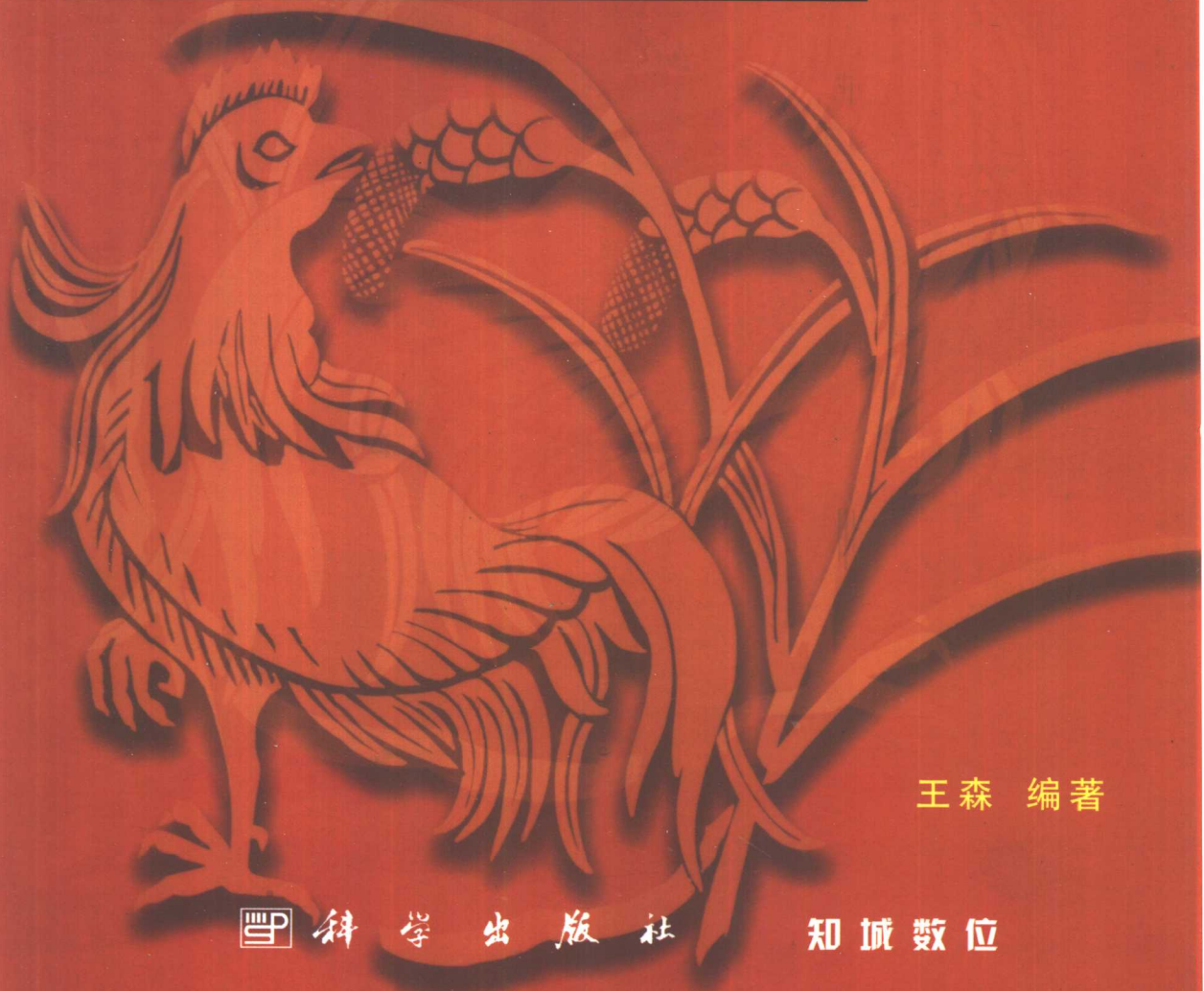
深入浅出

KJava

Java在PDA上的程序设计



王森 编著

科学出版社

知城数位

KJava 深入浅出

——Java 在 PDA 上的程序设计

王　森 编著

科学出版社

2001

内 容 简 介

随着 PDA 和手机的不断普及，编写手持移动装置上的应用程序，包括 PDA、手机上的应用程序，已经越来越广泛。

本书主要介绍了如何用 Java 在 PalmOS 上编写应用程序，共分为两个部分，第一部分是 J2ME 的简介，第二部分是 PalmOS 上的 Java 程序设计。本书内容深入浅出，语言简洁流畅。

本书适合于进行手持移动装置程序开发的各级用户。

图字： 01-2001-2185 号

图书在版编目（CIP）数据

KJava 深入浅出——Java 在 PDA 上的程序设计/王森编著.-北京：科学出版社，2001

ISBN 7-03-009442-5

I.K… II.王… III.移动通信—携带电话机—Java 语言—程序设计

IV. TN929.53

中国版本图书馆 CIP 数据核字(2001) 第 038531 号

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

北京双青印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2001 年 7 月第 一 版 开本：720×1000 1/16

2001 年 7 月第一次印刷 印张：19 1/2

印数：1—5 000 字数：318 000

定价：29.00 元

(如有印装质量问题,我社负责调换<环伟>)

序

“Java 在手持移动装置上非常有意思！”，我一直用这句话鼓励大家使用 Java 来编写手持移动装置上的应用程序，包括 PDA 上的应用程序、手机上的应用程序以及未来无限可能的应用。毕竟，这个世纪是移动计算(mobile computing)的世纪，PDA 和手机的普及率将远远超过 PC 机目前的普及率，其所能展现的应用前景将无可限量。

在这么多复杂的平台中，要程序设计师为每个平台编写应用程序是一件不可能的事情。于是，Java 在设计时的最初理念——“Write once， run anywhere”，使得 Java 成了移动通讯世纪中最佳的程序语言。Java 比起 C 或 C++都要容易学，正在看这本书的朋友可能没接触过 Java，可是我相信只要随手找一本 Java 相关的书籍，一定可以很快上手。更何况，在本书中我们所用到的属于 Java 的特殊语法实在不多。

本书主要介绍了如何用 Java 在 PalmOS 上编写应用程序，本书分为两个部分，第一部分是 J2ME 的简介；第二部分是 PalmOS 上的 Java 程序设计，其中包括编写第一个 PDA 程序，支持 J2ME 的开发工具，深入 Spotlet，KJava 数据库程序设计，KJava 图形处理，KJava 图形用户界面程序设计，KJava 对外沟通的桥梁，KJava 游戏设计等。希望这本书能带您顺利进入 PalmOS 程序设计的世界。

虽然我尽力将事情做到最好，可是百密一疏，希望大家随时来信给予指教。

E-mail：moli@pchome.com.tw 或 moli.mt88g@nctu.edu.tw

作 者

目　录

1

Java 2 Micro Edition 简介

我们所处的世界，比我们想象中的要怪异，
而且，比我们所能想象的还要怪异。

1.1 前 言

如果您曾经到 http：//www.javasoft.com 网站上查询有关 Java 2 Micro Edition 的资料，十之八九会被一大堆的技术名词搞得一头雾水。什么 KVM，什么 CLDC，CDC，MIDP，后面还冒出了 Personal Java，Embedded Java 以及 JES 等名词。虽然名为 Java 的微小版本，可是它的世界可真是不小，让我们有满脑子“见山不是山，见水不是水”的疑惑。

的确，在我刚开始接触 Java 2 Micro Edition 的时候，就感觉到这个玩意儿实在越看越让人摸不着头绪。因此在本章中，我舍弃了技术上的细节，希望带大家从宏观角度来看待 Java 2 Micro Edition 的世界。希望读过本章之后，可以使大家体验“见山是山，见水是水”，一切豁然开朗的感觉。

首先，我们必须先对 Java 2 Micro Edition 在整个 Java 技术之中的定位做个了解。

1.2 各种 Java 版本的定位

使用 Java 编写过程序的人都知道，Java 的函数库与一般大家所知道的函数库有些不同。一般我们在编写 C 程序时，会使用 C 标准函数库；使用 C++ 编写程序时，会使用 C++标准函数库；而使用 Java 编写程序时，则使用类函数库。之所以叫做类函数库，是因为所有的函数被分门别类地归类在不同的类别之下，比较起来，传统标准函数库的组成结构就显得有点松散。

Java 规范之中有一组所谓的核心类函数库（Core Class，即 java.* ），在核心类别之外还有所谓的扩展类函数库（Extended Class，即 javax.* ）。根据对这两种类别所支持的程度，Sun Microsystems 进而区分出四种不同的 Java 版本，如图 1-1 所示。

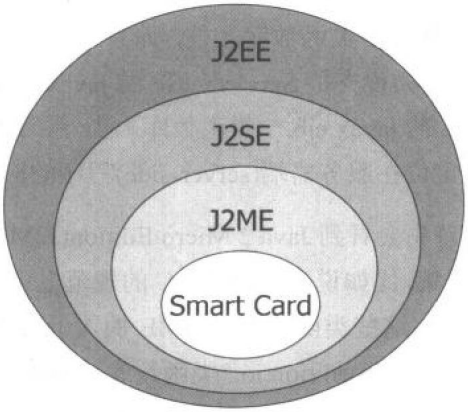


图 1-1

图 1-1 只是针对各种不同 Java 版本所支持的核心类函数库的范围来进行描述，并无法针对这些版本所支持的扩展类函数库做说明。另外，图 1-1 也说明了这些版本所支持的 Java 基本类别（Primitive type，即 boolean，byte，short，int，long，float，double 这些类别）的范围。越在同心圆的外面，所支持的核心类函数库就越完整。反之，位于同心圆内部的 Java 版本，所支持的就只是核心类函数库的子集，而且越往里面，所支持的核心类函数库子集就越小。同理，越在同心圆的外部，所支持的 Java 基本类别就越完整，而越往同心圆内部，所支持的 Java 基本类别就越少。

注意：请大家注意，在这里我们说“所支持的核心类别函数库是标准核心类别函数库的子集”，充其量也只是名称上为子集而已。这些类别函数库子集的内容却不一定与标准的类别函数库相同，这是因为即使是相同名称的类别函数库，为了对不同的平台予以最佳化(比如执行速度和内存使用量)，所以 Sun Microsystems 对类别函数库子集的内容做了翻修，因而导致类别函数库虽然在名称上相同，但是实质上却不一样。

要理解图 1-1，我们必须以 Java 2 Standard Edition(J2SE)作为基准，这个版本应用了所有 Java 标准规范之中所定义的核心类函数库，也支持所有的 Java 基本类别。J2SE 定位在客户端(client-side)程序的应用上。

从 J2SE 往外延伸，其外面的同心圆为 Java 2 Enterprise Edition(J2EE)，此版本除了支持所有的标准核心类函数库之外，而且还增加了许多支持企业内部使用的扩展类函数库。比如说支持 Servlet / JSP 的 javax.servlet.* 类函数库、支持 Enterprise Java Bean 的 javax.ejb.* 类函数库。当然，J2EE 必定支持所有的 Java 基本类别。J2EE 定位在服务器端(server-side)程序的应用上。

从 J2SE 向内看，首先会看到 Java 2 Micro Edition(J2ME)，它所支持的只有标准核心类函数库的子集，比如说 J2ME CLDC 的规范之中，只支持 java.lang.*，java.io.*，以及 java.util.* 这些类函数库。J2ME 加入了一些支持嵌入式系统的扩展类函数库，如 javax.microedition.io.*类函数库。然而，此版本并不支持所有的 Java 基本类别，就标准 J2ME CLDC 的实际操作，也就是能在 PalmOS 上执行的 KVM(K Virtual Machine)来说，它就不支持属于浮点数(float，double)的 Java 基本类别。J2ME 定位在嵌入式系统的应用上。

同心圆的最里面，还有一个 Java 的 Smart Card 版本，这个部分原本在 Java 的官方文件中并没有这样定义，但是以我对所有 Java 版本的了解，将它画在 J2ME 内部是非常合理的。因为 Smart Card 版本只支持 java.lang.*这个核心类函数库，而且比起 J2ME 所支持的核心类函数库更少，但是它也有属于自己的扩展类函数库，如 javacard.*，javacardx.*这些类函数库。Smart Card 版本只支持 boolean 与 byte 这两种 Java 基本类别。就如同其名称，此版本定位在 Smart Card 的应用上。

在以上每一种 Java 版本中，都有属于它们自己的虚拟机(VM)，从而达成“Write once，run anywhere”的最终目标。在 Smart Card 上有 Card VM，负责执行下载到 Smart Card 上的 Card Applet。在 J2ME 的世界里，其标准参考应用——KVM，用来执行下载至嵌入式装置上的 Spotlet 或 MIDlet；在 J2SE 与 J2EE 之中，有 Classic VM 与 HotSpot VM 负责执行 Java Applet，Java Servlet 或 Java Application。HotSpot VM 是新一代的 Java 虚拟机，相比之下 Classic VM 则是传统的 Java 虚拟机。根据 Sun Microsystems 的说法，HotSpot VM 执行 Java 程序的效率比起 Classic VM 有所提高。如果您的计算机上装有 Java 2 SDK，您可以在下面两个目录下分别找到 Jvm.dll。

```
JDK 安装目录\jre\bin\classic
JDK 安装目录\jre\bin\hotspot
```

这两个动态连接函数库就是我们所说的 Classic VM 与 HotSpot VM。至于详细的细节在此不多介绍，如果大家对 Hotspot VM 有兴趣，可以参考：

```
Java Platform Performance Strategies and Tactics
Addison Wesley 2000
```

其实 HotSpot VM 还分成 client 版与 server 版。为何要区分成这两种呢？主要是因为在客户端执行的应用程序与服务器上所执行的应用程序在基本需求上有所不同：客户端应用程序较注重画面上的美观、用户界面的灵敏度；而服务器上执行的应用程序则基本上没有画面上的需求，但是较注重对客户端请求的响应，因此 HotSpot VM 的两种版本就是为了针对这两种不同的需求分别予以最佳化而产生的结果，从而所有 Java 版本与其所依据的虚拟机所堆积起来的结构可以用图 1-2 表示。

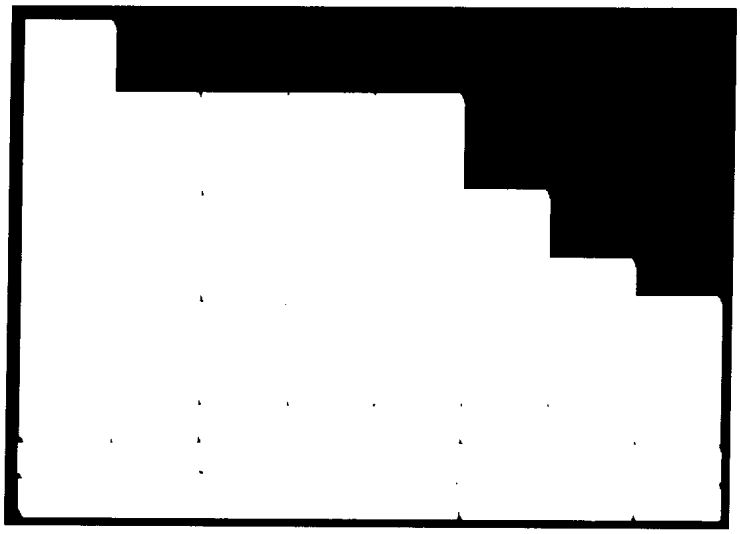


图 1-2

以上所叙述的虚拟机都只是 Sun Microsystems 根据 Java 虚拟机规范的标准应用，实际上还有许多家公司都有自己所开发出来的虚拟机，而且也符合 Java 虚拟机的规范，比如说 IBM 的 J9 VM 就是 KVM 之外另一个支持 J2ME 的虚拟机。所以大家千万别以为 Java 虚拟机只 Sun Microsystems 一家，如果您对其他公司支持 J2ME 的虚拟机有兴趣，请参考表 1-1 所列的网站。

表 1-1

名 称	URL
Esmertec	http：//www.esmertec.com/
J9 VM	http：//www.embedded.oti.com/palm/

1.3 JINI 技术

既然存在那么多不同的 Java 版本，那么，如果这些针对不同 Java 版本所开发的 Java 程序能够彼此连结在一块，相互联机，彼此分享各自的资源，岂不是一件很棒的事情吗？这就是 JINI 之所以被创造出来的原因。各种不同的 Java 版本可以通过 JINI 彼此联系，如图 1-3 所示。

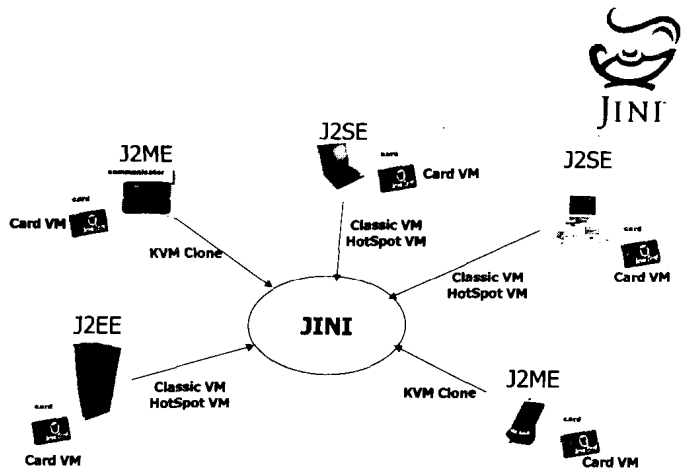


图 1-3

有关 JINI 的技术，在此笔者不多着墨，因为市面上已经有许多书籍讨论此技术。不过在上图中要请大家注意的是，Smart Card 版本由于其特性的原因，所以并没有通过 JINI 与其他 Java 版本的应用程序做沟通，而只是在 JINI 所造成的广大分布式环境中作为认证之用。

1.4　各种不同版本的 Java 程序的开发

如前面所说，各种不同的 Java 版本，在其支持的核心类函数库的完整性以及所支持的 Java 基本类别这两件事情上都有所差异，但是对于程序设计师而言，这些版本的关系如图 1-4 所示。

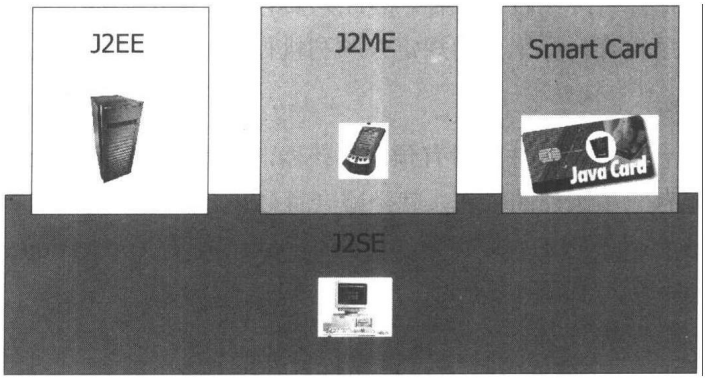


图 1-4

也就是说，不管您开发的是企业所使用的 Java 程序、嵌入式装置上执行的 Java 程序、浏览器上执行的 Applet，或是在 PC 上执行的应用程序，您都必须在您的计算机上先安装 J2SE，然后再安装各种版本的核心类函数库以及额外的扩展类函数库，这样才能成功地开发各种不同目的的 Java 程序。

J2SE 所提供的 Java 编译器(javac.exe)可以帮助我们编译各种不同平台上的 Java 程序，而 J2SE 所提供的 Java 虚拟机(java.exe)则可以帮助我们在 PC 上先行测试这些程序执行结果的正确与否。

另外，Java 编译器并不会帮用户检查程序是否符合各种平台上所支持的核心类函数库与 Java 基本类别。例如，虽然我们在前面说过，Smart Card 版本并不支持 boolean，byte 以外的 Java 基本类别，而且该平台也只支持 java.lang.* 核心类别，可是当我们在编写 Smart Card 平台上的程序时，就算在程序代码里用了 boolean 或 byte 以外的 Java 基本类别，或者使用了 java.lang.*之外的其他核心类别，编译器仍然可以照常帮您编译出类别文件。这时，大家一定开始产

生疑惑——那么这些程序如果放到 Smart Card 上面执行时，出了问题怎么办？难道不会造成 Smart Card 上的虚拟机执行时发生错误吗?针对这个可能发生的潜在问题，Sun Microsystems 在各种不同版本的开发套件中，有些会内附检查器(checker)或者预先审核器(preverifier)，这两个工具可以帮助用户在将程序放到目标平台之前先做好检查和预先审核的工作。

检查器会帮用户找出类别文件中不合目标平台规范的部分，并提醒用户这些地方可能无法在目标平台上执行。因此只要有检查器的协助，大致上就可以确定用户的程序符合目标平台的规定并顺利执行。Java Card 的开发套件中就附有检查器。

而某些平台的开发套件则附有预先审核器，预先审核器除了做检查器做的工作之外，还有一项额外的工作，就是减轻目标平台上虚拟机的负担，要解释预先审核器这个额外的工作，首先看看传统 Java 程序(Application，Servlet，Applet)的执行程序，如图 1-5 所示。

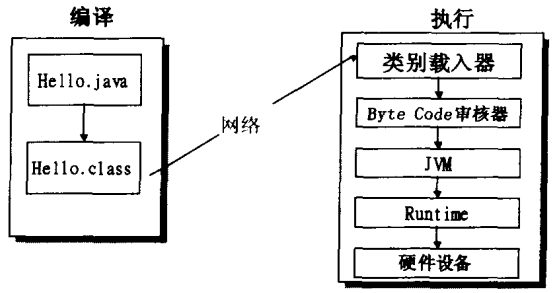


图 1-5

在传统的 Java 程序之中，为了安全上的考虑，任何进入执行环境的类别文件 (不管该类别文件是来自本机或是远程机器)，都必须先经过 Byte Code 审核器(Byte code verifier)的验证，以防止程序在传送途中遭到恶意的修改，而使得 Java 程序在执行时对系统有不良影响。经过审核之后，该类别文件才能开始被虚拟机执行。

如果这个审核的动作是在一般的 PC 机上执行，速度倒是还能够接受，可是一旦放到如 Palm 或是手机这些 CPU 较慢、内存也比较少的机器上面就显得

十分吃力了。为了节省 CPU 的运算时间(既能省电又能够加速程序执行)，因此，在程序设计师产生能够让某些特定平台执行的类别文件之前，程序设计师必须先在 PC 机上使用预先审核器做一些前置的审核工作，预先审核器会在类别文件中加入一些特殊标记或符号。这样一来，当这些程序放到目标平台上执行时，就可以大度幅减少在目标平台上做审核时的时间，从而加速程序的激活及执行速度，因此在 J2ME 之下的程序(Spotlet，MIDlet)，其执行步骤变成图 1-6 所示的样子。

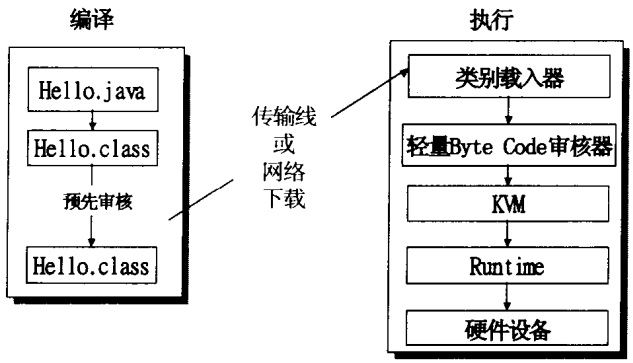


图 1-6

从图 1-6 中可以看到，因为预先审核的关系，执行时 Byte Code 审核器的工作就变少了，也因此从程序加载到开始执行之间的时间也缩短了。CLDC 标准应用和 MIDP 参考应用之中就附有预先审核器。

1.5　Java 版本的升级

相信熟悉 Java 升级历史的人或多或少都听说过，Java 技术并非一开始就叫做 Java，而是叫做 OAK，而且最早的时候就是为了嵌入式系统而设计的一项产品。后来因为互联网的发展，而 OAK 的诸多特性刚好又适合用在网络上(例如可移植性、编译后程序代码很小)，因为商标已被注册的关系，因此 OAK 被改名成 Java，从此成了网络上的闪亮巨星，并随之越来越成熟，也慢慢地产生了许多非原本预期中的相关运用。虽然 Java 已经被用到许多企业级软件上，其实还是非常适合用在嵌入式系统之中。