
                case 1:  
                    rebate = "九五折";  
                    break;  
                case 2:  
                    rebate = "九折";  
                    break;  
                case 3:  
                    rebate = "八五折";  
                    break;  
                case 4:  
                    rebate = "八三折";  
                    break;  
                case 5:  
                    rebate = "八折";  
                    break;  
                case 6:  
                    rebate = "七八折";  
                    break;  
                case 7:  
                    rebate = "七五折";  
                    break;  
                case 8:  
                    rebate = "七三折";  
                    break;  
                case 9:  
                    rebate = "七折";  
                    break;  
                case 10:  
                    rebate = "六五折";  
                    break;  
                default:  
                    rebate = "六折";  
            }  
        }  
        System.out.println("您的累计消费金额为：" + money); //输出消费金额  
        System.out.println("您将享受" + rebate + "优惠！"); //输出折扣比例  
    }  
}
```



Note



脚下留神：

在程序开发中经常使用的都是正数，负数因为使用得少，常常被忽略，例如“N%2=1”本来是用来计算数字 N 是否为奇数的，但是开发者没有考虑到负数的情况，从而导致这个算法的失败，因为任何负数应用这个算法都会等于-1。

技术要点

本实例主要应用 switch 语句实现。switch 语句是多分支选择语句，常用来根据表达式的值选择要执行的语句。switch 语句的基本语法格式如下：

```
switch(表达式){  
    case 常量表达式 1: 语句序列 1  
        [break;]  
    case 常量表达式 2: 语句序列 2  
        [break;]  
    ...  
    case 常量表达式 n: 语句序列 n  
        [break;]  
    default: 语句序列 n+1  
        [break;]  
}
```

switch 语句的参数说明如表 4.1 所示。

表 4.1 switch 语句的参数说明

参数名称	参数描述
表达式	必要参数。可以是任何 byte、short、int 和 char 类型的变量
常量表达式 1	如果有 case 出现，则为必要参数。该常量表达式的值必须是一个与表达式数据类型相兼容的值
语句序列 1	可选参数。一条或多条语句，但不需要大括号。当表达式的值与常量表达式 1 的值匹配时执行；如果不匹配则继续判断其他值，直到常量表达式 n
常量表达式 n	如果有 case 出现，则为必要参数。该常量表达式的值必须是一个与表达式数据类型相兼容的值
语句序列 n	可选参数。一条或多条语句，但不需要大括号。当表达式的值与常量表达式 n 的值匹配时执行
break	可选参数。用于跳出 switch 语句
default	可选参数。如果没有该参数，则当所有匹配不成功时，将不会执行任何操作
语句序列 n+1	可选参数。如果没有与表达式的值相匹配的 case 常量时，将执行语句序列 n+1

实例 023 判断用户输入月份的季节

(实例位置：配套资源\SL\04\023)

实例说明

一年有四季，每季 3 个月。其中，12 月、1 月和 2 月为冬季，3 月、4 月和 5 月为春季，6 月、7 月和 8 月为夏季，9 月、10 月和 11 月为秋季。本实例将根据用户输入的月份来判断季节，



Note



这是一个最典型的演示 switch 语法的例子，通过这个例子，读者可以完全掌握 switch 语句的用法与技巧。实例的运行效果如图 4.5 所示。



实现过程

Note

(1) 在 Eclipse 中创建项目 023，并在该项目中创建 com.mingrisoft 包。

(2) 在 com.mingrisoft 包中创建类文件，名称为 JudgeMonth，并在该类的主方法中创建扫描器接收用户输入的月份数字，然后判断该月份属于哪个季节并输出到控制台，对于非法月份也要给出提示。关键代码如下：

```
import java.util.Scanner;
public class JudgeMonth {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in); // 创建扫描器
        // 提示用户输入月份
        System.out.println("请输入一个月份，我能告诉你它属于哪个季节。");
        int month = scan.nextInt(); // 接收用户输入
        switch (month) { // 判断月份属于哪个季节
            case 12:
            case 1:
            case 2:
                System.out.print("您输入的月份属于冬季。");
                break;
            case 3:
            case 4:
            case 5:
                System.out.print("您输入的月份属于春季");
                break;
            case 6:
            case 7:
            case 8:
                System.out.print("您输入的月份属于夏季");
                break;
            case 9:
            case 10:
            case 11:
                System.out.print("您输入的月份属于秋季");
                break;
            default:
                System.out.print("你那有" + month + "月份吗？");
        }
    }
}
```

脚下留神：

switch 语句的每个 case 关键字都用于判断一个常量并做出相应的业务处理，熟练掌握 switch 语句之后可以组合多个 case 来完成多条件的处理，就是多个常量结果执行相同的业务处理，如本实例中应用的格式。

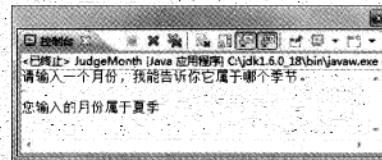


图 4.5 判断用户输入月份的季节



技术要点

本实例应用的主要技术是 switch 语句，关于 switch 语句的详细介绍请参见实例 022。

实例 024 使用 while 循环语句与自增运算符循环遍历数组

(实例位置：配套资源\SL\04\024)



Note

实例说明

在多数情况下，遍历数组都是使用 for 循环语句实现。其实应用 while 循环语句和自增运算符也可实现遍历数组。本实例将利用自增运算符结合 while 循环语句实现数组的遍历，并将遍历结果输出到控制台。实例的运行效果如图 4.6 所示。

实现过程

(1) 在 Eclipse 中创建项目 024，并在该项目中创建 com.mingrisoft 包。

(2) 在 com.mingrisoft 包中创建类文件，名称为 ErgodicArray，并在该类的主方法中创建一个鸟类数组，然后创建一个索引变量，这个变量用于指定数组下标，随着该索引的递增，while 循环会逐步获取每个数组的元素并输出到控制台中。关键代码如下：

```
public class ErgodicArray {
    public static void main(String[] args) {
        // 创建鸟类数组
        String[] aves = new String[] { "白鹭", "黄鹂", "鹦鹉", "乌鸦", "喜鹊", "布谷鸟", "斑鸠",
            "百灵鸟" };
        int index = 0; // 创建索引变量
        System.out.println("我的花园里有很多鸟，大约包括：");
        while (index < aves.length) { // 遍历数组
            System.out.println(aves[index++]); // 自增索引值
        }
    }
}
```

指点迷津：

自增自减运算符分前置和后置两种。其中，前置运算如“`++index`”，会先将 `index` 的值递增，然后再使用递增后的值；而后置运算如“`index++`”，会首先使用该变量的值，然后再把变量值递增。

技术要点

本实例应用的主要技术是 while 循环语句和自增运算符。while 循环语句的基本语法格式如下：

```
while(条件表达式){
    语句序列
}
```

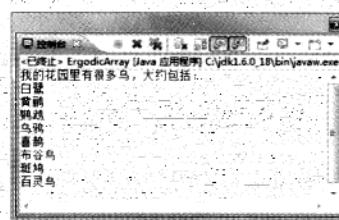


图 4.6 输出数组的遍历结果



参数说明

- 条件表达式：决定是否进行循环的表达式，其结果为 boolean 类型，也就是其结果只能是 true 或 false。
- 语句序列：也就是循环体，在条件表达式的结果为 true 时，重复执行。

**Note**

实例 025 使用 for 循环输出杨辉三角形

(实例位置：配套资源\SL\04\025)

实例说明

杨辉三角形由数字排列，可以把它看作一个数字表，其基本特性是两侧数值均为 1，其他位置的数值是其正上方的数值与左上角数值之和。本实例将使用 for 循环输出包括 10 行内容的杨辉三角形。实例的运行效果如图 4.7 所示。

The screenshot shows a Windows command-line interface window titled '控制台'. The title bar also includes the text '已停止 - YanghuiTriangle [Java 应用程序] C:\jdk1.6.0_18\bin\javaw.exe (2011-5-10 下午05:04:08)'. The main window displays a 10x10 grid of integers representing Pascal's triangle. The first few rows are as follows:

1	1								
1	2	1							
1	3	3	1						
1	4	6	4	1					
1	5	10	10	5	1				
1	6	15	20	15	6	1			
1	7	21	35	35	21	7	1		
1	8	28	56	70	56	28	8	1	
1	9	36	84	126	126	84	36	9	1

图 4.7 使用 for 循环输出杨辉三角形

实现过程

- (1) 在 Eclipse 中创建项目 025，并在该项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建类文件，名称为 YanghuiTriangle，并在该类的主方法中创建一个二维数组，并指定二维数组的第一维长度，这个数组用于存放杨辉三角形的数值表，通过双层 for 循环来实现第二维数组的长度，然后计算整个数组的每个元素的值。关键代码如下：

```
public class YanghuiTriangle {  
    public static void main(String[] args) {  
        int triangle[][]=new int[10][]; //创建二维数组  
        //遍历二维数组的第一层  
        for (int i = 0; i < triangle.length; i++) {  
            triangle[i]=new int[i+1]; //初始化第二层数组的大小  
            //遍历第二层数组  
            for(int j=0;j<=i;j++){  
                if(i==0||j==0||j==i){  
                    triangle[i][j]=1; //将两侧的数组元素赋值为 1  
                }else{ //其他数值通过公式计算  
                    triangle[i][j]=triangle[i-1][j]+triangle[i-1][j-1];  
                }  
            System.out.print(triangle[i][j]+" "); //输出数组元素  
        }  
    }  
}
```



```
        }  
        System.out.println(); //换行  
    }  
}
```

指点迷津：

在创建二维数组时，其第一维的长度即是要输出杨辉三角形的行数。例如，在本实例中，要输出 10 行的杨辉三角形就可以将其第一维的长度设置为 10。

指点迷津；

Java语言中的二维数组其实是每个元素都是一个一维数组的一维数组，所以第二维的长度可以任意，就像本实例中那样。这比其他语言的数组更灵活，而且多维数组也是如此。

技术要点

本实例应用的主要技术是应用 for 循环语句根据杨辉三角形的公式遍历二维数组。杨辉三角形的公式包括两部分，一部分是两侧数值都是 1，也就是说二维数组的 triangle[0][0]、triangle[i][0] 或者 triangle[i][i] 的元素值为 1，另一部分是其他位置的数值是其正上方的数值与左上角数值之和，也就是 triangle[i][j]=triangle[i-1][j]+triangle[i-1][j-1]。

指点迷津：

在二维数组中，第 1 个下标值代表的是行数，第 2 个下标值代表的是列数，即 triangle[i][0] 代表的是第 i 行第 0 列的元素。

实例 026 使用嵌套循环在控制台上输出九九乘法表

(実測位置: 配達資源\SI\04\026)

实例说明

对于九九乘法表读者都不会陌生，几乎每个人都能倒背如流。那么如何通过嵌套循环，在控制台上输出九九乘法表呢？这个内容在掌握了 for 循环的嵌套后，就不难实现。本实例将应用嵌套的 for 循环实现在控制台上输出九九乘法表。实例的运行效果如图 4.8 所示。



图 4.8 应用嵌套的 for 循环输出九九乘法表



实现过程

- (1) 在 Eclipse 中创建项目 026，并在该项目中创建 com.mingrisoft 包。
 (2) 在 com.mingrisoft 包中创建类文件，名称为 MultiplicationTable，并在该类的主方法中通过双层 for 循环输出九九乘法表。关键代码如下：

```
public class MultiplicationTable {
    public static void main(String[] args) {
        for(int i=1;i<=9;i++){
            //循环控制变量从 1 遍历到 9
            for(int j=1;j<=i;j++){
                //第二层循环控制变量与第一层最大索引相等
                System.out.print(j+"*"+i+"="+i*j+"\t");
                //输出计算结果但不换行
            }
            System.out.println();
            //在外层循环中换行
        }
    }
}
```

脚下留神：

在上面的代码中，在内层循环输出计算结果时，应用的是 System.out 对象的 print() 方法输出，应用该方法不换行，在内层循环结束后，再应用 System.out 对象的 println() 方法换行。

指点迷津：

循环语句可用于完成复杂的运算，也可以用于控制程序的递归流程，而多层循环可以实现更加复杂的业务逻辑，是学习编程必须掌握的一种应用。在处理有规则的大量数据时，应该考虑使用多层循环来优化程序代码，但是建议添加详细的代码注释，便于以后的维护与修改工作。

技术要点

要在控制台上输出九九乘法表，可以通过嵌套的 for 循环来实现，即创建双层的 for 循环。第一层 for 循环，也称外循环，用于控制表格的行；第二层 for 循环，也称内循环，用于控制表格的列。其中，第一层 for 循环的控制变量的最大值是 9，第二层 for 循环的控制变量的最大值要等于行数的最大值。然后输出内层与外层循环控制变量的乘积，即可实现九九乘法表。

实例 027 使用 while 循环计算 $1+1/2!+1/3!+\dots+1/20!$

(实例位置：配套资源\SL\04\027)

实例说明

本实例在计算阶乘的算法之上应用 while 循环语句计算 $1+1/2!+1/3!+\dots+1/20!$ 的和。如果使用基本数据类型 double 是无法精确地显示运算结果的，所以本实例使用了 BigDecimal 类的实例来完成这个运算。实例的运行效果如图 4.9 所示。



```

控制台 C:\jdk1.6.0_18\bin\javaw.exe (2011-5-11 上午10:32:04)
+已终止> Example (1) [Java 应用程序] C:\jdk1.6.0_18\bin\javaw.exe (2011-5-11 上午10:32:04)
1+1 / 2! +1 / 3! ... 1 / 20! 的计算结果等于:
1.718281826459045223672588625247325636348672462695581044481308321361012393467199
8641438311024379710974822620657493330665752903534700022508204525385549479876951
2924935163613915851424115809152330833591985613426109509774258229517247637411337
44312953455167334812919173837519915685570192090006173940177559703307592803194249
25748716392429788018965622511631047241914024775263254452027548064334842631749679
27093078963954272457795367333379155540025721955796334108908561918415272591238383
61068125695892585741135318759102712228192803080836192262375397641816524727180463
78704910462924352995976893461209516866339555490458432187283021232937484781081946
9522374995930233329983054713200991533366762525321300854523561569371243184556121
683188623896572782265052602992424431046018124286759305381313769541119226363806860
46919663567678071558475494384765625

```

图 4.9 用 while 循环计算 $1+1/2!+1/3! \cdots 1/20!$

指点迷津:

由于本实例的运行结果精度非常高，小数位数过长，默认情况下，运行结果只能显示单行的数字，所以需要对控制台进行折行设置。具体方法是，在控制台中单击鼠标右键，在弹出的快捷菜单中选择“首选项”命令，在弹出的“首选项”对话框中选中“固定宽度控制台”复选框，并设置每行的最大字符数，可以采用默认的 80 个，然后单击“确定”按钮即可。这时，再运行本实例，将显示如图 4.9 所示的运行效果。

实现过程

(1) 在 Eclipse 中创建项目 027，并在该项目中创建 com.mingrisoft 包。

(2) 在 com.mingrisoft 包中创建类文件，名称为 Example，并在该类的主方法中保存总和的 sum 变量和计算阶乘的 factorial 变量，为保证计算结果的精度，这两个变量都是 BigDecimal 类的实例对象，然后通过 while 实现 20 次循环，并完成计算。关键代码如下：

```

import java.math.BigDecimal;
public class Example {
    public static void main(String args[]) {
        BigDecimal sum = new BigDecimal(0.0);                                //和
        BigDecimal factorial = new BigDecimal(1.0);                            //阶乘项的计算结果
        int i = 1;                                                            //循环增量
        while (i <= 20) {
            sum = sum.add(factorial);                                         //累加各项阶乘的和
            ++i;                                                               //i 加 1
            factorial = factorial.multiply(new BigDecimal(1.0 / i));        //计算阶乘项
        }
        System.out.println("1+1 / 2!+1 / 3! ... 1 / 20! 的计算结果等于: " + sum); //输出计算结果
    }
}

```

脚下留神：

对于高精度要求或者运算数较大的计算，应该使用 BigDecimal 类实现，否则 Java 基本类型的数据无法保证浮点数的精度，也无法对超出其表示范围的数字进行运算。

技术要点

本实例主要应用了 while 循环和 BigDecimal 类的实例对象的相关方法实现计算 $1+1/2!+1/3! \cdots$





1/20!的和。BigDecimal 类型的数字可以用来做超大的浮点数的运算，如加、减、乘、除等。使用 BigDecimal 对象的 add()方法可以实现加法运算，使用 multiply()方法可以实现乘法运算。

实例 028 使用 for 循环输出空心的菱形

(实例位置：配套资源\SL\04\028)



Note

实例说明

本实例在输出菱形的基础上加大难度，输出空心的菱形图案，这在等级考试与公司面试时也出现过类似题目，实例目的在于要求熟练掌握 for 循环的嵌套使用。实例的运行效果如图 4.10 所示。

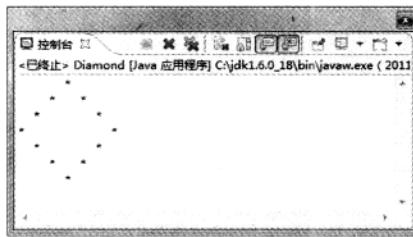


图 4.10 使用 for 循环输出空心的菱形

实现过程

(1) 在 Eclipse 中创建项目 028，并在该项目中创建 com.mingrisoft 包。

(2) 在 com.mingrisoft 包中创建类文件，名称为 Diamond，并在该类的主方法中调用 printHollowRhombus()方法完成 7 行的空心菱形输出。其中 printHollowRhombus()方法是实例中自定义的，该方法使用两个双层 for 循环分别输出菱形的上半部分与下半部分。关键代码如下：

```
public class Diamond {  
    public static void main(String[] args) {  
        printHollowRhombus(7);  
    }  
    public static void printHollowRhombus(int size) {  
        if (size % 2 == 0) {  
            size++;  
        }  
        for (int i = 0; i < size / 2 + 1; i++) {  
            for (int j = size / 2 + 1; j > i + 1; j--) {  
                System.out.print(" ");  
            }  
            for (int j = 0; j < 2 * i + 1; j++) {  
                if (j == 0 || j == 2 * i) {  
                    System.out.print("* ");  
                } else {  
                    System.out.print(" ");  
                }  
            }  
        }  
    }  
}
```



```

        System.out.println("");
    }
    for (int i = size / 2 + 1; i < size; i++) {
        for (int j = 0; j < i - size / 2; j++) {
            System.out.print(" ");
        }
        for (int j = 0; j < 2 * size - 1 - 2 * i; j++) {
            if (j == 0 || j == 2 * (size - i - 1)) {
                System.out.print("* ");
            } else {
                System.out.print(" ");
            }
        }
        System.out.println("");
    }
}

```



Note

脚下留神：

for 循环中有 3 个表达式，这 3 个表达式都是可选的，也就是说 for 循环可以没有表达式。例如 for(;;) 这样的 for 循环将是一个无限循环，读者在使用 for 循环时应注意避免无限循环。

技术要点

本实例应用的主要技术是 for 循环语句的嵌套和 if…else 语句。其中，if…else 语句用于控制菱形中心位置不输出*号，即输出空心菱形。

实例 029 终止循环体

(实例位置：配套资源\SL\04\029)

实例说明

循环用于复杂的业务处理，可以提高程序的性能和代码的可读性，但是循环中也有特殊情况，例如由于某些原因需要立刻中断循环去执行下面的业务逻辑，这时就可以使用 break 语句实现。本实例将实现应用 break 语句中断单层循环和中断双层 for 循环。实例的运行效果如图 4.11 所示。

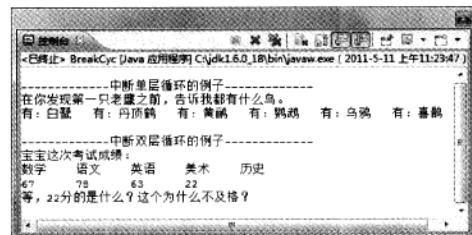


图 4.11 终止循环体



实现过程

- (1) 在 Eclipse 中创建项目 029，并在该项目中创建 com.mingrisoft 包。
(2) 在 com.mingrisoft 包中创建类文件，名称为 BreakCyc。在该类的主方法中，首先创建一个字符串数组，然后在使用 for 循环遍历时判断如果发现数组中包含字符串“老鹰”则立刻中断循环。最后再创建一个整数类型的二维数组，并使用双层 for 循环遍历，当发现第一个小于 60 的数组元素时，立刻中断整个双层循环，而不是内层循环。关键代码如下：

```
public class BreakCyc {  
    public static void main(String[] args) {  
        System.out.println("\n-----中断单层循环的例子-----");  
        //创建数组  
        String[] array = new String[] { "白鹭", "丹顶鹤", "黄鹂", "鹦鹉", "乌鸦", "喜鹊",  
            "老鹰", "布谷鸟", "老鹰", "灰纹鸟", "老鹰", "百灵鸟" };  
        System.out.println("在你发现第一只老鹰之前，告诉我都有什么鸟。");  
        for (String string : array) {  
            //for 遍历数组  
            if (string.equals("老鹰"))  
                //如果遇到老鹰  
                break;  
            System.out.print("有：" + string + " ");  
            //否则输出数组元素  
        }  
  
        System.out.println("\n\n-----中断双层循环的例子-----");  
        //创建成绩数组  
        int[][] myScores = new int[][] { { 67, 78, 63, 22, 66 }, { 55, 68, 78, 95, 44 }, { 95, 97, 92, 93,  
            81 } };  
        System.out.println("宝宝这次考试成绩：\n 数学\t语文\t英语\t美术\t历史");  
        No1: for (int[] is : myScores) {  
            //遍历成绩表格  
            for (int i : is) {  
                System.out.print(i + "\t");  
                //输出成绩  
                if (i < 60) {  
                    //如果中途遇到不及格的，立刻中断所有输出  
                    System.out.println("\n 等， " + i + "分的是什么？这个为什么不及格？");  
                    break No1;  
                }  
            }  
            System.out.println();  
        }  
    }  
}
```

指点迷津：

充分利用循环可以提高程序的开发与执行效率，但是如果注重循环中的算法很容易导致程序的死循环，所以在循环体中要对可能出现的特殊情况使用 break 语句中断循环。

技术要点

本实例应用的主要技术是 break 语句。break 语句用于强行退出循环。使用带标签的 break 语句，可以强行退出多层循环。例如，要退出双层的 for 循环，可以使用下面的语法：

```
lable:  
for(元素变量 x: 遍历对象 obj){
```



```
for(元素变量 x1 : 遍历对象 obj1) {
    引用了 x 的 Java 语句;
    if(条件表达式) {
        break lable;
    }
}
}
```

在上面的语法中，如果执行到 `break`，正常情况下应该结束内层循环去执行外层循环，但是由于 `break` 后带有标签，所以程序将结束标签处的外层循环。



Note

实例 030 循环体的过滤器

(实例位置：配套资源\SL\04\030)

实例说明

循环体中可以通过 `break` 语句中断整个循环，这增加了循环的控制能力，但是对于特殊情况还是不够，例如某些条件下需要放弃部分循环处理，而不是整个循环体。Java 提供了 `continue` 语句来实现这一功能，`continue` 语句可以放弃本次循环体的剩余代码，不执行它们而开始下一轮的循环。本实例利用 `continue` 语句实现了循环体过滤器，可以过滤“老鹰”字符串，并做相应的处理，但是放弃 `continue` 语句之后的所有代码。实例的运行效果如图 4.12 所示。

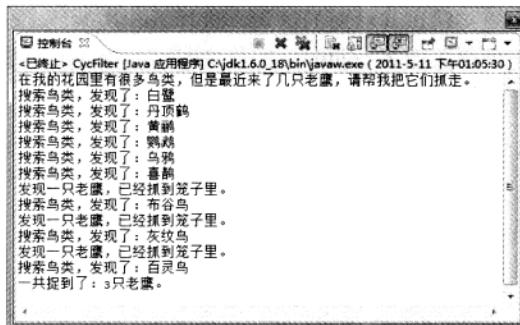


图 4.12 过滤“老鹰”字符串的结果

实现过程

- (1) 在 Eclipse 中创建项目 030，并在该项目中创建 com.mingrisoft 包。
- (2) 在 com.mingrisoft 包中创建类文件，名称为 CycFilter，并在该类的主方法中创建鸟类名称的字符串数组，其中包含多个“老鹰”字符串，然后通过 for 循环遍历该数组，在循环过程中如果遍历的数组元素是“老鹰”字符串，则输出发现老鹰的信息并过滤循环体之后的所有代码。关键代码如下：

```
package com.mingrisoft;
public class CycFilter {
    public static void main(String[] args) {
        // 创建数组
        String[] array = new String[] { "白鹭", "丹顶鹤", "黄鹂", "鹦鹉", "乌鸦", "喜鹊",
        "老鹰" };
        for (int i = 0; i < array.length; i++) {
            if ("老鹰".equals(array[i])) {
                System.out.println("发现一只老鹰，已经捉到笼子里。");
                continue;
            }
            System.out.println("搜索鸟类，发现了：" + array[i]);
        }
        System.out.println("一共捉到了：" + i + "只老鹰。");
    }
}
```



```
"老鹰", "布谷鸟", "老鹰", "灰纹鸟", "老鹰", "百灵鸟" };
System.out.println("在我的花园里有很多鸟类, 但是最近来了几只老鹰, 请帮我把它们
抓走。");
int eagleCount = 0;
for (String string : array) {
    if (string.equals("老鹰")) {
        System.out.println("发现一只老鹰, 已经抓到笼子里。");
        eagleCount++;
        continue; //中断循环
    }
    System.out.println("搜索鸟类, 发现了: " + string);
}
System.out.println("一共捉到了: " + eagleCount + "只老鹰。");
}
```

多学两招:

`break` 语句和 `continue` 语句都是对循环体的控制语句, 它们不仅应用于 `for` 循环, 在任何循环体中都可以使用这些语句, 灵活使用可以让循环实现更加复杂的运算和业务处理。

技术要点

本实例应用的主要技术是 `continue` 语句。`continue` 语句只能应用在 `for`、`while` 和 `do...while` 循环语句中, 用于让程序直接跳过其后面的语句, 进行下一次循环。`continue` 语句的语法比较简单, 只需要在循环体中使用关键字 `continue` 加分号即可。具体的语法格式如下:

```
continue;
```