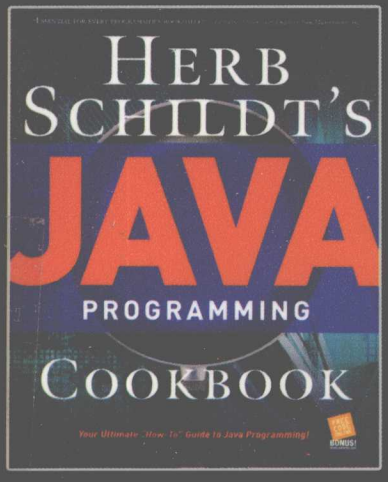
知名作者编程经验总结，成为Java高手的必备参考书

JAVA 编程Cookbook



最佳Java编程进阶书

HERB SCHILDT' S JAVA PROGRAMMING COOKBOOK

[美] Herbert Schildt 著
张君施 等译

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

Java 编程 Cookbook

Herb Schildt's Java Programming Cookbook

[美] Herbert Schildt 著

张君施 等译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内容简介

著名编程类作家 Herbert Schildt 的这本最新著作是如何在 Java 中执行各种编程任务的技术总结。本书采用知名的烹饪式图书架构，以配方的形式详细讲解和展示了如何完成一个特定的操作，这些配方包含以下方面的内容：使用字符串和正则表达式、文件处理、格式化数据、使用集合、applet 和 servlet、多线程编程、Swing 及其他流行的 Java 应用等。这些经过优化筛选的配方，在实际的编程中重复使用率相当高，通过所有这些配方的学习，读者将快速领会 Java 编程技巧，成为一名优秀的 Java 编程开发人员。

本书适合开发人员学习 Java 编程使用，也可作为相关人员的参考资料。

Herbert Schildt: **Herb Schildt's Java Programming Cookbook, First Edition**

ISBN-13: 978-0-07-226315-2, ISBN-10: 0-07-226315-6,Copyright © 2008 by The McGraw-Hill Companies, Inc

Original language published by The McGraw-Hill Companies, Inc. All rights reserved. No part of this publication may be reproduced or distributed in any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Simplified Chinese translation edition published by McGraw-Hill Education (Asia) Co. and Publishing House of Electronics Industry. Copyright © 2008

本书中文简体字翻译版由美国麦格劳－希尔教育出版(亚洲)公司授予电子工业出版社。未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 McGraw-Hill 公司激光防伪标签，无标签者不得销售。

版权贸易合同登记号　图字：01-2008-2751

图书在版编目(CIP)数据

Java 编程 Cookbook/(美)希尔特(Schildt, H.)著；张君施等译. —北京：电子工业出版社，2008.7

书名原文：Herb Schildt's Java Programming Cookbook

ISBN 978-7-121-06362-6

Ⅰ. J…　Ⅱ. ①希…②张…　Ⅲ. JAVA 语言－程序设计　Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2008)第 097627 号

责任编辑：谭海平　朱道立

印　　刷：北京市天竺颖华印刷厂

装　　订：三河市金马印装有限公司

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

开　　本：787×1092　1/16　印张：25.5　字数：653 千字

印　　次：2008 年 7 月第 1 次印刷

定　　价：45.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010)88254888。

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。

服务热线：(010)88258888。

前　言

多年来，许多朋友和读者都在问我，什么时候为 Java 写一本烹饪式的书(cookbook)，以分享我在编程时所使用的一些技术和方法？其实我有这个想法也由来已久，但实在是无法从我本已繁忙的写作计划中挤出时间来完成它。因为许多读者都知道，我的著作涉及编程的许多方面，尤其关注的是 Java，C/C++ 及 C#。由于这些语言的版本更新周期非常快，所以我必须几乎将全部精力花在更新我的著作上，以覆盖这些语言的最新版本。有幸的是，2007 年初，我终于有片刻闲暇，能够集中精力写作这本 Java 的“烹饪书”了。必须承认的是，写作这本书很快就成为了我最高兴的工作之一。

以传统的烹饪式图书格式为基础，这本书提炼出了许多通用技术的本质特性，并将它们汇集成一系列的逐步讲解的“配方”。每个配方都描述一组关键“配料”，比如类、接口和方法。然后，书中展示了将这些成分集成到一个代码序列中的步骤，以获得欺望的结果。这种组织便于查找所感兴趣的技术，也使得将技术付诸实践变得简单。

实际上，“付诸实践”是本书的一个重要部分。我相信一本好的编程类图书应包含两个因素：扎实的理论和实际的应用。配方中的逐步指导和讨论提供了理论。为了将这些理论用于实践，每个配方都包含一个完整的代码例子。通过具体的、明确的方法，例子演示了如何应用配方。换句话说，例子消除了“猜想的工作”，并由此节省了时间。

尽管不可能将某个人所期望的每一个配方都包含在一本菜谱式图书中(可能的配方数量几乎是无限的)，但我尽量涉及较广的主题范围。挑选配方的标准会在第 1 章详细讨论。简而言之，我所挑选的配方对多数程序员都是有用的，或者是回答那些经常被问及的问题。尽管有这些标准，但要对它们做出取舍，仍然是困难的。这是写作本书时遇到的最大挑战。根本而言，它们体现在经验、判断力和直觉上。乐观地估计，本书中所包含的内容能满足每一位程序员的口味。

Web 上的例子代码

本书中所有例子的源代码都可以从 www.osborne.com 免费下载。

Herbert Schildt 的更多著作

本书只不过是 Herbert Schildt 众多编程类著作中的一本，以下是你所感兴趣的其他一些图书。要想学习更多的 Java 知识，我们推荐：

Java: The Complete Reference[①]
Java: A Beginner's Guide
The Art of Java
Swing: A Beginner's Guide

要学习 C++，你会发现如下的这些图书尤其有用：

C++: The Complete Reference

① 本书已由电子工业出版社翻译出版，书名：《Java 完全手册》(第 7 版)，书号：978-7-121-12345-1。

C++ : A Beginner's Guide

C++ From the Ground Up

STL Programming From the Ground Up

The Art of C++

要学习 C#，推荐使用 Schildt 的如下著作：

C# : The Complete Reference

C# : A Beginner's Guide

如果想学习 C 语言，则下面的图书就是你所感兴趣的：

C : The Complete Reference

当需要可靠的答案时，可迅速求助于 Herbert Schildt 这位知名的编程权威。

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第1章 概 述

本书是如何在 Java 中执行各种编程任务的技术总结。正如书名所暗示的那样，它采用知名的烹饪式图书(cookbook)的形式，每一个配方(recipe)都展示了如何完成一个特定的操作。例如，书中对如下操作都开出了配方：从文件读字节、迭代集合、格式化数字数据、构造 Swing 组件、创建 servlet 等。烹饪书中的菜谱描述的是配料的组成以及为准备一道美味而必须完成的一系列步骤，与此相同，本书中描述的是一套关键的程序元素以及为完成编程任务而必须使用这些元素的步骤序列。

根本而言，本书的目的就是在程序开发过程中节省读者的时间和精力。许多编程任务都由一套 API 类、接口以及必须以某种顺序实现的方法组成。麻烦的是，有时我们并不知道该使用哪些 API 类，或者以什么顺序调用方法。我们不必费力地在众多 API 文档和在线教程中寻找完成某个任务的答案，只需看一看它的配方即可。每个配方展示的都是构思一个解决方案的一种方法，描述一些必要的元素以及使用它们的顺序。利用这些信息，我们能够设计出满足特定需求的方案。

1.1 本书的内容

没有一本烹饪书能穷尽天下美食，作者必须对其内容进行取舍。对这本“烹饪书”而言，也面临相同的情况。在挑选本书中的配方时，我将重点放在如下几个方面：

- 字符串处理(包括正则表达式)
- 文件处理
- 格式化数据
- applet 和 servlet
- Swing
- 集合框架
- 多线程

之所以选择这几个方面，是因为它们与绝大多数程序员都有关(我特意避免了一些专门化的主题，它们只适合有限的场合)。每一个这样的类别就成为了每一章的基础。除了与前面的主题相关的那些配方外，还有几个我希望包含在其中的配方，但由于它们不适合成为完整的一章，因此我将它们放在最后一章中。

当然，挑选主题只不过是精选过程的开始，在每个类别中，还必须决定应该包含哪些内容和舍弃哪些内容。一般而言，如果某个配方满足如下两个标准，则会将它包含进来：

1. 技术对绝大多数程序员都是有用的。
2. 它提供经常被问及的编程问题的答案。

第一个标准几乎不言自明，它基于我的经验。书中所包含的配方描述如何完成一系列的任务，它们在创建 Java 应用程序时经常会遇到。其中的一些配方展示的是一般性概念，可改写它们，以解决一些不同类型的问题。例如，第 2 章展示的是使用正则表达式 API 的配方，可用于从字符串中查找和抽取子串。这个通用过程在几种上下文中都是有用的，比如在一个句子中查找电子邮

件地址或一个电话号码，或者从一个数据库查询中抽取一个关键字。其他配方描述的是更为明确但广泛使用的技术。例如，第 4 章展示的是如何用 SimpleDateFormat 格式化时间和日期。

第二个标准基于我作为编程类图书的作者的经验。在我写作的许多年中，读者问过我无数的“如何做”的问题。这些问题几乎涉及 Java 编程的每个方面，从最简单的到十分复杂的都有。但是，我发现问题的一个核心会一次又一次地出现。这里是一个示例：“该如何格式化输出?”下面是另一个：“该如何压缩文件?”还有许多其他的问题。在 Web 上的各种程序员论坛中，相同类型的问题也经常出现。我利用这些经常被问到的“如何做”的问题来指导我对配方的选择。

书中的这些配方横跨各种技能水平。一些演示的是基本技术，比如从文件中读字节，或者创建一个 Swing JTable。而另一些是更高级的技术，比如创建一个 servlet，或者使用反射来在运行时实例化一个对象。因此，对单一的配方而言，难度水平可从相对容易变化到相当高级。当然，只要你知道了如何使用这些配方，有关编程的多数事情就会变得容易，反之它们就是困难的。因此，如果某个配方是显而易见的，不要对它感到奇怪，这只不过意味着你已经知道了如何完成这个任务。

1.2 配方是如何组织的

书中的每个配方都遵循相同的格式，它包含如下部分：

- 对配方所解决的问题的描述。
- 配方所使用的关键配料的列表。
- 完成配方所必须的步骤。
- 对步骤的深度讨论。
- 实际使用配方的一个代码示例。
- 选择和替代方法，它们是构思一个解决方案的其他建议。

配方都从描述要完成的任务开始，而配方所使用的关键配料放在一个表格中，包括创建解决方法所要求的 API 类、接口及方法。当然，实际使用配方时可能意味着需要使用其他的元素，但关键配料对任务而言是手边最基础的。

然后，每个配方呈现的是总结这个过程的逐步执行的指令。跟在它们之后的是对这些步骤的深度讨论。多数情况下，这个总结是足够的，但如果需要，也会有些细节存在。

接下来展现的是一个代码示例，它将配方付诸实践。所有的代码示例都是完整体现的，这避免了模糊性，能让读者清楚地看到发生了什么，而不必亲自去填充附加的细节。有时，书中会包含一个额外的示例，更进一步地演示如何应用配方。

每个配方都以对各种选项和替代方法的讨论结束。这一部分尤其重要，因为它提出了不同的方法，以实现一个解决方案，或者引出思考问题的不同途径。

1.3 一些警告

当阅读本书时，有几个重要的细节应当牢记在心。首先，一个配方展示的是构思解决方法的一种方法，还存在其他的方法(而且经常如此)。对特定的应用程序而言，可能需要不同于此处所展示的方法。可将本书中的配方看成起点，它们能帮助你挑选解决问题的一般方法，并且能激发你的想像力。但是，在所有情况下，对你的应用程序而言，必须判断什么是合适的，什么是不合适的。

此为试读，需要完整PDF请访问：www.ertongbook.com

其次，要着重理解的是，就性能而言，代码示例没有进行优化，而是对清晰性进行了优化，以易于理解。这样做的目的是清楚地演示配方中的步骤。多数情况下，几乎毫不费力就能写出更紧凑、更有效的代码。而且，示例被严格限制成这样：它们易于使用，不必反映为你自己的应用程序编写代码的方法。在所有情况下，你必须创建自己的解决方法，以满足你的应用程序的需求。

第三，每个代码示例都包含错误处理部分，它们对特定的示例而言可能是合适的，但在其他的情况下就不一定恰当了。在所有情况下，你必须正确地处理各种错误和异常，当你改造某个配方以用于自己的代码时，它们可能会出现。下面再陈述一遍这个要点：当实现某个解决方案时，必须为你的应用程序提供合适的错误处理。不能简单地认为对你的应用程序而言，示例中对错误或异常的处理（或没有处理）是足够的或适当的。典型情况下，对真实的应用程序而言，额外的错误处理都是必需的。

1.4 所要求的 Java 经验

本书适合每一位 Java 程序员，不管他是初学者还是有经验的专家。但是，本书假设读者已经懂得了 Java 编程的基础，包括 Java 的关键字、语法及基本的 API 类。读者还应当能创建、编译和运行 Java 程序。本书中对这些事情都不涉及（正如所解释的，本书讲述的是将 Java 应用到各种真实世界的编程问题中，它不涉及 Java 语言的基础）。如果需要提高你的 Java 技巧，我推荐我的图书 *Java: The Complete Reference* 以及 *Java: A Beginner's Guide*，它们都由 McGraw-Hill 公司出版。

1.5 Java 的版本

正如多数读者所知道的，自从 Java 诞生之日起，它就处于持续的变化当中。随着每次新版本的发布，都会增加一些新的特性。多数情况下，一个新的版本也会抛弃（放弃、荒废）一些旧的特性。结果，并不是所有的现代 Java 代码都能在老式的 Java 编译器下编译。这一点非常重要，因为本书中的代码基于 Java SE 6，它是当前的 Java 版本（到写本书时为止）。Java SE 6 中的开发工具集是 JDK 6，它也是本书中用来测试所有代码示例的 JDK。

你可能已经知道，从 JDK 5 开始，Java 中增加了几个重要的新特性，包括泛型、枚举和自动装箱。本书中的某些技术用到了这些特性。如果你使用的 Java 版本是 JDK 5 以前的，就不能编译使用这些新特性的示例。因此，强烈建议使用 Java 的当前版本。

第2章　使用字符串和正则表达式

最常见的编程任务之一要算字符串处理。几乎所有的程序都会以这样或那样的形式与字符串打交道，因为字符串经常是人类与数字信息沟通的桥梁。正是由于字符串处理所扮演的重要角色，因此 Java 对它提供了广泛的支持。

正如所有的 Java 程序员所了解的，主要的字符串类是 String，它提供大量的字符串处理方法。其中的许多方法提供基本的字符串操作，它们是多数 Java 程序员都非常熟悉的。这些方法涉及比较两个字符串、在一个字符串中查找另一个字符串等。不过，String 还包含大量不太出名的方法，它们是对 Java 能力的重要提升，因为这些方法操作的是正则表达式。正则表达式定义的是一般的模式，而不是某个特定的字符序列。然后，这个模式可用于在字符串中查找匹配该模式的子串。这是一个强大的概念，它使 Java 程序员思考字符串处理的方法正在发生变革。

从版本 1.4 开始，Java 就提供了对正则表达式的支持。正则表达式由正则表达式 API 所支持，该 API 位于 java.util.regex 包中。正如前面所说，String 中的某些方法也支持正则表达式。有了正则表达式，一些原来难于应付的字符串处理工作现在变得容易了。

本章包含的配方演示了各种字符串处理技术，它们超越了 String 中基本的查找、比较及替换操作的范畴，其中的几个配方还使用了正则表达式。在某些情况下，也会利用 String 的正则表达式能力，其他一些则使用正则表达式 API 本身。

本章所包含的配方如下所示：

- 以逆序排序字符串数组
- 排序字符串数组时忽略大小写差异
- 查找或替换子串时忽略大小写差异
- 利用 split()将字符串分成几块
- 从字符串中取得键/值对
- 利用正则表达式 API 匹配并抽取子串
- 利用正则表达式 API 标记字符串

2.1　Java 的字符串类概述

字符串是字符的一个序列。与其他编程语言不同，Java 不将字符串作为字符数组对待，而是将它作为对象实现。这使得 Java 对字符串定义了大量的方法。尽管对多数 Java 程序员而言，字符串是他们所熟悉的对象，但回顾一下它的关键属性和功能仍是有用的。

在程序中的多数字符串都是 String 类型的对象，String 是 java.lang 的一部分。因此，对所有的 Java 程序而言，String 是自动可用的。String 的一个最有趣的方面是它创建固定不变的字符串。这意味着一旦创建了 String，就不能改变它的内容。尽管看起来这似乎是个严格的约束，但事实并非如此。如果需要改变字符串，则只需通过包含 String 修饰符来创建一个新的字符串，而原来的字符串保持不变。如果不再需要原来的字符串，可以放弃它。当下一次运行垃圾回收器时，将回收不使用的字符串。通过使用固定不变的字符串，String 比使用可更改的字符串时能更高效地实现。

字符串可用多种方式创建。可以通过使用 String 的一个构造函数来显式地构造字符串。例如，存在从字符数组、字节数组或其他字符串创建 String 实例的构造函数。但是，创建字符串的最简单方法是使用字符串字面值，它是带引号的字符串。所有的字符串字面值都自动成为 String 类型的对象。因此，可以将字符串字面值赋值给 String 引用，如下所示：

```
String str = "Test";
```

这一行创建一个包含单词“Test”的 String 对象，然后给 str 赋值一个该对象的引用。

String 只支持一个运算符：+，它连接两个字符串。例如，

```
String strA = "Hello";
String strB = " There";
String strC = strA + strB;
```

这几行代码的结果是使 strC 包含序列“Hello There”。

String 定义了对字符串进行操作的一些方法。由于多数读者已经对 String 有所了解，因此没有必要对其全部的方法进行详细描述。而且，本章中的配方会全面地描述它们所采用的 String 方法。但是，将 String 的核心字符串处理能力分组成几个类别，重温一下它们是有帮助的。

String 定义了在一个字符串中查找另一个字符串内容的如下方法：

contains	如果一个字符串包含另一个，则返回 true
endsWith	如果字符串以指定的字符串结尾，则返回 true
indexOf	返回一个字符串在另一个字符串中首次出现的索引。如果没有找到，则返回 -1
lastIndexOf	返回一个字符串在指定的字符串中最后一次出现的索引。如果没有找到，则返回 -1
startsWith	如果字符串以指定的字符串开始，则返回 true

如下的方法比较两个字符串：

compareTo	将一个字符串与另一个进行比较
compareToIgnoreCase	将一个字符串与另一个进行比较。忽略大小写差异
contentEquals	将字符串与指定的字符序列进行比较
equals	如果两个字符串包含相同的字符序列，则返回 true
equalsIgnoreCase	如果两个字符串包含相同的字符序列，则返回 true。忽略大小写差异
matches	如果字符串匹配指定的正则表达式，则返回 true
regionMatches	如果一个字符串的指定地域匹配另一个字符串的指定地域，则返回 true

如下一组方法会用一个字符串替换另一个字符串的一部分：

replace	用一个字符或子串替换另一个字符或子串所有出现的位置
replaceFirst	替换匹配指定的正则表达式的第一个字符序列
replaceAll	替换匹配指定的正则表达式的所有字符序列

以下的方法从字符串中抽取子串：

split	基于正则表达式指定的分隔符序列将字符串分成几个子串
substring	返回字符串的指定部分
trim	返回删除了起始和结尾空格的字符串

以下的两个方法改变字符串中字母的大小写：

toLowerCase	将字符串转换成小写
toUpperCase	将字符串转换成大写

除了前面所描述的核心字符串处理方法之外，String 还定义了一些其他的方法，其中常用的两个是 length()和 charAt()，前者返回字符串中字符的个数，后者返回指定索引位置的字符。

多数情况下，本章中的配方都使用 String，而且它通常是处理字符串时的最佳选择。但是，偶尔也存在需要能修改的字符串的情况，Java 提供两个选择。第一个是 StringBuffer，从 Java 发布之日起它就是 Java 的一部分。除了允许改变字符串的内容外，StringBuffer 与 String 相似。因此，它提供了 setCharAt()和 insert()之类的修改字符串的方法。第二个选择是较新的 StringBuilder，它是在 Java 1.5 中新增的。除了不是线程安全的之外，StringBuilder 类似于 StringBuffer。因此，当不使用多线程时，它比 StringBuffer 更高效(在多线程的应用程序中，必须使用 StringBuffer，因为它是线程安全的)。StringBuffer 和 StringBuilder 都位于 java.lang 包中。

2.2 Java 的正则表达式 API

Java 中的正则表达式由 Matcher 和 Pattern 类支持，这两个类都位于 java.util.regex 包中，它们共同起作用。我们使用 Pattern 定义正则表达式，使用 Matcher 匹配其他序列中的模式。具体的过程将在使用它们的配方中描述。

Java API 的其他部分也可以使用正则表达式。也许最重要的是，String 中的各种方法，比如 split()和 matches()，都将正则表达式作为变元(argument)接收。因此，我们经常会使用正则表达式，而不是显式地使用 Pattern 或 Matcher。

本章中的几个配方就使用了正则表达式，其中的多数用的是 String 方法，但有三个使用了 Pattern 和 Matcher。为了详细地控制匹配过程，使用 Pattern 和 Matcher 通常是较容易的。但是，在许多情况下，String 所提供的正则表达式功能就已经足够了，而且更加方便。

当企图使用语法错误的正则表达式时，有些使用正则表达式的方法会抛出 PatternSyntaxException 异常。这个异常由正则表达式 API 定义，它位于 java.util.regex 包中。必须用适合于你的应用程序的方法处理这个异常。

2.3 正则表达式介绍

在使用正则表达式之前，必须首先理解它们是如何构造的。如果不熟悉正则表达式，则这个概述能帮助你入门。在继续讲解之前，需着重强调的是，正则表达式的主题是非常庞大的。实际上，有几本完整的书所讲的就是正则表达式。详细描述正则表达式完全超出了本书的范围，这里给出的简单介绍为理解配方中的示例提供了足够的信息，它也能让你亲自体验正则表达式。但是，如果要大量使用正则表达式，则需要更加细致地研究它们。

作为这里使用的一个术语，“正则表达式”(regular expression)是一串描述一个模式的字符，模式将匹配任何适合它的字符序列。因此，模式就构成了匹配大量特定序列的通用形式。结合正则表达式引擎(比如由 Java 的正则表达式 API 提供的那些)，模式可用来查找在另一个字符序列中的匹配情况。当操作字符串时，这就是正则表达式提供的能力。

正则表达式由如下的一个或多个部分组成：常规字符、字符类(字符集合)、通配符、量词、边界匹配符、运算符和组，下面将逐一简要介绍。

> **说明** 当正则表达式被不同的正则表达式引擎处理时，在处理方式上存在某些不同。这里的讨论描述的是 Java 的实现。

2.3.1　常规字符

常规字符(即字符字面值)按其字面含义匹配。因此,如果模式由 xy 组成,则匹配该模式的唯一输入序列就是“xy”。类似换行符和制表符这样的字符可使用标准的转义序列来指定,转义序列以“\”开头。例如,新行符由 \n 指定。

2.3.2　字符类

字符类是字符的集合。字符类由位于一对方括号中的字符指定,一个类将匹配该类中的任何字符部分。例如,类[wxyz]匹配 w,x,y 或 z。要确定一个排除性集合,可在字符前加符号“^”。例如,[^wxyz]匹配 w,x,y 或 z 之外的任何字符。要指定一定范围的字符,可使用连字符。例如,要指定一个匹配数字 1~9 的字符类,可使用[1-9]。只需简单地指定两个或多个范围,类就能包含它们。例如,类[0-9A-Z]匹配所有的数字及从 A 到 Z 的大写字母。

Java 正则表达式 API 提供了一些预定义的类,以下是常用的一些。

预定义的类	匹　　配
\d	0~9 的数字
\D	所有的非数字字符
\s	空白符
\S	所有的非空白符
\w	可以成为单词的一部分的字符。在 Java 中,它们是大写和小写字母、数字 0~9 及下划线,它们通常被称为单词字符
\W	所有的非单词字符

除了这些类之外,Java 还提供其他大量的字符类,它们都具有如下的通用形式:

\p{*name*}

其中,*name* 指定类的名称。以下是几个示例:

\p{Lower}　　包含小写字母
\p{Upper}　　包含大写字母
\p{Punct}　　包含所有的标点符号

还有其他更多的类。有关你的 JDK 所支持的字符类,应当参考 API 文档。

一个类可以包含其他的类。例如,[[abc][012]]定义的类将匹配字符 a,b 或 c,或者数字 0,1 或 2。因此,它是两个集合的联合体。当然,这个示例可以更简便地写成[abc012]。但是,在另外的上下文中,嵌套的类是非常有用的,比如当使用预定义的集合或者想创建两个集合的交集时。

要创建包含两个或多个字符集合的交集的类,可以使用 && 运算符。例如,如下的形式创建一个集合,匹配大写字母以外的所有单词字符:[\w &&[^A-Z]]。

有另外两点需指出:在字符类之外,“-”被当做一个常规字符。而且,在类之外,“^”用来指定一行的开始,下面很快将会看到关于它的描述。

2.3.3　通配符

通配符是“.”(点号),它可以匹配任何字符。因此,由“.”组成的模式将匹配以下这些(以及其他)输入序列:“A”,“a”,“x”及“!”。本质上,点号就是一个预定义的类,它匹配所有字符。

要创建匹配一个点号的模式,可在点号之前加一个“\”。例如,给定输入字符串:

Game over.

表达式

```
over\.
```

将匹配序列“over.”。

2.3.4 量词

量词决定了表达式将匹配多少次。量词如下所示：

+	匹配 1 次或多次
*	匹配 0 次或多次
?	匹配 0 次或 1 次

例如，“x+”将匹配 1 个或多个 x，比如“x”，“xx”，“xxx”，等等。模式“.*”将匹配任何字符 0 次或多次。模式“,?”将匹配 0 个或 1 个逗号。

还可以指定匹配模式指定次数的量词。以下是它的一般形式：

{num}

因此，x{2}将匹配“xx”，但不匹配“x”或“xxx”。利用以下的量词，可以指定匹配模式的最小次数：

{min,}

例如，x[2,]匹配“xx”，“xxx”，“xxxx”，等等。

利用以下的量词，可以指定匹配模式的最小次数和最大次数：

{min,max}

2.3.5 贪婪量词、胁迫量词和占有量词

实际上，量词有三种变体：贪婪量词、胁迫量词和占有量词。前面所示的示例都是贪婪量词的示例，它们匹配最长的匹配序列。胁迫量词（也称为懒惰量词）匹配最短的匹配序列。要创建一个胁迫量词，可在其后面加一个“?”。占有量词匹配最长的序列，不匹配较短的序列，即使它能使整个表达式匹配成功。要创建一个占有量词，可在其后面加一个“+”。

下面给出几个量词类型的示例，它们试图从字符串“simple sample”中查找匹配。模式“s.+e”将匹配最长的序列，即整个字符串“simple sample”，因为贪婪量词“.+”将匹配从第一个 s 开始，直到最后的 e 的所有字符。

模式“s.+?e”将匹配“simple”，它是最短的匹配。这是因为，胁迫量词“.+?”在找到第一个匹配序列之后将停止。

模式“s.++e”会失败，因为占有量词“.++”将匹配第一个 s 之后的所有字符。由于模式是占有的，它不会释放最后的 e，以使整个的模式匹配。因此，最后的 e 将不会找到，匹配失败。

2.3.6 边界通配符

有时，我们希望指定模式以某个边界开始或结束，比如指定单词以什么结尾，或者一行以什么开始。为此，可以使用边界通配符。也许用得最多的边界通配符是^和$，它们匹配所搜索的行的开始和结尾，默认情况下就是输入字符串的开始和结尾。例如，给定字符串“test1 test2”，模式“test.?$”将匹配“test2”，而模式“^test.?”匹配“test1”。如果想匹配这些字符本身，则需要使用转义序列：\^或 \$。

另外的边界通配符如下所示。

此为试读，需要完整PDF请访问：www.ertongbook.com