CHAPTER 3

第3章 J2ME及移动开发工具

开发 J2ME 手机游戏程序，通常要使用 WTK、Eclipse 等工具，因此配置开发环境是进行 J2ME 程序开发的第一步。本章将介绍 J2ME 开发环境 WTK 和 Eclipse 的安装和配置方法。

在本章，读者将学习到：

- ◇ J2ME 的体系结构；
- ◇ WTK 的开发环境；
- ◇ Eclipse 开发环境的配置方法；
- ◇ 简单的 J2ME 游戏开发过程。

3.1 J2ME 简介

J2ME(Java 2 Micro Edition)是 Java 2 的一个组成部分，它与 J2SE、J2EE 并称。根据 Sun 公司的定义：J2ME 是一种高度优化的 Java 运行环境，主要针对消费类电子设备，如蜂窝电话和可视电话、数字机顶盒、汽车导航系统等。

3.1.1 三层体系结构

J2ME 技术在 1999 年的 JavaOne Developer Conference 大会上正式推出，它将 Java 语言的与平台无关的特性移植到小型电子设备上，允许移动无线设备之间共享应用程序。J2ME 通过配置和简表来设置 Java 运行环境(JRE)。

J2ME 的体系分为 3 层，如图 3-1 所示。其中，最底层是操作系统(Operating System)，可以是 Linux、Symbian 或者 PalmOS，这充分说明了 Java 语言的平台无关性；中间层是配置(Configuration)，包含了基础运行环境，这个环境拥有运行 Java 程序的 Java 虚拟机；上层是简表(Profiles)，是针对一系列设备提供的开发包集合，是给定配置上的一组 API 的集合；顶层

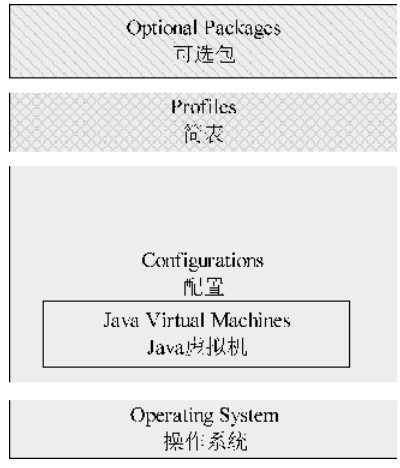


图 3-1 J2ME 的体系结构

是一些可选包，如蓝牙组件、红外组件等。

3.1.2 J2ME 配置、简表和规范

J2ME 是专门应用在微型设备当中的，但在微型设备中存在着巨大的差异。例如，一个典型的个人数据助理(PDA)有比移动电话大得多的屏幕、内存和更快的处理器。为了支持这些差异性，J2ME 引入了配置(Configuration)和简表(Profile) 的概念。基本上来说，配置描绘了一组设备要求(内存和连接性等)，而简表是在给定配置之上的 API，它为某一范围内的设备提供了特定的功能和能力。

1. 配制

配置(Configuration)定义了一个设计在一系列类似硬件上运行的 Java 平台。本质上，只是提供了一个 J2SE 的最小集。在配置中规定了所支持的 Java 语言特征、Java 虚拟机特征和所支持的基本 Java 类库和 API。目前，J2ME 定义了两种配置：将有稳定的电源供应、自用资源较少的设备划为分连接设备配置(Connected Device Configuration，CDC)，如车载设备、机顶盒等；将使用电池供电、设备性能有限的设备划分为受限连接设备配置(Connected Limited Device Configuration，CLDC)，如手机。图 3-2 所示的 CLDC 是 CDC 的一个子集，但是由于 CDC 和 CLDC 针对的设备不同，所以它们使用的虚拟机和核心类库也不相同，CDC 相对于 CLCD 所包含的库的范围要大一些。CDC 和 CLDC 所对应的虚拟机也有所不同，CDC 对应的 Java 虚拟机是 CVM，而 CLDC 对应的虚拟机为 KVM。由于 CDC 比 CLDC 所包含的类库要多，相应地，CVM 比 KVM 需要支持的类库也多一些。KVM 是一个专门为移动设备设计的 Java 虚拟机，主要应用于 CLDC 配置。如图 3-3 所示，JVM 虚拟机主要用于 J2EE 和 J2SE，CVM 主要用于 J2ME 的 CDC 配置，KVM 主要应用于 J2ME 的 CLDC 配置。之所以称之为 KVM 主要因为 KVM 的内用容量是 KB 级别的，K 这里是 kilo 的意思，KVM 实现的最小内存空间为 128KB，包括虚拟机、最小运行库和执行程序所需要的空间。

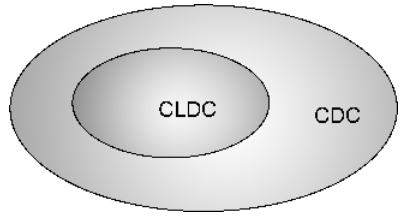


图 3-2 CDC 和 CLDC 关系图

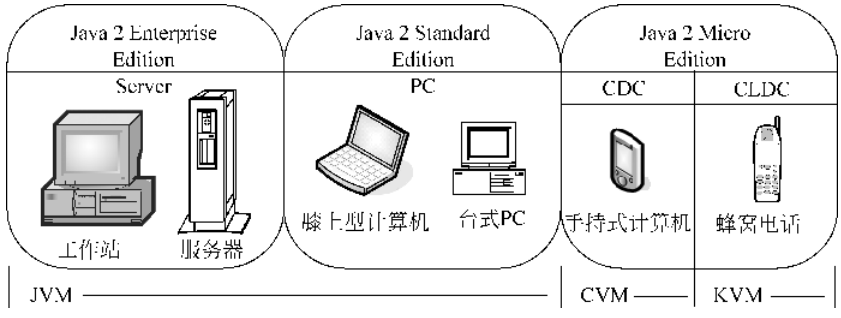


图 3-3 Java 虚拟机

2. 简表

简表(Profile)是配置的扩展,它定义了开发人员在编写针对特定的设备类型时可以使用的库,是一个完整的运行环境。一个完整的 JRE 是由配置和简表组成的,使用哪种 JVM 是由配置决定的,而操作硬件设备的 API 是由简表来决定的。目前使用最为广泛的是移动信息设备简表(Mobile Information Device Profile,MIDP),MIDP 扩展了 CLDC,它定义了用户界面组件、输入和事件处理、持久性存储以及联网和定时器的 API,这些都考虑到了在 CLDC 配置中移动设备的屏幕和内存的限制。

3. 规范

规范(Specification) 是为了让软件可以运行在多种不同的平台中而制定的一套标准。因为所有的 J2ME 配置和概要都是作为 Java 社区过程(Java Community Process,JCP)来发展的,所以,JCP 把业界的领跑者聚集在一起,目的是制定一个通用的规范,这样它们就可以用来设计产品。每一个配置或者概要都是作为 Java 规范要求(Java Specification Request,JSR)来发布的。JSR 定义了工作的范围和覆盖的领域。这个规范由一个专家组来创建,然后经过内部投票和修订再发布出来,经过公众的检验和最后的修订,最终的草案产生,JSR 也就完成了。已经完成的并定义目前的 J2ME 配置和概要的 JSR 规范如表 3-1 所示。

表 3-1 J2ME 规范

编号	规 范 名 称
213	Micro WSCI Framework for J2ME
214	Micro BPSS for J2ME Devices
180	SIP API for J2METM
190	Event Tracking API for J2ME
179	Location API for J2METM
172	J2METM Web Services Specification
177	Security and Trust Services API for J2METM
209	Advanced Graphics and User Interface Optional Package for the J2METM Platform
75	PDA Optional Packages for the J2METM Platform
37	Mobile Information Device Profile for the J2METM Platform
30	J2METM Connected, Limited Device Configuration
68	J2METM Platform Specification
184	Mobile 3D Graphics API for J2METM
226	Scalable 2D Vector Graphics API for J2METM
120	Wireless Messaging API
62	Personal Profile Specification
129	Personal Basis Profile Specification
197	Generic Connection Framework Optional Package for the J2SE Platform
229	Payment API
230	Data Sync API
246	Device Management API
118	Mobile Information Device Profile 2.0

续表

编号	规 范 名 称
139	Connected Limited Device Configuration 1.1
169	JDBC Optional Package for CDC/Foundation Profile
195	Information Module Profile
228	Information Module Profile-Next Generation(IMP-NG)
238	Mobile Internationalization API
242	Digital Set Top Box Profile-"On Ramp to OCAP"
257	Contactless Communication API
211	Content Handler API
218	Connected Device Configuration(CDC) 1.1
219	Foundation Profile 1.1
266	Mobile Sensor API
302	Safety Critical JavaTM Technology
66	RMI Optional Package Specification Version 1.0
135	Mobile Media API

3.1.3 有限连接设备配置

CLDC 提供了一套标准的、面对小型设备的 Java 应用开发平台。2000 年 5 月，Java Community Process(JCP)公布了 CLDC 1.0 规范(即 JSR30)。作为第一个面对小型设备的 Java 应用开发规范，CLDC 是由包括 Nokia、Motorola 和 Siemens 在内的 18 家全球知名公司共同协商完成的。CLDC 是 J2ME 核心配置中的一个，可以支持一个或多个 Profile。其目标主要是面向小型的、网络连接速度慢、能源有限(主要是电池供电)且资源有限的设备，如手机、机顶盒、PDA 等。CLDC 的核心是虚拟机和核心类库。虚拟机运行在目标操作系统之上，对下层的硬件提供必要的兼容和支持，核心类库提供操作系统所需的最小的软件需求。

1. CLDC 的目标

CLDC 的设计目标是为小型的、资源受限的连接设备定义一个 Java 平台标准，并且可以允许目标设备动态地传递 Java 应用和内容，从而使 Java 开发人员能够轻松地在这些设备上进行应用开发。

2. CLDC 的整体需求

设计需求是在 CLDC 上开发的能够运行在绝大多数的小型的、资源受限的连接设备上，并且尽可能地不使用设备的本地系统软件(做到与平台、设备无关)，定义能应用在绝大多数上述设备上的最小子集的规范，保证在不同类型上述设备之间代码级的可移植性和互操作性。

3. CLDC 的硬件需求

由于 CLDC 要面向尽可能多的设备，而这些设备所使用的硬件又各不相同。因此，

CLDC 规范中并没有指明需要某种硬件支持，只是对设备的最小内存进行了限制。CLDC 规范中要求硬件必须有 160KB 的固定内存以供虚拟机和 CLDC 核心类库使用，至少有 32KB 的动态内存以供虚拟机运行时使用(堆栈等)。

这里所说的固定内存是指拥有写保护，不会因关机而抹去的 ROM。对于具体设备的具体实现，这些需求也可能有变化。这里所规定的 160KB 是 CLDC 规范中的要求，实际上也可以是 128KB 左右。

4. CLDC 的软件需求

和硬件类似，CLDC 上运行的软件也是多种多样的。例如，有些设备支持多进程操作系统或者支持文件系统；而有些功能极其有限的设备并不需要文件系统。对于这些不确定性，CLDC 只定义了软件所必需的最小集合。CLDC 规范中要求操作系统不需要支持多进程或是分址空间寻址，也不用考虑运行时的协调和延迟，但是必须提供至少一个可控制的实体来运行虚拟机。

在 CLDC1.0 版本中定义了以下功能。

(1) Java 核心语言与 Java 虚拟机的特性。

(2) 核心 Java 类库。

(3) 输入/输出。

(4) 对网络的支持。

(5) 对安全性的支持。

(6) 对国际化的支持。

(7) CLDC 不包含的功能。

(8) 对应用程序生命周期的管理。

(9) 用户界面。

(10) 事件处理。

(11) 高级应用程序模式(这里指用户与应用程序的交互)。

由于 CLDC 主要针对 16 位、32 位主频在 16MHz 以上的处理器，设备内存只有 512KB 甚至更少，而目前 Windows 平台下运行的 JVM 需要的最小内存为 16MB。因此，CLDC 所使用的虚拟机和核心类库与 J2SE 的并不相同。CLDC 核心类库与 J2SE 的主要区别如下。

(1) 不支持浮点数据类型(没有 float 和 double)。因为很多使用 CLDC 的设备硬件都不支持浮点运算，而且处理浮点运算需要较大的内存。因此在 CLDC 1.0 中，并没有要求虚拟机支持浮点数据类型。

(2) 不支持 JNI(the Java Native Interface)。CLDC 不提供 native code 的支持，除了因为设备内存有限外，还出于安全性的考虑。因为 CLDC 中缺少完整的安全性模型，禁用了这些 J2SE 的特性可以使潜在的安全风险降到最低。

(3) 不支持以及用户自定义的 Java 级的类载入器(class loaders)。CLDC 不允许用户自定义类载入器。按照 CLDC 规范的要求，类的载入是不能被覆盖、替换和修改的。与 JNI 类似，这些是出于安全方面的一些考虑。

(4) 不支持反射(reflection)。不支持 java.lang.reflect 包以及 java.lang.Class 中和 reflection 有关的函数，其目的主要是节省内存占用。

(5) 不支持线程组(thread groups)或守护线程(daemon threads)。CLDC 提供了对线程和多线程的支持,但线程组和守护线程是不被允许的。每个线程都要生成独立的 Thread 对象来实现。如果应用程序想实现对一组线程的操作,则必须在应用程序的级别上自行实现多个 Thread 对象的控制,如使用 Hashtable 和 Vector 来存取多个 Thread 对象。

(6) 不支持类实例(class instance)的终结(finalization)。CLDC 类库不包含 java.lang.Object.finalize()方法,因此类对象的终结是不支持的。对于应用 CLDC 的设备来说,对象终结相对于它所起的作用来说实现起来过于复杂,并不被需要。

(7) 有限的错误处理(error handling)。在 J2SE 中定义了大量的类用来描述各种错误和异常,而 CLDC 仅仅包含有限的几个 J2SE 的核心类库,因此大部分 java.lang.Error 的子类都未支持,这包括异步异常。这是因为在嵌入式系统中,应用程序并不期望获得设备的出错处理机制;定义和运行出错处理需要较大的虚拟机的开销,而这些出错的代码信息对于连用户界面都没有的有限连接设备来说是没有用处的。

3.1.4 移动信息设备简表

MIDP(Mobile Information Device Profile,移动信息设备简表)为运行在 MIDP 容器中的 MIDlets 应用定义了一个 API,此 API 本身是建立在 CLDC API 之上的。

MIDP 用户接口 API 的 Java 类设计不是基于 Java Abstract Window Toolkit(AWT)类的,而是为移动电话和寻呼机等这类小型移动信息设备而特别设计的。这类设备只有有限的屏幕尺寸和键盘功能。当程序员采用 MIDP 编写图形界面时,它们只能使用 MIDP 或者 CLDC API。该组 API 的包名均以 javax.microedition 开头。MIDP 到目前为止发展到 2.0 版本,之前的版本为 1.0(MIDP 的第一个版本,也就是 MIDP 1.0)。在 MIDP 1.0 的实现中,要求必须支持如下几个包。

(1) javax.microedition.midlet——MIDlet 类包。

(2) javax.microedition.lcdui——界面类包。

(3) javax.microedition.rms——持久存储类包。

(4) 另外还有 javax.microedition.io 包中的一部分类。

在 MIDP 第二个版本,也就是 MIDP 2.0 实现中,除了兼容 MIDP 1.0 的实现以外,不仅扩充了很多已有类的功能,还增加了以下几个包。

(1) javax.microedition.lcdui.game——Game API,MIDP 2.0 游戏变成扩展。

(2) javax.microedition.media——多媒体类包。

(3) javax.microedition.media.control——多媒体控制类包。

(4) javax.microedition.pki——数字签名类包。

MIDP 用户接口的基本抽象图形是屏幕。Screen 类对面向设备的图形和用户交互进行了封装。每次应用智能显示一个屏幕,而且只能浏览或者使用屏幕上的条目。图 3-4 给出了一个基于屏幕的 MIDP 图形用户接口 GUI 的例子。

下面对 MIDP 2.0 针对移动设备的特性进行详细介绍。相对于 MIDP 1.0,在 MIDP 2.0 版本中提供了对多媒体的支持,更加丰富的网络协议支持,针对游戏的 UI 的支持,OTA 以及安全性等方面的支持。在 MIDP 2.0 中,Game API 是一个非常重要的 API,它是专门为游戏开发而设计的。在 MIDP 1.0 中,开发人员必须自己定义一套图像类来获得更好的界

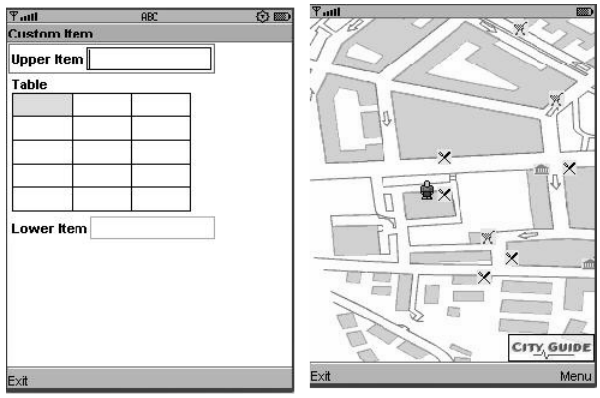
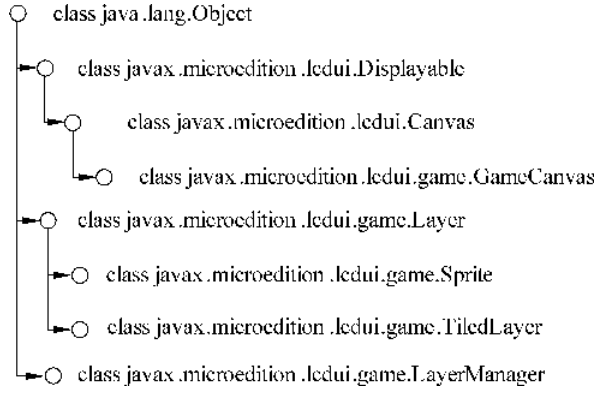


图 3-4 MIDP 应用界面

面和游戏性能。在 MIDP 2.0 中，新的 API 可以帮助开发人员完成这些工作，这个 API 的结构如下。



新的游戏 API 为游戏的开发提供专门的设计方法。在新的 API 中，游戏的界面由图层组成，背景在一个图层上面，而游戏人物在另一个图层上面，游戏 API 分别控制不同的图层。为了解决传统方法中控制游戏屏幕的滚动问题，新的 API 提出了观察窗口（view window）的概念，这个窗口是可以看到的游戏场景，通过移动观察窗口便可以容易地移动和定位。在上面的 API 结构中可以看到这个包中包含了 5 个新类，其中 GameCanvas 是提供了游戏基本接口的抽象类，扩充了基本 Canvas 的功能，增加屏幕缓冲和得到游戏键盘状态的功能。Layer 定义的是一个游戏的元素，其中 Sprite 和 TiledLayer 是它的子类。Sprite 是包含了若干帧图像的 Layer，所有的帧保存在 Image 对象中，并且可以通过其中的部分帧来创建一个动画。在游戏过程中，Sprite 还具有碰撞检测的功能，通过碰撞检测可以检测它是否和其他的 Sprite 或者 TiledLayer 有重合碰撞。TiledLayer 类似于 Sprite，但是 TileLayer 主要应用于创建背景。LayerManager 则负责管理所有的 Layer 对象并按照指定的顺序位置来绘制它们。

在 MIDP 2.0 中还增加了对声音的支持，以前包含在 Mobile Media API（MMAPI）中的一些类，现在已经包含在 MIDP 中，这使开发者开发程序时不一定要依赖于 MMAPI 包。对通信协议的支持也是 MIDP 2.0 的一大改进，例如对 HTTP 协议的支持，在 MIDP 1.0 中

仅支持 HTTP，而 MIDP 2.0 则增加了对 HTTPS 的支持，并且对于服务器 Push 体系的支持使手机可以收到来自服务器的消息广播。

3.1.5 MIDlet

一个建立在 CLDC 和 MIDP 之上的 Java 应用程序称为 MIDlet。一个 MIDlet 套件由打包在一个 JAR 中的一个或者多个 MIDlet 所构成。每个 MIDlet 应用都必须包含 3 个函数：startApp()、pauseApp()和 destroyApp()。因为这 3 个函数是在 MIDLet 状态转换中完成的，如图 3-5 所示，应用程序创建时进入的状态是 Paused(暂停)状态，启动时直接调用 startApp()函数，此时状态变成了 Active(活动状态)，应用的暂停会重新调用 pauseApp()。需要注意的是，startApp()函数不仅仅是程序启动的时候才会调用，恢复暂停的时候也会执行 startApp()函数。不论是暂停状态还是活动状态，当退出应用程序的时候会自动调用 destroyApp()函数，此时程序状态为 Destroyed，应用关闭。

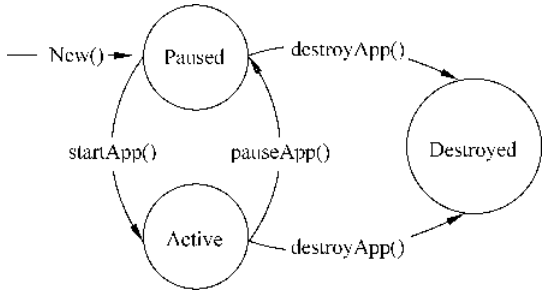


图 3-5 MIDlet 的状态图

一个 MIDlet 套件将包含 JAR 文件和 JAD 文件。在 JAR 文件中通常会包含一个清单文件(manifest.mf)，其属性描述如表 3-2 所示。清单文件是一个以 mf 为扩展名的纯文本文件，其中包含了对 JAR 文件的描述，文件中是以冒号分割的值对组，冒号前面是属性名，后面是对应的值。

表 3-2 清单文件属性值

属 性	描 述
必选属性	
MIDlet-Name	MIDlet 包的名字
MIDlet-Version	MIDlet 包的版本号
MIDlet-Vendor	应用的所有者/开发者
MIDlet-<n>	包中每一个 MIDlet 的名称、图标文件、类。例如： MIDlet-1——SuperGame,/supergame.png,com.your.SuperGame MIDlet-2——PowerGame,/powergame.png,com.your.PowerGame
MicroEdition-Profile	运行包中 MIDlet 需要的描述文件的名字。值要同系统属性 microedition.profile 完全相同。对于 MIDP 版本 1 使用 MIDP 1.0
MicroEdition-Configuration	运行包中 MIDlet 需要的配置的名字。使用系统属性 microedition.configuration 中包含的属性，如 CLDC 1.0

续表

属　　性	描　　述
可选属性	
MIDlet-Icon	作为标识该 MIDlet 包的图标的 PNG 图像文件的名称
MIDlet-Description	向潜在用户介绍这个包的描述性的文字
MIDlet-Info-URL	包中详细信息的 URL
MIDlet-Jar-URL	下载 JAR 的 URL
MIDlet-Jar-Size	以字节计算的 JAR 文件大小
MIDlet-Data-Size	MIDlet 需要的最小非易失性内存(持久存储),默认值为零

3.2 J2ME Wireless Toolkit

WTK 的全称是 Sun J2ME Wireless Toolkit——Sun 的无线开发工具包。这一工具包的设计目的是帮助开发人员简化 J2ME 的开发过程。使用其中的工具可以开发与 Java Technology for the Wireless Industry (JTWI,JSR 185) 规范兼容的设备上运行的 J2ME 应用程序。该工具箱包含了完整的生成工具、实用程序以及设备仿真器。到目前为止可以获取 4 个版本,分别是 1.0.4、2.0、2.1 和 2.2。每个版本都包括英语、日语、简体中文和繁体中文 4 个语种包。

3.2.1 建立 JDK 环境

进行 J2ME 编程第一步就是安装 JDK(Java Development Kit)。JDK 是 Sun Microsystems 针对 Java 开发的产品。自从 Java 推出以来,JDK 已经成为使用最广泛的 Java SDK(Software Development Kit)。

以在 Windows 环境下安装 JDK 为例,首先要下载安装文件,这里要下载的是 Sun 公司的 J2SE Development Kits,网址为 http://java.sun.com/javase/downloads/index.jsp,图 3-6 是 J2SE 5.0 下载页面,截止到目前为止,JDK 的最高版本为 1.6.0,这里假设使用 Window 操作系统作为操作平台,所以下载时选择 Windows Multi-language。

图 3-6　JDK 的下载界面

下载后是一个 jdk-6u13-windows-x64.exe 可执行文件，下载完毕，运行安装文件，根据安装向导进行配置安装。安装向导界面如图 3-7 所示，只需要单击“下一步”按钮即可。

图 3-7 JDK 安装界面

安装成功后需要配置系统的环境变量。右击“我的电脑”图标，在弹出的快捷菜单中选择“属性”命令，弹出“系统属性”对话框，打开“高级”选项卡，单击“环境变量”按钮，如图 3-8 所示。

在弹出的“环境变量”对话框中单击“新建”按钮，在弹出的“新建系统变量”对话框中的“变量名”文本框中输入“JAVA_HOME”，在“变量值”文本框中输入 JDK 安装路径“C:\ProgramFiles\Java\jdk1.6.0_13”，如图 3-9 所示。

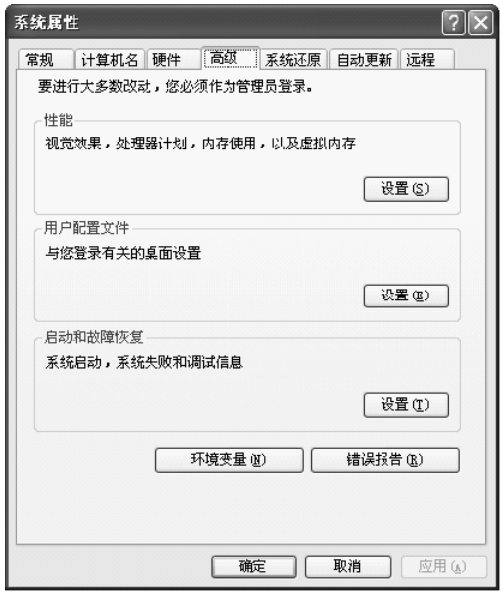


图 3-8 设置 JDK 环境变量

图 3-9 新建系统变量 JAVA_HOME

单击“确定”按钮返回“环境变量”对话框，在“系统变量”选项区域选中 Path 选项，单击“编辑”按钮，弹出“编辑系统变量”对话框，添加变量值“%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin”(其中“%JAVA_HOME%”指的是刚才设置的 JAVA_HOME 的值)，如

图 3-10 所示。

单击“确定”按钮返回“环境变量”对话框，单击“新建”按钮，在弹出的“新建系统变量”对话框中的“变量名”文本框中输入“CLASSPATH”，在“变量值”文本框中输入 JDK 的安装路径“.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar”，如图 3-11 所示。

图 3-10 添加系统变量 Path

配置完成后验证安装是否成功，选择“开始”→“运行”命令，在弹出的“运行”对话框中输入 cmd 命令，如图 3-12 所示。

图 3-11 新建系统变量 CLASSPATH

图 3-12 “运行”对话框

单击“确定”按钮，打开命令提示符窗口，输入命令 java-version，出现 JDK 版本信息，如图 3-13 所示，说明安装成功。

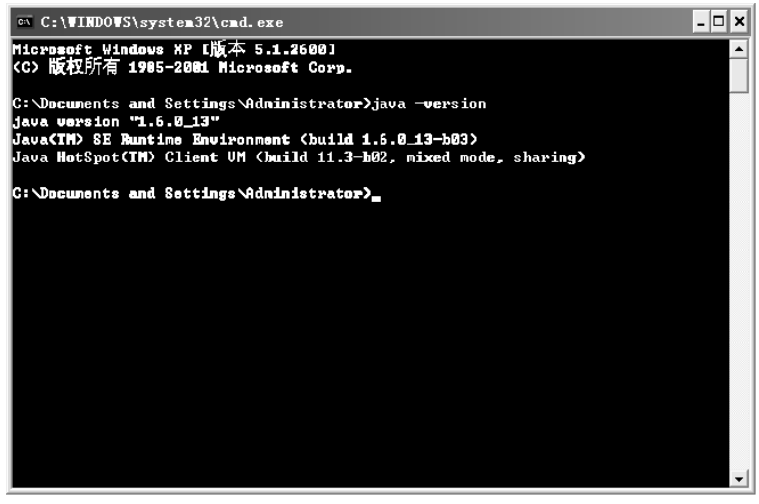


图 3-13 Windows 提示符