J2ME 技术

第一章：J2ME 概述

- **介绍**

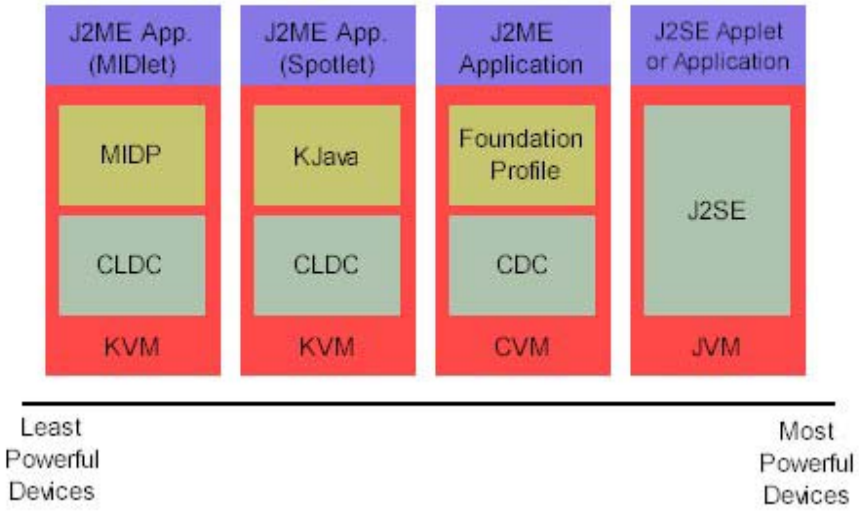
  这一章将开始教您使用 J2ME。我们将从定义 J2ME 开始，然后讨论它的总体架构并学习 J2ME 目标设备。作为架构讨论的一部分，我们将提供有关简表和配置的概述（后面的章节中将详细介绍简表和配置）。同时我们会简要介绍打包和配置 J2ME 应用程序过程中的一些注意事项。

- **J2ME 是什么？**

  Sun Microsystems 将 J2ME 定义为“一种以广泛的消费性产品为目标的的高度优化的 Java 运行时环境，包括寻呼机、移动电话、可视电话、数字机顶盒和汽车导航系统。”
  自从 1999 年 6loper Conference 上声明之后，J2ME 为小型设备带来了 Java 语言的跨平台功能，允许移动无线设备共享应用程序。有了 J2ME，Sun 已经使 Java 平台能够适应集成了或基于小型计算设备月在 JavaOne Deve 的用户产品。

- **J2ME 总体架构**

  J2ME 使用配置和简表定制 Java 运行时环境 (JRE)。作为一个完整的 JRE，J2ME 由配置和简表组成，配置决定了使用的 JVM，而简表通过添加特定于域的类来定义应用程序。
  配置将基本运行时环境定义为一组核心类和一个运行在特定类型设备上的特定 JVM。我们将在 J2ME 配置一章中详细讨论配置。
  简表定义应用程序；特别地，它向 J2ME 配置中添加特定于域的类，定义设备的某种作用。我们将在 J2ME 简表一章中深入介绍简表。
  下面的图表描述了不同的虚拟机、配置和简表之间的关系。它同时把 J2SE API 和它的 Java 虚拟机进行了比较。虽然 J2SE 虚拟机通常被称为一种 JVM，但是 J2ME 虚拟机、KVM 和 CVM 都是 JVM 的子集。KVM 和 CVM 均可被看作是一种 Java 虚拟机 -- 它们是 J2SE JVM 的压缩版，并特定于 J2ME。



- **配置概述**

  配置将基本运行时环境定义为一组核心类和一个运行在特定类型设备上的特定 JVM。虽然还

可能在将来定义其他的配置，但当前 J2ME 存在两种配置：

· 连接限制设备配置 (CLDC) 特别与 KVM 一起用于内存有限的 16 位或 32 位设备。这是用于开发小型 J2ME 应用程序的配置（虚拟机）。（从开发的角度来看）它的大小限制让它比 CDC 更有趣、更具挑战性。CLDC 同时还是用于开发绘图工具应用程序的配置。Palm 电脑便是一个运行小应用程序的小型无线设备的示例。我们将在 J2ME 配置一章中深入介绍 CLDC。

· 连接设备配置 (CDC) 与 C 虚拟机 (CVM) 一起使用，用于要求内存超过 2 兆的 32 位体系结构。互联网电视机顶盒便是这类设备的一个示例。虽然稍后我们将在 CDC API 一章中简要介绍 CDC，但它并不在本教程的范围内。

- **简表概述**

简表定义了您的应用程序所支持的设备类型。特别地，它向 J2ME 配置添加了特定于域的类来定义设备的某种作用。简表建立在配置的顶部。已经为 J2ME 定义了两种简表：KJava 和移动信息设备简表 (MIDP)，它们也被建立在 CDLC 上。这两种简表适用于小型设备。

有一种纲要简表，您可以在它的上面创建自己的简表，这种纲要简表也称为基础表，可供 CDC 使用。然而，在本教程中，我们只重点介绍建立在 CLDC 顶部，适用于小型设备的简表。

我们将在后面的章节中讨论上述这些简表，还会使用 KJava 和 MIDP 建立一些示例应用程序。

- **J2ME 目标设备**

使用 CLDC 开发的 J2ME 应用程序的目标设备通常具有以下特征：

· 可供 Java 平台使用的 160 到 512 千字节的总内存

· 功率有限，常常是电池供电

· 网络连通性，常常是无线的、不一致的连接并且带宽有限

· 用户接口混乱，程度参差不齐；有时根本就没有接口

一些 CLDC 支持的设备，包括无线电话、寻呼机、主流个人数字助手 (PDA)，以及小型零售支付终端。

依照 Sun Microsystems，CDC 的目标设备通常具有以下特征：

· 使用 32 位处理器

· 2 兆字节或更多可供 Java 平台使用的总内存

· 设备要求的 Java 2 “蓝皮书”虚拟机的全部功能

· 网络连通性，常常是无线的、不一致的连接并且带宽有限

· 用户接口混乱，程度参差不齐；有时根本就没有接口

一些 CDC 支持的设备，包括常驻网关、智能电话和通讯器、PDA、管理器、家用电器、销售网点终端以及汽车导航系统。

- **J2ME、J2SE 与 J2EE 之间的比较**

下面的图表描述了支持 J2ME 应用程序的设备，同时说明了 J2ME 适合 Java 平台之处：

PAGE TO TOP

J2ME 技术

第二章：开发 J2ME 应用程序

- **介绍**
  在这一章中，我们将复习一下在为小型设备开发应用程序时需要牢记的一些注意事项。我们将看一下在使用 J2SE 编译 J2ME 应用程序时调用编译器的方法。最后我们将探究打包和部署，以及在这个过程中提前验证所扮演的角色。

- **设计开发小型设备应用程序要注意的事项**
  为小型设备开发应用程序，需要您在设计阶段制定某种策略。最好是在开始编写代码之前，战略性地为小型设备设计应用程序。由于无法考虑到所有的 "gotchas" ，在开发应用程序之前更正代码是一件很痛苦的工作。
  下面是一些可以考虑的设计策略：
  · 保持程序简单。除去不必要的功能，如果可能的话，将它们做成独立的、次要的应用程序。

  · 程序越小越好。这一点对所有的开发者来说应该是显而易见的。越小的程序占用的设备内存越少，并且花费的安装时间越少。可考虑将您的 Java 应用程序打包，作为压缩的 Java 档案 (jar) 文件。

  · 运行时占用最少的内存。为尽可能减少运行时占用的内存，使用标量类型代替对象类型。同时，不依赖垃圾收集程序。您应该在使用完对象时将对象引用置空，这样可有效管理内存。另外一种减少运行时所需内存的方法是使用“惰性”实例，它仅在必需时才分配对象。其它一些减

少小型设备上过量和峰值内存使用的方法有快速释放资源、重新使用对象以及避免异常。

- **设计开发移动设备应用程序的注意事项**

  开发移动设备应用程序的规则与我们前面提及的开发小型设备的相同：先设计再编码。让我们检查一下开发移动设备应用程序时可考虑的一些设计建议：

  · 让服务器做大部分的工作。将计算性较强的任务放到服务器上，让服务器为您做这些工作。让移动设备处理界面和最少的计算工作，而让服务器做繁重的工作。当然，您为其开发应用程序的移动设备对设备连接到服务器上的难易程度和频率有重要影响。

  · 谨慎地选择编程语言。J2ME 仍然处于成长期，可能还不是最好的选择。根据您的需要，选择其它的面向对象语言，如 C++，可能会更好。

- **性能注意事项**

  为性能而编码。下面有一些以获得最优性能为目标的编码方法：

  · 使用局部变量。访问局部变量比访问类成员更快。

  · 避免字符串串联。字符串串联不仅会降低性能，而且会增加应用程序的内存峰值占用量。

  · 使用线程，避免同步。任何运行时间超过 1/10 秒的操作都需要一个独立的线程。避免同步同样能提高性能。

  · 使用模型视图控制器 (MVC) 分离模型。MVC 将代码中控制显示的逻辑分离出来。

- **编译注意事项**

  同其它的 Java 应用程序一样，您在打包和部署应用程序之前要先进行编译。尽管有了 J2ME，您仍然使用 J2SE 编译器并且需要用适当的选项来进行调用。

  特别的，您需要使用 -bootclasspath 选项来指示编译器使用 J2ME 类，而不是 J2SE 类。不要在编译器的 CLASSPATH 中设置配置类。这样将导致运行时错误，因为不管 CLASSPATH 中有什么，编译器将首先自动搜索 J2SE 的核心类。换句话说，编译器将无法引用特定 J2ME 配置中缺少的类或方法，结果导致在尝试运行应用程序时出现运行时错误。

- **打包和部署注意事项**

  由于 J2ME 是为内存有限的小型设备设计的。大部分常用的 Java 提前验证已经从虚拟机中除去以形成一个较小的覆盖区域。结果，在配置之前提前验证 J2ME 应用程序就很有必要。在运行时会附加一个检查以确保这个类在提前验证之后还没有改变过。

  如何严格执行提前验证或者检查类的正确性依靠的是工具包。CLDC 提供一个称为提前验证的命令行实用程序，它能够进行实际的验证并且可以把一些额外的信息插入到类文件中去。MIDP 使用无线工具包，这种工具包提供一种 GUI 工具，也可从命令行运行这种工具。

  部署工作取决于您要部署的平台。应用程序必须以一种适合 J2ME 设备类型的格式进行打包和配置，就如简表定义的那样。

J2ME 技术

第三章：J2ME 配置

- **J2ME 配置是什么？**
  正如前面所学，配置将基本运行时环境定义为一套核心类和一个运行在特定类型设备上的特定的 JVM。您也可以学到 J2ME 的两种配置类型是 CLDC 和 CDC。
  Sun 提供的 J2ME 配置是适合不同层次的市场需求的 -- CLDC 适合小型设备，而 CDC 适合大型设备。J2ME 环境可以被动态地配置为提供运行应用程序所需要的环境，而不用考虑是否为设备提供了运行该应用程序所需的所有 Java 技术库。核心平台接收应用程序代码和库。运行在网络上的服务器软件执行配置工作。
  在下面的几页中，您将学到关于 CLDC 和 CDC 的更多知识以及与它们关联的简表。

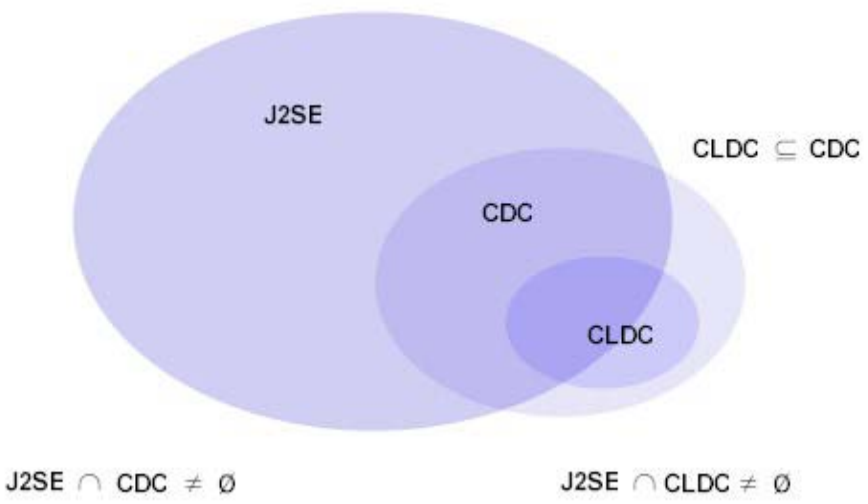
- **连接限制设备配置 (CLDC)**
  CLDC 是由 Java Community Process 创建的。正如 Sun Microsystems 的 Web 站点所定义的那样，它的标准是："轻便、覆盖区域最小的 Java 构建块，适合小型的、有资源限制的设备。"
  J2ME CLDC 配置是为将在业界定义的简表中使用的一个虚拟机和一套核心库准备的。正如第 2 章中提及的，一个简表通过在基本 J2ME 配置顶部提供特定于域的类来为特定的设备定义应用程序。K 虚拟机 (KVM)、虚拟机的 CLDC 的参考执行和它的 KJava 简表运行在 CLDC 的顶部。
  CLDC 简要描述了高度限制设备上每个 J2ME 执行所要求的一套最基本的库和 Java 虚拟机特征。CLDC 主要面向那些网络连接速度慢、能源有限（经常是电池供电）、具有大于等于 128 KB 的稳定内存、以及大于等于 32 KB 的不稳定内存的设备。不稳定内存是不持久的并且没有写保护，这意味着如果关掉设备，内存中的内容将全部丢失。而稳定内存中的内容是持久的，并且有写保护。CLDC 设备使用稳定内存来存储运行时的库和 KVM，或存储为某个特殊设备创建的另一个虚拟机。不稳定内存被用来分配运行时的内存。

- **CLDC 要求**
  CLDC 定义了下列要求：
  · 完整的 Java 语言支持（除浮点支持、最终定案和错误处理之外）
  · 完整的 JVM 支持
  · CLDC 的安全性
  · 有限国际化的支持
  · 继承类 -- 所有不针对 CLDC 的类都必须是 J2SE 1.3 类的子类
  · 针对 CLDC 的类都在名为 javax.microedition 的软件包和它的子包里
  除 javax.microedition 软件包以外，CLDC API 还由 J2SE 的子集 java.io，java.lang 以及

java.util 等软件包组成。我们将在 CLDC API 这一章中学习有关的细节问题，然后使用 CLDC API 来开发我们的绘图应用程序。

- **连接设备配置 (CDC)**
  连接设备结构 (CDC) 被定义为一种添加了 CLDC 类的 Java 2 标准版 (J2SE) 的简化版。因此，CDC 是建立在 CLDC 的基础之上，并且为 CLDC 设备开发的应用程序也可以运行在 CDC 设备上。
  CDC 也是由 Java Community Process 开发的，它为用户的电子设备和嵌入式设备如智能电话、双向寻呼机、PDA、家用电器、销售网络终端以及汽车导航系统等提供一种标准化的、轻便的、功能齐全的 Java 2 虚拟机构建块。这些设备运行 32 位的微处理器和超过 2 MB 的内存，这些对于存储 C 虚拟机和库是必需的。K 虚拟机支持 CLDC，而 C 虚拟机 (CVM) 支持 CDC。CDC 与基础表相关联，这一点不在本教程的范围之内。
  我们将在 CDC API 这一章中更详细地学习 CDC。

- **CLDC 与 CDC 的比较**
  面这幅图描述了 CDC 和 CLDC 之间的关系。同时该图也揭示了它们与整个 J2SE API 系。正如前面所说，CDC 是加上一些额外类的 J2SE 的子集。我们也可以看到 CLDC 是 CDC 的子集。



PAGE TO TOP

- **J2ME 简表是什么?**
  如我们在前面教程中提及的，一个简表定义了受支持设备的类型。例如，移动信息设备简表(MIDP)，定义了蜂窝电话的类。它把一些特定于域的类加入 J2ME 配置中来定义对类似设备的使用。已经为 J2ME 定义了两个简表：KJava 和 MIDP 。它们都构建在 CLDC 之上。KJava 和 MIDP 都和 CLDC 及小型设备相关联。
  简表被构建在配置的顶部。由于简表是特定于运行应用程序的设备的大小（内存的数量），所以，某个简表是与某种特定的配置相关联的。
  在纲要简表上，您可以创建自己的简表，这种纲要简表也称为基础表，它对于 CDC 也是可用的。然而，在本教程和本节中，我们将只重点学习建立在 CLDC 上的 KJava 和 MIDP 简表。

- **简表 1：KJava**
  KJava 是归 Sun 公司所有的简表，它包含 KJava API。KJava 简表建立在 CLDC 配置的顶部。KJava 虚拟机，KVM，像标准的 J2SE 虚拟机那样接受相同的字节代码和类文件格式。
  KJava 包含一个特定于 Sun 的、运行在 Palm 操作系统上的 API。这个 KJava API 和 J2SE 抽象视窗工具包 (AWT) 有很多地方都是相同。然而，由于它不是一个标准的 J2ME 软件包，它的主软件包是 com.sun.kjava。在以后的教程中开发一些示例应用程序时会学到更多关于 KJava API 的知识。

- **简表 2: MIDP**
  MIDP 适合诸如蜂窝电话和寻呼机等移动设备。MIDP 和 KJava 一样，也是建立在 CLDC 之上的，并且提供一个标准的运行时环境，允许在终端用户设备上动态地部署新的应用程序和服务。
  MIDP 是一个公共的、为移动设备设计的工业标准简表，它不依赖某个特定的商家。对于移动应用程序开发来说，它是一个完整的、受支持的基础。
  MIDP 包含下列软件包，前面 3 个是核心 CLDC 软件包，另加 3 个特定于 MIDP 的软件包。我们将在后面的教程中讨论每个软件包：
  · java.lang
  · java.io
  · java.util
  · javax.microedition.io
  · javax.microedition.lcdui
  · javax.microedition.midlet
  · javax.microedition.rms

- **MIDP 取代 KJava**
  正如我们在前面的章节中提及的，KJava 是 Sun 公司所拥有的一个 API。它不作为一种完整的、功能齐全的简表，而是作为一种示范，示范简表如何与 CLDC 一起工作。根据 CLDC 发行说明 （其中包含 CLDC 下载版本）：
  在软件包 com.sun.kjava 中提供的 GUI 类不是连接限制设备配置 CLDC 的一部分。Java 2 平台袖珍版的正式 GUI 类，将通过 Java Community Process 分别定义并被包含进 J2ME 简表中。
  不管这个事实存在与否，KJava 简表已经被早期的采用者广泛使用了。在 2001 年

JavaOne 的开发商会议上，Sun 公司宣称早期 Palm 操作系统的 MIDP 可用性（MID 简表的最初发行版焦点主要集中在无线电话上）。 Palm 操作系统 MIDP 的规格是由 Java Community Process (JCP) 定义的。因此，它是独立于供应商的。当前众多限制中的一个 -- 它将随时间消失 -- 就是 Palm 操作系统的 MIDP 需要最新的 Palm 操作系统，3.5 版。在我们写这个教程的同时， Palm 操作系统的 MIDP 的规范仍然在随着 JCP 不断发展，其细节也会随时改变。尽管目前还无法确定其发行日期，但是规范一旦出台，将必然减少对 KJava 的需求。

PAGE TO TOP

J2ME 技术

第五章：设置您的开发环境

- **介绍**

在这一章，我们将学习如何下载和安装开发 J2ME 应用程序必需的软件。现在就让我们开始吧，首先学习 Windows 或 UNIX 环境下下载和安装 CLDC。目前 CLDC 1.0 发行版包含 Win32、Solaris 和 Linux 平台上的 CLDC 执行。我们将在您的 Palm 设备上安装 KVM，然后学习如何编译 Palm 数据库开发工具。

下一步，您将学到如何下载和安装 Palm 操作系统仿真器 (POSE) 以及如何把 Palm 设备中的 ROM 映像传送到您的 PC 以便和仿真器一起使用。

最后，我们来看一看如何下载和安装 J2ME 无线工具包，该工具包可用来开发 MIDP 设备的 J2ME 应用程序。

- **在 Win32 或 UNIX 中下载和安装 CLDC**

在 Windows 或 UNIX 平台上安装 CLDC 和 Sun 的 KVM 软件：

1. 下载 CLDC。需要下载和安装的两个软件包为：j2me_cldc-1_0_2-fcs-winunix.zip 和 j2me_cldc-1_0_2-fcs-kjava_overlay.zip。

2. 解压第一个软件包。

o 在 Windows 环境下，您可以把包里的内容解压至根目录 c:\ 下。

o 在 UNIX 环境下，把内容解压至您喜欢的目录下。可以是主目录，但如果您为这台机器的所有用户安装的话，就把它安装在经常安装共享应用程序文件的地方（例如，通常是类似 /usr/local 或 /opt 的目录。）

在安装目录下，会创建一个新的文件夹 j2me_cldc，并且它还包含下列子目录：jam、docs、build、tools、api、kvm、samples 和 bin。

3. 将第二个软件包解压至刚在 CLDC 安装过程中创建的 j2me_cldc 目录下。如：Windows 下的 c:\j2me_cldc，或者 UNIX 下的 /usr/local/j2me_cldc 或 /opt/j2me_cldc。如果出现提示，则覆盖现有的所有文件。

4. 把目录 j2me_cldc/bin 添加到 PATH 中，以免每次运行 CLDC 程序 kvm 和提前验证时都要键入完全路径。

- **在您的 PDA 上安装 CLDC 和 KVM**
  使用 PDA 设备的 HotSync 功能在您的 Palm 操作系统上安装 kvm.prc 和 kvmutil.prc 文件。从 Windows 环境安装以下文件：
  1. 把 PDA 放置在其初始位置。

  2. 在 Palm 设备的桌面上，单击安装图标。

  3. 单击浏览按钮选择目录 c:\j2me_cldc\bin。

  4. 选择文件 kvm.prc 和 kvmutil.prc。注意在相同的目录下有各种其它的 prc 文件 -- 这些文件包含一些示例应用程序，您可能希望安装这些应用程序用来试验。

  5. 在 PDA 的初始位置按下 HotSync 按钮来安装选中的 prc 文件。

  6. 在 PDA 上，定位任一个刚安装的文件。单击 PDA 上的两个文件中的一个图标来装入和运行应用程序。现在就可以设置堆的最大尺寸和屏幕输出选项。

- **编译 Palm 数据库工具**
  通过安装 CLDC 软件包，您已经设置了开发环境。现在您可以访问分别位于 j2me_cldc/docs 和 j2me_cldc/bin/api/classes 中的文档和类。
  j2me_cldc/tools 是和两个软件包同时安装的其它目录中的一个目录，存储着用来生成 .prc 文件的实用程序。这些实用程序允许您在 PDA 上安装 J2ME 应用程序。为了能使用目录 j2me_cldc/tools 中的工具，您必须首先编译类文件：
  1. 进入或转入 j2me_cldc/tools/palm 目录。

  2. 创建一个称为 classes 的子目录。

  3. 通过键入下面的一行命令来编译 .java 源文件：
  4.
  5. javac -d classes src/palm/database/*.java
  现在类已经编译好了，可以在类的子目录中找到。

  6. 把 src/palm/database/Wrapper.prc 和 src/palm/database/DefaultTiny.bmp 文件复制到类目录中去。
  现在您已经成功编译了 Palm 数据库工具的类文件并把它们放在了 j2me_cldc/tools/palm/classes 和它的子目录里。您可以加入整个路径到 Java CLASSPATH 的类的子目录中。或者，当使用 Palm 数据库工具时，您可以把它加入到指定 java 命令行的 CLASSPATH 中。

- **安装 Palm 操作系统仿真器 (POSE)**
  Palm 操作系统仿真器 (POSE) 应用程序软件仿真不同的 PDA 模型。仿真器允许在下载 PDA 应用程序到 PDA 之前，对它们进行开发、测试和调试。POSE 可从 Palm 操作系统仿真器 Web 站点上是免费获取（请参阅参考资料）。
  它的二进制版本仅可用于 Windows。尽管 POSE 也可以在 UNIX 环境中运行，但您必须

从 Palm OS Web 站点上下载源文件然后对它们进行编译，以适合您特定的 UNIX 平台。
在 Windows 下下载和安装 POSE：
1. 从 Palm OS Web 站点上下载 POSE 的最新压缩文件。

2. 将压缩文件中的内容解压至自己的目录下。

3. emulator.exe 文件现在已经在 POSE 的安装目录中了。当启动时，emulator.exe 就运行 Palm 操作系统仿真器。

- **上载 ROM 映像**
为了使用仿真器，目标 PDA 需要一个 ROM 的副本（“ROM 映像”）。ROM 提供一个可以被 POSE 仿真器精确模仿的操作系统。ROM 映像可以从一些制造厂商处获得，但是您也能从自己的 PDA 上下载一个 ROM 映像。
为了察看您的桌面或工作站上的 Palm 计算机的 ROM 映像，您可以使用 POSE 从 PDA 上下载 ROM 映像。
注意：经常变动的用户 (handspring users) 必须使用常规的串行电缆和 COM 端口，而不能使用 USB 电缆。
操作步骤：
1. 启动与 Palm 设备一起提供的 Palm 桌面软件，然后单击安装图标。

2. 浏览 POSE 目录并选择 ROM Transfer.prc 文件。

3. 把 Palm 设备放置在其初始位置。

4. 在初始位置按下 HotSync 按钮，开始安装文件。当完成这个过程时，会有一个 ROM Transfer 的图标显示在您的 Palm 设备上。

5. 为传送 ROM 映像，您必须退出 HotSync 管理器，并确认它不在 PC 上运行。然后把您的 Palm 设备留在初始位置。

6. 在您的 Palm 设备上单击 ROM transfer 图标。
7. 在您的 PC 上，转到 POSE 目录下并运行 emulator.exe 程序。出现模拟器窗口。

8. 在您的 PC 上，从菜单中选择 Download 按钮。在您的 Palm 设备上，单击 Begin Transfer 按钮。ROM 映像的传送需要几分钟。

9. 当传送完成时，会提示您选择一个目录来保存文件。在 POSE 目录中以 palm.rom 为名保存该文件。

10. 在 Palm 操作系统仿真器 窗口中，选择 New。出现 New Session Emulator 窗口。

11. 在 New Session Emulator 窗口中选择恰当的设置，然后单击 Browse 按钮选择刚刚传送过的 ROM 文件、palm.rom。

12. 最后，单击 OK 按钮。
如果传送成功，可以看见一个 Palm 设备的映像加载在您的 PC 或工作站监视器上。

- **下载和安装 J2ME 无线工具包 (J2ME Wireless Toolkit)**
  J2ME 无线工具包提供一个完整的开发环境来编写和测试 MIDP 应用程序。下载包括工具、文档和仿真环境，例如，一个与 Forte for Java 集成的模块。
  目前，J2ME 无线工具包仅支持 Windows 98 第二版、Windows NT 4.0 和 Windows 2000 是可用的，不支持 Windows 95。J2ME 无线工具包的 Solaris 和 Linux 版本在本教程编写的同时正处于筹划之中。
  在 Windows 环境下安装 J2ME 无线工具包请按如下步骤：
  1. 下载 J2ME 无线工具包。
  2. 运行 j2me_wireless_toolkit-1_0_1-fcs.exe 安装无线工具包。
  当系统提示您提供安装目录时，请确保该安装目录的完全合法的路径中不包含任何空格。这将帮助您在今后使用工具包时避免可能出现的问题。
  如果您计划使用 Forte for Java 进行开发的话，在 Setup Type 对话框中选择 Integrated setup。

PAGE TO TOP

第六章：CLDC API

- **介绍**
  迄今为止，我们知道 CLDC 比较适合 J2ME，并且也已经建立了我们的开发环境。在这一章中，我们将更深入地探索 CLDC API。CLDC API 实际上只是 J2SE 的一个子集，它包括 java.lang、java.io 和 java.util，另加一个新软件包 -- javax.microedition。我们将逐个来研究这些软件包，并突出显示每一个包中的重要的类。
  尽管每一个类都在 J2SE 中，但是没有必要让每一个类的 CLDC 实现都能实现 J2SE 支持的所有方法。您可以检查 CLDC API 文档以确认哪些方法是受支持的。文档的副本位于安装 J2ME CLDC 时创建的 j2me_cldc/docs 目录下。它提供 PDF 和 javadoc 两种格式。

- **java.lang**
  CLDC java.lang 软件包是 J2SE java.lang 软件包的一个子集。与 J2SE 相比，它最引人注目的可能便是冗长的浮点操作了，特别是浮点 (Float) 和双精度 (Double) 类。如果使用浮点的话，这些冗余将涉及到所有其它的类。
  相对于 J2SE v1.3 API，CLDC API 中删去了几个其它的类。其中包括 ClassLoader、Compiler、InheritableThreadLocal、Number、Package、Process、RuntimePermission、SecurityManager、StrictMath、ThreadGroup、ThreadLocal 和 Void。
  我们描述了可从下面几页表中的 CLDC java.lang 软件包中获取的主要的类。Java 开发人员对所有这些类的使用都应该是非常熟悉了。
  除这些核心类之外，您还将看到 CLDC 支持的 Runnable 接口，正象 Exception 、Error 和其它有关的类一样。

- **java.lang 核心运行时类**
  java.lang 软件包的核心运行时类有：
  · Class -- 显示正在运行的 Java 应用程序中的类和接口。

  · Object -- 与在 J2SE 中相同，Object 是所有 Java 对象的基本类。

  · Runtime -- 为 Java 应用程序提供一种与运行时环境（Java 应用程序在其中运行）进行交互的方法。

  · System -- 提供一些静态的帮助方法，就像为 J2SE 提供方法一样。

  · Thread -- 定义 Java 程序的一个执行线程。
  · Throwable -- Java 语言中所有错误和异常的超级类。

- **java.lang 核心数据类型类**
  java.lang 软件包中的核心数据类型类有：
  · Boolean -- 包装 boolean 原始数据类型。
  · Byte —- 包装 byte 原始数据类型。
  · Character -- 包装 char 原始数据类型。
  · Integer -- 包装 int 原始数据类型。
  · Long -- 包装 long 原始数据类型。
  · Short -- 包装 short 原始数据类型。

- **java.lang 帮助类**
  java.lang 软件包的帮助类有：
  · Math -- 包含执行基本数学运算的方法。请注意，所有执行浮点值运算的方法都被省略了，仅保留了关于 integers 和 longs 的方法：abs()、min() 和 max()。

  · String -- 在 Java 中代表对象 String，就像在 J2SE 中一样。

  · StringBuffer -- 代表一个可以修改的串，就像在 J2SE 中一样。

- **java.io 输入类**
  CLDC API 包含许多 J2SE 中共同使用的输入类。特别地，CLDC java.io 软件包中包括下面一些类：
  · ByteArrayInputStream -- 包含一个内部缓冲器，它代表可能从输入流中读取的字节。

  · DataInput -- 一个接口，从二进制输入流提供字节以供读取并把它们转换成原始 Java 数据类型。DataInputStream 提供该接口的实现。

  · DataInputStream -- 允许应用程序以独立于平台的方式从基层输入流中读取原始 Java 数据类型。

· InputStream -- 一个抽象类，它是所有代表字节输入流的类的超级类。

· InputStreamReader -- 读取字节并把它们按照指定的字符编码方法转换成字符。
· Reader -- 一种读取字符流的抽象类。
注意：其中一些类可能不包含 J2SE 姊妹版支持的所有方法，就像在 java.lang 软件包中一样。尤其是省略了浮点和双精度方法。

- **java.io 输出类**
  CLDC API 包含了许多 J2SE 中的共同使用的输出类。特别是，CLDC java.io 软件包中包括下面一些输出类：
  · ByteArrayOutputStream -- 实现一个输出流，在此输出流中数据被写入字节数组。

  · DataOutput -- 一种接口，提供原始 Java 数据类型以供写入二进制输出流。
  DataOutputStream 提供该接口的实现。

  · DataOutputStream -- 一个输出流，允许应用程序以一种便捷的方式编写原始 Java 数据类型。

  · OutputStream -- 一个抽象类，它是所有代表字节输出流的类的超级类。

  · OutputStreamReader -- 给出字符，并按指定的字符编码方法将其转换为字节。

  · PrintStream -- 添加一种便捷的方法来打印数据值的文本表现形式。
  · Writer -- 编写字符流的一个抽象类。
  其中一些类可能不包含 J2SE 支持的所有方法，比如浮点和双精度方法。

- **java.util 收集类**
  CLDC java.util 软件包中包含 J2SE java.util 软件包中最常用的类。这些类中包括四个收集类（实际是三个收集类和一个接口），以及日期／时间和实用程序类。
  CLDC 支持的 java.util 收集类有：
  · Enumeration -- 一个接口，通过项目集允许对例程进行重复调用。

  · Hashtable -- 实现 hashtable，将键映射到值。

  · Stack -- 代表了一个后进先出 (LIFO) 的对象集合或堆栈。

  · Vector -- 代表可以调整大小的对象“数组”或者矢量。

- **java.util -- 其它的类**
  CLDC 支持的 java.util 类中其余部分包括日期和时间类，以及 Random 实用程序类。下表中简要列出了这些类。
  · Calendar -- 一个抽象类，使用一套整型字段如 YEAR、MONTH、 DAY 等来获取和设置日期。

  · Date -- 代表特定的时间和日期，精确到毫秒级。

· Random -- 一个实用程序类，用来生成 int 或 long 的随机值流。

· TimeZone -- 代表时区的偏移量，也用于校正时间。

- **javax.microedition.io**
  迄今为止，我们在 CLDC API 中看到的所有的类都是 J2SE API 的子类。CLDC 还包含一个附加的软件包 -- javax.microedition.io。
  在这个包里唯一被定义的类就是 Connector 类，也称为工厂类，包含创建 Connection 对象或输入、输出流的方法。
  当动态识别一个类的名字时，Connection 对象就被创建了。类名称的识别基于平台名称和被请求连接的协议。描述目标对象的参数串应该满足 RFC 2396 规范所要求的格式。请使用下列格式：
  {scheme}:[{target}][{params}]
  {scheme} 是一个协议的名称，如 http 或 ftp。{target} 通常是一个网络地址，但是面向非网络的协议则可能把它当作一个相当灵活的字段来处理。还有一些参数，如 {params} 被指定为一系列形如 ";x=y" 的分配形式（例如，;myParam=value）。

- **javax.microedition.io 帮助接口**
  除类属连接工厂类之外，javax.microedition.io 软件包中还包含下列面向连接的接口：
  · Connection -- 定义了最基本的连接类型。这个接口也是此软件包中所有其它连接接口的基本类。

  · ContentConnection -- 定义了一个可以通过内容的流连接。

  · Datagram -- 定义了一个类属数据报接口。

  · DatagramConnection -- 定义了类属数据报连接和它必须支持的性能。

  · InputConnection -- 定义了一个类属输入流连接和它必须支持的性能。

  · OutputConnection -- 定义了一个类属输出流连接和它必须支持的性能。

  · StreamConnection -- 定义了一个类属流连接和它必须支持的性能。

  · StreamConnectionNotifier -- 定义了一个流连接的通告程序必须具有的性能。

第七章：使用 Kjava GUI 组件的开发

- **介绍**
  本章中，我们来学习如何使用 KJava API 进行 GUI 开发。首先看一下 KJava GUI 开发的介绍，然后使用 KJava API 开发我们的第一个 J2ME 应用程序。HelloWorld ，将示范一个使用 CLDC 的简易 J2ME 应用程序、KJava 简表以及 Palm 操作系统的 KVM。
  在下一章我们继续进行 KJava GUI 开发，构建另一个应用程序并重点学习事件处理模块。

- **Spotlet 介绍**
  KJava API 提供了一套开发 Palm 操作系统设备应用程序的类。KJava 提供了一个 Spotlet 类，com.sun.kjava.Spotlet，它和 J2SE Canvas 类在添加用于事件处理的回调方法上类似。因此，应用程序可以扩展 Spotlet 类，不使用合适的事件处理方法也可提供需要的功能。应用程序可以创建并使用多个 spotlets 来显示不同的窗口。就像使用 J2SE Canvas（一个负责画出自身以及放置在其上的 GUI 控件的 spotlet）一样。
  在我们的两个 KJava 示例中，都将使用 Spotlet 类。这两个示例中一个是很快就看到的 HelloWorld 应用程序，另一个是 Scribble 应用程序，后者将在使用 KJava 事件处理的开发这一章中构建。

- **KJava 应用程序 HelloWorld**
  这个应用程序将在屏幕中央显示 "Hello World!" 和一个 Exit 按钮，按下后即终止该应用程序。HelloWorld.java 开始时使用下面的几行代码导入将在后面的 HelloWorld 类中使用的类:

  ```
  import com.sun.kjava.Button;
  import com.sun.kjava.Graphics;
  import com.sun.kjava.Spotlet;
  ```

  下面的代码行将 HelloWorld 类定义为扩展 Spotlet：

  ```
  public class HelloWorld extends Spotlet
  ```

  请记住 Spotlet 类提供用于处理事件的回调功能。在这个简单的示例中，我们只对一个事件感兴趣，即用户何时按下 Exit 按钮。下一个代码行存储对 Exit 按钮的引用：

  ```
  private static Button exitButton;
  ```

  如同在 J2SE 中一样，main() 方法定义程序的主要入口点。对于 J2ME 应用程序，main 也定义了入口点。在本例中，main() 创建了一个新的 HelloWorld 类的实例，它运行我们的应用程序。

  ```
  public static void main(String[] args)
  {
  (new HelloWorld()).register(NO_EVENT_OPTIONS);
  }
  ```

  下一个代码块定义了构造程序。在构造程序中，我们首先创建一个 Button 并为其加上 "Exit" 标签。按钮起初是不可见的。当我们得到对图形对象的引用后，此按钮成了一个可画的屏幕，先清屏然后在屏幕中央画出文本 "Hello World!"。最后，我们在屏幕上添加 Exit 按钮。

```
public HelloWorld()
{
// Create (initially invisible) the "Exit" button
exitButton = new Button("Exit",70,145);

// Get a reference to the graphics object;
// i.e. the drawable screen
Graphics g = Graphics.getGraphics();
g.clearScreen();

// Draw the text, "Hello World!" somewhere near the center
g.drawString("Hello World!", 55, 45, g.PLAIN);
// Draw the "Exit" button
exitButton.paint();
}
```

最后，我们定义 penDown 事件处理程序，用来简单地检查 Exit 按钮是否被按下。如果已按下，就退出应用程序。

```
public void penDown(int x, int y)
{
// If the "Exit" button was pressed, end this application
if (exitButton.pressed(x,y))
System.exit(0);
}
```

- **HelloWorld -- 完整的代码清单**

  以下便是 Palm 设备的 HelloWorld 应用程序的完整代码示例:

```
import com.sun.kjava.Button;
import com.sun.kjava.Graphics;
import com.sun.kjava.Spotlet;

/**
* Simple demonstration, "Hello World" program. Note that Spotlet is
* the class that provides callbacks for event handling.
*/
public class HelloWorld extends Spotlet
{
/** Stores a reference to the "Exit" button. */
private static Button exitButton;
/**
* Main entry point for this program.
*/
public static void main(String[] args)
{
(new HelloWorld()).register(NO_EVENT_OPTIONS);
}
```

```
/**
* Constructor: draws the screen.
*/
public HelloWorld()
{
// Create (initially invisible) the "Exit" button
exitButton = new Button("Exit",70,145);

// Get a reference to the graphics object;
// i.e. the drawable screen
Graphics g = Graphics.getGraphics();
g.clearScreen();

// Draw the text, "Hello World!" somewhere near the center
g.drawString("Hello World!", 55, 45, g.PLAIN);

// Draw the "Exit" button
exitButton.paint();
}
/**
* Handle a pen down event.
*/
public void penDown(int x, int y)
{
// If the "Exit" button was pressed, end this application
if (exitButton.pressed(x,y))
System.exit(0);
}
}
```

- **KJava GUI 组件**
  除 Spotlet 类之外，KJava API 还定义了一些基础 GUI 组件。下面列出了由 KJava 提供的一些更基础 GUI 组件。注意它们与 J2SE AWT 同名组件的相似性。
  · Button -- 定义了一个简单的 GUI 按钮。按钮可以包含文本标签如 "OK" 或 "Cancel"，也可以包含位图图象。

  · Checkbox -- 定义了一个 GUI 复选框组件，它可以是已选中的，也可以是未选中的。

  · Dialog -- 定义了一个弹出式、模式对话框，包含标题、文本字符串和一个 "Dismiss" 按钮。
  · Graphics -- 这个类和其 J2SE 姊妹版很类似，提供各种绘图的方法。

  · RadioButton -- 定义了一个有两个状态的单选按钮。通常被用作一组使用 RadioGroup 对象分组的 radio 按钮的一部分，在某一时刻只能使用一个。

  · RadioGroup -- 代表一组单选按钮，在某一时刻只能有一个处于开着或选中状态。

  · ScrollTextBox, SelectScrollTextBox -- 定义了一个带滚动条的文本框组件，用户可在该组件中输入多行文本。它和 J2SE TextArea AWT 组件功能相似。

· Slider -- 定义了一个图形化滑块，使用该组件，用户可以沿着刻度尺拖动标志来选择一个值。

· TextBox -- 定义了一个基本的文本框，但仅用于显示少量文本。对于大量文本，请使用 ScrollTextBox。
· TextField -- 定义了一个文本框提供给用户进行输入。与 J2SE TextField AWT 组件相似。

· ValueSelector -- 一个接受用户输入的整型值的 GUI 组件。用户可以选择 "+" 来递增该值，也可以选择 "-" 来递减该值。

- **其它的 KJava 类**
  KJava 定义了一些附加的类。在早期的开发工作中很少用到它们，但是它们的用处还是很值得一提的，因为说不定今后开发时就会用到它们。
  · Bitmap -- 表示一个黑白两色的位图图象。

  · Caret -- 仅被 TextField 使用。（API 文档指出这个类可能是 TextField 类私有的）。

  · Database -- 给 Palm 操作系统数据库管理器提供一个接口。

  · DialogOwner -- 由希望显示模式对话框的类使用的接口。

  · HelpDisplay -- 定义了一个简单的帮助对话框。

  · IntVector -- 并不是真正意义上的 GUI 组件，该类提供一个可扩展的整型矢量，很象 java.util.Vector。
  · List -- 并不是真正意义上的 GUI 组件，它是另一个代表一列对象的帮助类，就象 java.util.Vector。

  · ScrollOwner -- ScrollTextBox 使用的类。

  · VerticalScrollBar -- 定义了一个垂直滚动条组件。

PAGE TO TOP

第八章：使用 Kjava 事件处理的开发

- **介绍**

在这一章，我们将学习 KJava 事件处理，并用简单的绘图应用程序 Scribble 来示范它是如何工作的。

KJava 事件处理模型不如 J2SE 的 action-listene 模型先进。通过使 Spotlet 类子类化，所有感兴趣的事件都是可访问的，无论怎样，KJava 应用程序都将完成这项工作。目前，只有 spotlet 受到了事件的关注。为了关注 spotlet，我们使用 register()。如要停止，则使用 unregister()。

注意：如果您用 WANT_SYSTEM_KEYS 注册一个 spotlet ，设备不会通过按下按钮和排队等候停止它的应用程序来自动终止这个应用程序。相反，按下按钮事件会通报这个应用程序，然后负责适当地处理事件。除非当按下按钮时您提供一种通过调用 System.exit 终止应用程序的方法，否则这个应用程序将会继续不确定地运行。唯一能终止这个应用程序的方法就只有重启该设备。

KJava 支持三种基本类型事件：显示屏上笔的移动、键盘输入和电子束定向发送／接收。另外，还有一种全面的包罗万象的方法 -- unknownEvent()。在后面的章节中我们将讨论这些不同的事件类型。

- **处理笔的移动**

  处理 PDA 显示器上笔的移动的事件处理程序有：penDown、penMove 和 penUp。

  当用户将笔在显示器上移动时，penDown() 方法将被调用，它传递显示器上笔的放置点的 X 和 Y 轴坐标。

  public void penDown(int x, int y)

  当用户在显示器上移动笔时 penMove() 过程将被调用。X 和 Y 轴坐标定义笔的当前位置。

  public void penMove(int x, int y)

  当用户将笔从显示器上移开时 penUp() 过程将被调用，它传递两个参数：笔被移开点的 X 和 Y 轴坐标。

  public void penUp(int x, int y)

- **处理键盘输入，电子束定向发送／接收，以及未知事件**

  在 J2SE AWT 中，接口 java.awt.event.KeyListener 包含处理不同键盘事件的 keyPressed、keyReleased 和 keyTyped。与此相比，KJava 则只有一个函数，keyDown()。如果用户在可书画区写下一个字符，按下计算器或菜单图标，或者是按下任何“硬键”（缺省情况下，Date Book、Address、page up、page down、To Do List 或是 Memo Pad key ）时，事件 keyDown 就会被调用。参数 keyCode 标识用户输入的键的代码。如果按了其中一个“硬键”，事件 keyDown 就开始匹配这个类中定义过的相应常量中的一个。

  public void keyDown(int keyCode)

  beamReceive() 方法被用于接收从红外线 Palm 设备传来的数据包。数据以一种字节数组的方式被接收，并用虚拟机自动分配这些数据。

  public static boolean beamReceive(byte[] data)beamSend() 方法不是一个事件处理程序，但是它显然与 beamReceive() 相关联，所以我们在这儿还是要提一下。这种方法被用来给发送到另一个红外线 Palm 设备的数据包定向。在给数据定向时，您可以调用这个函数，但是目标设备必须在接收数据的 spotlet 中注册一个 beamReceive 处理器。

  public static boolean beamSend(byte[] data)

  unknownEvent 是一个常规的所有未知事件处理例程。

  public void unknownEvent(int event, java.io.DataInput in)

- **Scribble 应用程序介绍**

现在我们了解了事件处理的基础，我们即将进入更为高级一些的 J2ME 开发。在本章中，我们将开发 Scribble 应用程序。我们不会逐行描述代码，但是，我们将描绘重要的代码行和每个方法的目的。您可从参考资料上访问完整的代码清单。
Scribble 应用程序是一个独立的应用程序，示范使用 Sun 公司的 KJava API 开发 Palm 操作系统。Scribble 允许您在显示屏上徒手绘图，或添加想要的正规文本。

- **开始使用 Scribble**
  在 HelloWorld 应用程序中，我们需要导入 Scribble 应用程序要使用的类。下面是标识被 Scribble 使用的类的文本块:

  ```
  import com.sun.kjava.Bitmap;
  import com.sun.kjava.Button;
  import com.sun.kjava.Dialog;
  import com.sun.kjava.Graphics;
  import com.sun.kjava.HelpDisplay;
  import com.sun.kjava.Spotlet;
  ```

  Scribble 类扩展了 J2SE 中使用的 Spotlet 类。它从定义最终的静态类变量开始。

  ```
  public class Scribble extends Spotlet
  ```

  类变量 g 担当了对单独的 Graphics 对象的引用，该对象在类中始终被使用：

  ```
  static Graphics g = Graphics.getGraphics ();
  ```

- **定义方法和事件处理程序**
  此外，main() 定义了应用程序的主要入口。

  ```
  public static void main(String[] args)
  ```

  默认的构造程序，Scribble 初始化成员变量，清屏并画出初始框架。

  ```
  public Scribble()
  ```

  paint() 方法负责更新或刷新显示。它使用类变量 g -- 一个 Graphics 对象，类似于 Java 2 AWT 中使用的 Graphics 对象。

  ```
  private void paint()
  ```

  penDown() 方法执行事件处理程序来处理在屏幕上放置笔的事件。它通过 X 和 Y 坐标来定位。在 Scribble 中，程序测试 Clear 或 Exit 按钮是否被按下，如果按下的话，就处理相应的事件。

  ```
  public void penDown(int x, int y)
  ```

  keyDown() 方法处理那些在 Palm 设备的即兴书画框内随手写下的东西。传送到这个方法中的整型值 keyCode 就是输入的字符键值。在 Scribble 应用程序中，我们存储了成员变量 lastKey 中被按下的键，然后调用 paint() 方法刷新屏幕。

  ```
  public void keyDown(int keyCode)
  ```

  penMove() 方法处理在屏幕上拖动笔的事件。在 Scribble 中，它负责用笔绘画。

  ```
  public void penMove(int x, int y)
  ```

  使用的最后一个方法 clearDrawingArea()，在用户按下 Clear 按钮时由 penDown 事件处理程序调用。由于它仅在 Scribble 类的内部使用，所以 clearDrawingArea() 是一个私有方

法。

private void clearDrawingArea()

PAGE TO TOP

J2ME 技术

第九章：MIDP API

- **介绍**
  移动信息设备简表 (MIDP) 适合类似于蜂窝电话和寻呼机这样的设备。MIDP，就象 KJava 一样，同样也建立在 CLDC 之上。MID 简表提供一种标准的运行时环境，允许在终端用户设备上动态地配置新的应用程序和服务。
  在本章中，我们将详细地讨论 MID 定义的七个软件包。我们还将建立一个 MIDP 应用程序示例。

- **UI 设计注意事项**
  MIDP 包括一个低级的 UI API 和一个高级的 UI API。低级的 API 允许您完全访问一个设备的显示屏，也允许访问原始键和指针事件。然而，使用低级 API 时， 没有可用的用户界面控件。应用程序必须精确地绘制出按钮和其它所有的控件。
  相反，高级 API 提供简单的用户界面控件但不能直接访问原始的输入事件或显示屏。由于显示屏的尺寸和 MIDP 设备输入方法的差异，控件显得很抽象。MIDP 的实现确定了绘制控件的方法，也确定了如何管理用户输入。
  让我们在后面的章节里更进一步了解 MIDP 的软件包和类。

- **MIDP API**
  MIDP 包含四个核心 CLDC 软件包 (java.lang、java.io、java.util 和 javax.microedition.io)，另加下面的三个特定于 MIDP 的软件包：
  · javax.microedition.lcdui
  · javax.microedition.midlet
  · javax.microedition.rms
  我们将在本章的后面部分详细介绍特定于 MIDP 软件包。除了上面新的软件包之外，MIDP 还向核心 CLDC 软件包添加了四个新类，如下所示。
  · java.util.Timer -- 用于为后台线程中将来要执行的任务确定时间。

  · java.util.TimerTask -- 被 java.util.Timer 类使用，用来为后台线程中稍后的执行定义任务。

  · javax.microedition.io.HttpConnection -- 一个接口，为 HTTP 连接定义必要的方法和常量。

  · java.lang.IllegalStateException -- 一个 RuntimeException， 指出在不合法或不合适的时

间已经调用的一个方法。

- **MIDlet 介绍**
  MIDlet 是一个 Java 类，它扩展了 javax.microedition.midlet.MIDlet 抽象类。实现 startApp()、pauseApp()和 destroyApp()方法，这些方法类似于 J2SE 的 java.applet.Applet 类中的 start()、stop()和 destroy()方法。
  除了扩充 javax.microedition.midlet.MIDlet 的主 MIDlet 类之外，一个 MIDP 应用程序通常还包括其它一些类，这些类能随它们的资源一起被打包成为 jar 文件 -- 称之为 MIDlet 套件。一个 MIDlet 套件中的不同 MIDlet 能共享 jar 文件的资源，尽管不同套件中的 MIDlets 不能直接相互作用。
  MIDlet 在应用程序生命周期中有三种可能的存在状态 -- 运行状态、暂停状态、销毁状态。运行状态，正如其名称所暗示的，意味着 MIDlet 正在运行中。这种状态始于 startApp 方法被调用时。在暂停状态中，MIDlet 持有的所有资源将被释放，但是它准备着再次被运行。调用 notifyPaused 方法时，MIDlet 处于暂停状态。在销毁状态中，MIDlet 已经永久地将其自身关闭，释放所有的资源，等待着废物清理程序的处理。它是通过 notifyDestroyed 方法来调用的。
  在接下来的两页中，我们来看一个简单的 HelloWorld MIDlet。

- **HelloWorld MIDlet**
  与使用 KJava HelloWorld 应用程序一样，这个 MIDlet 也会在 MIDP 设备的显示屏上显示 "Hello World!" 和 Exit 按钮，按下该按钮会终止应用程序。
  HelloWorld.java 文件以下面的代码行开始，这些代码行导入稍后会在 HelloWorld 类中使用的类:

  ```
  import javax.microedition.midlet.MIDlet;
  import javax.microedition.lcdui.Command;
  import javax.microedition.lcdui.CommandListener;
  import javax.microedition.lcdui.Display;
  import javax.microedition.lcdui.Displayable;
  import javax.microedition.lcdui.Form;
  ```

  由于 HelloWorld 类是一个 MIDP 应用程序，它扩展了 MIDlet。它也实现 CommandListener 接口来处理事件：

  ```
  public class HelloWorld extends MIDlet implements CommandListener
  ```

  下面的方法是一个缺省构造程序，它创建一个新表单，在上面初始化控件，然后显示出来：

  ```
  private Form form;

  public HelloWorld()
  {
  // Create a new form on which to display our text
  form = new Form("Test App");

  // Add the text "Hello World!" to the form
  form.append("Hello World!");
  ```

```
// Add a command button labeled "Exit"
form.addCommand( new Command( "Exit", Command.EXIT, 1 ) );

// Register this object as a commandListener
form.setCommandListener( this );
}
```

调用 startApp() 方法启动应用程序与小应用程序的启动方法很象。在 MIDle 的一次执行中它可能会被调用多次。如果 MIDlet 暂停，pauseApp() 将会被调用。要重新启动 MIDlet，需调用 startApp()。仅须执行一次的主初始化代码应该放置在构造程序中：

```
public void startApp()
{
// Get a reference to the display, and show the form
Display display = Display.getDisplay(this);
display.setCurrent( form );
}
```

pauseApp() 被调用使得 MIDlet 处于暂停状态。在此应用程序中，当进入暂停状态时，我们没执行任何操作；但是我们仍然需要在 MIDlet 中实现 pauseApp 方法，因为它是父 MIDlet 类中的抽象方法。

```
public void pauseApp() { }
```

destroyApp() 被调用，破坏了 MIDlet 并使其处于销毁状态。在此应用程序中，我们通过将引用设为 null，释放了对表单的引用。

```
public void destroyApp(boolean unconditional)
{
form = null;
}
```

commandAction() 方法是事件处理程序，被请求实现 CommandListener 接口。目前，它破坏了应用程序并通知应用程序管理软件 MIDlet 已经完成。

```
public void commandAction(Command c, Displayable d)
{
// Destroy this MIDlet
destroyApp(true);

// Notify the application management software that this MIDlet
// has entered the destroyed state
notifyDestroyed();
}
```

- **MIDP 软件包**

  除标准 CLDC 软件包之外，MIDP 还包含三个附加的软件包：

  · javax.microedition.lcdui -- 定义用来控制 UI 的类。这个软件包既包含高级 UI 类（例如 Form、Command、DateField 和 TextField 等），又包含低级 UI 类（允许用低级方式控制 UI）。

  · javax.microedition.midlet -- 包含 MIDP 主类中的一个，MIDlet 类，为 MIDP 应用程序提供访问关于其运行所在环境信息的权限。

javax.microedition.rms -- 定义一个类的集合，这些类为 MIDlet 提供了永久存储并随后重新得到数据的机制。

PAGE TO TOP

J2ME 技术

第十章：CDC API

- **介绍**
  在这一章中，我们将描述 CDC 的目标设备和这些设备的要求。同时您也将了解到 CDC 支持的软件包和类。
  由于本教程重点针对小型移动设备，我们将不会象学习 CLDC 是那样深入研究 CDC API。然而，我们会识别 CDC 所使用的 J2SE 软件包和类，以及 CDC 给 J2SE 软件包带来的附加功能。

- **CDC 的目标设备**
  CDC 允许您为消费性电子产品和嵌入式设备开发应用程序，例如智能电话、双向寻呼机、PDA、家用电器、销售网点终端以及汽车导航系统等。这些设备运行 32 位微处理器，拥有超过 2 兆的内存，用于存储 C 虚拟机和库。
  CDC 运行在 C 虚拟机 (CVM) 的顶部，与基础表关联在一起。基础表 (FNDp) 是一套 Java API，专为要求定制用户界面 (UI) 的高端设备服务，通常由设备制造商提供。

- **CDC API 概述**
  CDC 是建立在 CLDC 顶部的 API，是整个 J2SE API 的一个更完整的子集。它还包含一个额外的软件包 -- javax.microedition.io 软件包 -- 包含 CLDC 中定义的所有相同的类和接口，及其它。
  CDC 中的一些更值得注意的功能是 CLDC 中所没有的：
  · 支持浮点数（包括 java.lang.Float、java.lang.Double 和 java.lang.StrictMath 类）

  · classloader 类 (java.lang.ClassLoader)

  · 支持本地进程 (java.lang.Process)

  · 高级多线程支持（包括支持线程组和更多线程）
  · 串行化的类 (java.io.Serialiable 和 java.io.Externalizable)

  · 映象 API（包括 java.lang.reflect 软件包）

· 文件系统支持

· 支持 J2SE 类型网络 (java.net)

· 对 J2SE Collections API 更完全的支持

· 为 javax.microedition.io 软件包增加一个 HttpConnection 接口。这样可为 HTTP 连接提供必要的方法和常量。

· 支持 J2SE 的 java.lang.ref、java.math、java.security、java.security.cert、java.text、java.util.jar 和 java.util.zip 软件包。

PAGE TO TOP

第十一章：总结

- **总结**
  在本教程中，我们已经考察了 J2ME 的背景，研究了 J2ME 的配置和简表。让我们再来看看怎样为开发 J2ME 应用程序设置开发环境。
  我们讨论了一些论题，如与连接限制设备配置 (CDLC) API 共同使用的 K 虚拟机 (KVM) 和 KJava 简表，以及也使用 CLDC 的移动信息设备简表 (MIDP) 。同时我们还简要介绍了用于大型应用程序的连接设备配置 (CDC)。
  最后，教您一步步建立了一个简单的 HelloWorld 应用程序，让您了解了能用 J2ME 做些什么。我们还使用 CLDC 和 KJava 开发了 Scribble（一个基本的绘图应用程序）和一个小型的 MIDP 应用程序。
  虽然仍处于成长阶段，适用于移动设备的 Java 已经改变了人们的商务和个人通讯方式。随着 J2ME 的发展和移动设备技术的更加成熟，支持商务和个人移动通讯的应用程序也会得到发展和普及。

- **参考资料**
  请使用下面的参考资料深入探讨 J2ME 并扩展您的无线编程知识。
  · KJava 示例应用程序 zip 文件包含 HelloWorld.java 和 Scribble.java 的完整源代码以及支持文件。

  · MIDP 示例应用程序 zip 文件包含 HelloWorld.java 的完整源代码以及与 J2ME 无线工具包一起使用的支持文件。

  · 欲获得全面的 J2ME 知识（包括本教程中使用的所有 API 和 VM 下载），请访问官方 J2ME 主页。

  · Sun 和 IBM 均提供各种版本的 Java 2 SDK。

· 下载 Palm 操作系统仿真器 (POSE)。

· IBM VisualAge Micro Edition，最近被 JavaPro 杂志提名为最好的嵌入式开发工具，将为您提供快速创建无线设备的嵌入式 Java 应用程序所需的所有支持。

· 您对自 1999 年以来 J2ME 所发生的改变感兴趣吗？请阅读 Todd Sundsted 的 "J2ME grows up"（developerWorks，2001 年 5 月），它深受近来由 Sun Java 提倡者 Bill Day 主持的全天技术会议的影响。

· 如果您有关于 J2ME 的问题，请访问我们的无线 Java 编程论坛获取帮助。

· 无线编程很好地扩展了 Java 平台的应用领域。请访问 developerWorks 上的 Wireless page 获取有关信息。

PAGE TO TOP