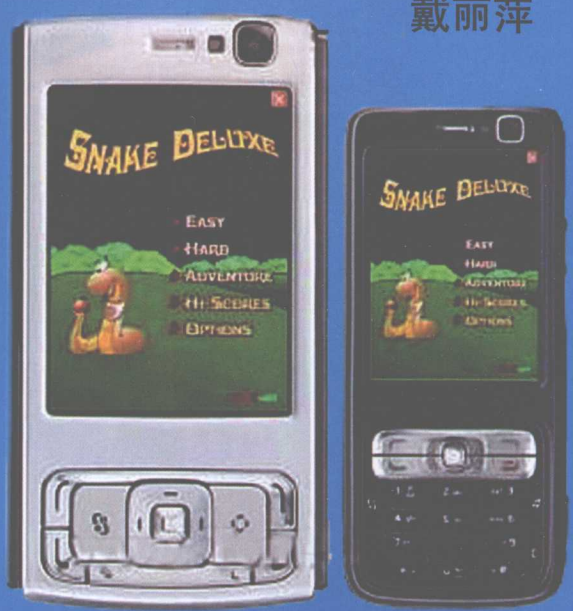
Java ME

手机游戏开发从入门到精通

戴丽萍　李 磊　许永辉　李振英　编著



国防工业出版社
National Defense Industry Press

Java ME 手机游戏开发从入门到精通

戴丽萍　李磊　许永辉　李振英　编著

国防工业出版社

·北京·

内容简介

本书定位于对制作游戏有极大热情，但编程能力不甚深厚的初学者。全书共5篇，以引导读者顺利学习。

第1篇（第1章至第4章），对Java ME进行了概述，指导读者正确地搭建开发平台，并完成了一个Java ME的小程序——“We are developer”。考虑到有些读者对于Java不太了解，所以单独划分了一章对Java进行了简单的介绍。

第2篇（第5章、第6章），介绍了键盘响应机制和Java ME自带的一些工具。到此，读者已经有能力进行简单Java ME的编写了。

第3篇（第7章至第12章），介绍了手机游戏的开发，当读者掌握了地图和精灵的编写技巧后，就可以编写自己的手机游戏了。第12章介绍了Java ME程序的简单优化，更复杂的优化请参阅相关书籍。

第4篇（第13章、第14章），对未来手机游戏的展望，随着手机与无线网络性能的提升，3D游戏与网络游戏的普及就在不远的将来。

第5篇（第15章、第16章），通过3个游戏引领读者制作实际的游戏，代码有大量的注释，读者可以参考。

图书在版编目（CIP）数据

Java ME手机游戏开发从入门到精通 / 戴丽萍等编著.
北京：国防工业出版社，2009.1
ISBN 978-7-118-05372-2

Ⅰ.J... Ⅱ.戴... Ⅲ.①JAVA语言—程序设计②移动通信—携带电话机—游戏—应用程序—程序设计 Ⅳ.TP312 G899

中国版本图书馆CIP数据核字（2007）第147387号

※

国防工业出版社出版发行
（北京市海淀区紫竹院南路23号 邮政编码100048）
北京诚信伟业印刷有限公司印刷
新华书店经售
*
开本 787×1092 1/16 印张 21¼ 字数 489千字
2009年1月第1版第1次印刷 印数 1—4000册 定价 39.00元

（本书如有印装错误，我社负责调换）

国防书店：（010）68428422 发行邮购：（010）68414474
发行传真：（010）68411535 发行业务：（010）68472764

前　言

随着时代进步和科技的飞速发展。移动电话从20年前的大哥大变成了今天不可缺少的办公用具，不仅管理人们的交际网络，还应用于人们日常生活的安排。

移动电话中的软件数量正在飞速增长，最近一段时间，甚至超出了摩尔定律：在过去的3年里，高端电话中的嵌入式软件大小已经从2MB跳增至20MB。

作为IT行业的新兴产业，手机游戏已经成为许多人生活中不可缺少的娱乐方式。手机游戏的兴起在中国始于20世纪90年代，虽然有十几年了，可是受到SUN JSR和手机内存、运算速度等多方面限制，可以说2004年之后这个行业的黄金时期才真正开始。对于现在的手机而言，平台比较多，各种平台的本地语言的编写格式的不同导致游戏针对不同的平台开发移植存在问题，而如果采用无关语言的Java ME平台进行开发，可以做到开发成本低、周期短，并且Java ME封装的接口对于开发游戏来说足够丰富和好用。

游戏的开发包括几个必须解决的问题：第一是游戏的构架设计，框架设计涉及到游戏的流畅性、游戏性，主要是大的方面；第二是游戏的算法，不同的游戏会用到不同的算法，如角色扮演游戏、益智类游戏、冒险类游戏，都有不同的算法作为基础；第三是Java语言的一些特性，很多与语言有关的问题，可以直接影响游戏的效率、流畅性。一般来说，如果能成功地解决以上三个问题，游戏的开发入门阶段就可以顺利的度过了。

未来几年，手机游戏可能和电脑游戏在开发方面存在将近10年的差距，也就是说，现在的手机游戏还只停留在10年前的PC游戏质量上，2D游戏将作为游戏开发的主流存在，然而手机发展的速度在这几年出奇的快，3D的支持也开始逐步增加。可以预见不久的将来，3D的开发也将进入人们的视野。在3D游戏中，许多令人耳目一新的创意将在手机中实现。

手机是一个新生的事物，手机游戏也将逐步成为生活的一部分，必将拥有一个美好的前景，无论游戏的设计者还是消费者，目标都是让游戏更加绚丽和好玩。

读者可以发邮件给作者（lp_may@ hotmail. com）免费索取本书实例源代码。

作　者

2007年8月

目　录

第1篇　基础篇

第1章　传说中的技术——Java ME …… 1

1.1　Java 平台的划分——Java SE, Java EE, Java ME …… 1

1.1.1　Java 的由来 …… 2

1.1.2　Java SE 的企业级扩充——Java EE …… 5

1.1.3　Java SE 向移动设备的发展——Java ME …… 5

1.1.4　JCP 与 JSR …… 6

1.2　Java ME 平台体系结构 …… 6

1.3　CLDC …… 7

1.3.1　Configuration …… 7

1.3.2　CLDC1.0 与 CLDC1.1 …… 8

1.4　MIDP …… 10

1.4.1　Profile …… 10

1.4.2　MIDP1.0 与 MIDP2.0 …… 11

小结 …… 13

第2章　Java ME 开发准备 …… 14

2.1　Java 虚拟机与 JDK …… 14

2.1.1　Java 虚拟机 …… 14

2.1.2　JDK …… 15

2.2　适合新手的开发环境 …… 16

2.2.1　JDK 与 WTK 的安装、配置与使用 …… 16

2.2.2　方便的文本编辑工具——UltraEdit32 …… 30

小结 …… 33

第3章　Java 核心语法与面向对象基础 …… 34

3.1　Java 的数据类型 …… 34

3.1.1　整型数据 …… 35

3.1.2　字符型数据 char …… 35

3.1.3　浮点型数据 float …… 36

3.1.4　布尔型数据 Boolean …… 36

3.1.5 字符串数据 String …… 36
3.1.6 数组数据 …… 36
3.2 Java 关键字与控制结构 …… 37
3.2.1 关键字 …… 37
3.2.2 控制结构 …… 37
3.3 类与对象 …… 40
3.4 继承 …… 41
3.5 接口与抽象类 …… 42
3.5.1 接口 …… 42
3.5.2 抽象类 …… 43
3.6 异常与 I/O 流 …… 43
3.6.1 异常 …… 43
3.6.2 I/O 流 …… 46
小结 …… 46

第 4 章 第一个 Java ME 程序——"We are developer" …… 47

4.1 运行第一个 Java ME 程序——"We are developer" …… 47
4.1.1 建立项目 …… 47
4.1.2 输入代码 …… 48
4.1.3 运行程序 …… 50
4.2 MIDlet 的生命周期 …… 51
4.2.1 MIDlet 程序的三种状态 …… 51
4.2.2 MIDlet 程序生命周期方法 …… 51
4.2.3 三种状态的转变方法 …… 52
4.3 JAD 与 JAR …… 53
4.3.1 JAR 概述 …… 53
4.3.2 JAD 概述 …… 54
小结 …… 55

第 2 篇 入门篇

第 5 章 游戏操作的实现——键盘响应 …… 56

5.1 键盘响应 …… 56
5.1.1 键盘码的定义与获取 …… 56
5.1.2 keyPressed——按键按下 …… 59
5.1.3 keyRleased——按键松开 …… 61
5.1.4 keyRepeated——重复按键 …… 63
5.2 Command 的使用 …… 64
5.2.1 Command 类 …… 64

5.2.2 Command 类型 …… 64
5.2.3 通用事件处理——CommandListener …… 64
5.3 指针实例讲解 …… 65
小结 …… 78

第 6 章 方便的工具——系统工具的调用 …… 79

6.1 获取系统参数 …… 79
6.2 执行时间的测量 …… 80
6.3 随机数 …… 81
6.4 Collection 类的使用 …… 82
6.5 线程的使用 …… 84
6.6 Timer 与 TimerTask 的使用 …… 87
6.7 综合实例——色子程序的讲解 …… 89
小结 …… 97

第 3 篇 功能篇

第 7 章 MIDP 低级用户界面——Canvas …… 98

7.1 低级用户界面开发简介 …… 98
7.1.1 Canvas …… 98
7.1.2 Font …… 99
7.1.3 Image …… 99
7.1.4 Graphics …… 99
7.2 Canvas 的开发 …… 99
7.2.1 屏幕坐标的指定 …… 99
7.2.2 字体的设定与字符串的绘制 …… 102
7.2.3 图片的绘制 …… 105
7.2.4 让人物走起来 …… 107
7.2.5 图片的裁减 …… 110
7.3 屏幕抖动的处理——双缓存技术 …… 113
7.4 游戏的基本框架 …… 117
小结 …… 119

第 8 章 游戏的背景——贴图 …… 120

8.1 地图简介 …… 120
8.2 地图数组的生成——MAPPY 的使用 …… 121
8.3 贴砖的方法与技巧 …… 125
8.3.1 读取地图文件与生成地图数组 …… 125

8.3.2 贴砖方法1 …… 126
8.3.3 贴砖方法2 …… 132
8.4 多层地图的实现 …… 137
小结 …… 138

第9章 精灵与障碍物的碰撞检测 …… 139

9.1 碰撞检测介绍 …… 139
9.2 碰撞检测方法介绍 …… 139
9.2.1 点与矩形的碰撞检测 …… 139
9.2.2 矩形与矩形的碰撞检测 …… 143
9.2.3 圆与圆的碰撞检测 …… 148
9.2.4 矩形与砖块的碰撞 …… 152
小结 …… 157

第10章 聪明的敌人——AI的方法 …… 158

10.1 AI的基础知识 …… 158
10.2 AI的基本类型 …… 158
10.2.1 漫游AI …… 158
10.2.2 行为AI …… 166
10.2.3 策略AI …… 171
10.3 追踪AI范例——解决“卡怪”问题 …… 172
小结 …… 180

第11章 最高分的存储:记录管理系统——RMS …… 181

11.1 数据持久存储开发简介 …… 181
11.2 记录文件的创建与删除 …… 182
11.2.1 记录文件的创建 …… 182
11.2.2 记录文件的关闭与删除 …… 184
11.3 存储记录的添加与读取 …… 186
11.4 数据类型与字节数组的转换技巧 …… 187
11.5 存储记录的更新与监听 …… 188
11.6 RMS的高级接口 …… 195
11.6.1 RecordEnumeration(遍历接口) …… 195
11.6.2 RecordFilter(过滤接口) …… 196
11.6.3 RecordComparator(比较接口) …… 197
11.7 综合实例Test2 …… 197
小结 …… 203

第 12 章　手机游戏的限制与性能优化 …… 204

12.1　手机游戏的限制 …… 204
12.1.1　内存 …… 204
12.1.2　类库文件 …… 205
12.1.3　屏幕大小和按键 …… 205
12.2　性能的优化 …… 206
12.2.1　运行的速度 …… 206
12.2.2　内存的合理利用 …… 208
12.2.3　JAR 文件 …… 209
小结 …… 211

第 4 篇　展望篇

第 13 章　令人兴奋的 3D 游戏 …… 212

13.1　3D 坐标系介绍 …… 212
13.2　照相机(camera) …… 212
13.3　光线(light) …… 213
13.4　mesh 的建立方法 …… 213
13.4.1　mesh 简介 …… 213
13.4.2　VertexBuffer …… 214
13.4.3　IndexBuffer …… 214
13.4.4　mesh 的建立 …… 215
13.4.5　Appearance 的设定 …… 215
13.5　保留模式建模 …… 219
小结 …… 219

第 14 章　无线网络的应用 …… 220

14.1　无线网络的概述与前景 …… 220
14.1.1　无线网络概述 …… 220
14.1.2　无线网络前景 …… 221
14.2　初识通用连接框架 …… 221
14.2.1　GCF 的面貌 …… 221
14.2.2　GCF 的使用 …… 221
14.3　服务器 Tomact 的安装与配置 …… 222
14.4　HTTP 协议的无线程序 …… 226
14.4.1　HTTP 协议简介 …… 226
14.4.2　HTTP 协议无线网络程序的过程概述 …… 227

14.4.3 实例:使用 HTTP 协议读取服务器 HTML 页信息 …… 229
14.4.4 实例:客户机与服务器(Servelet)交互信息 …… 233
小结 …… 240

第 5 篇 实例篇

第 15 章 拼图游戏与贪吃蛇游戏 …… 241

15.1 拼图游戏 …… 241
15.1.1 拼图游戏的整体设计 …… 242
15.1.2 图片块的设计 …… 244
15.1.3 Options 菜单项功能的分析 …… 245
15.1.4 Board. java 类文件的分析 …… 248
15.2 贪吃蛇游戏 …… 258
15.2.1 游戏的整体设计 …… 259
15.2.2 食物类(WormFood)分析 …… 260
15.2.3 WormLink 类分析 …… 262
15.2.4 蛇体类 Worm 的分析 …… 266
15.2.5 游戏的核心 WormPit 类分析 …… 273
15.2.6 计分系统 WormScore 类分析 …… 287
小结 …… 290

第 16 章 游戏开发实例 …… 291

16.1 整体游戏设计 …… 291
16.2 游戏整体架构建立 …… 292
16.3 piece 类的构建 …… 292
16.4 gameCanvas 类的构建 …… 294
小结 …… 329

参考文献 …… 330

第 1 篇　基 础 篇

第 1 章　传说中的技术——Java ME

本章主要内容

本章介绍了 Java 的历史及发展现状，现阶段 Java 主流的 3 个平台——Java SE，Java EE 及 Java ME，之后介绍了 Java ME 平台体系结构、CLDC 及 MIDP 的定义。

1.1　Java 平台的划分——Java SE，Java EE，Java ME

现阶段 Java 分为 3 个主流平台（图 1-1），即 Java SE，Java EE 及 Java ME。Java SE（Java Standard Edition）即 Java 标准版，是以界面程序、Java 小程序和其他一些经典的应用程序为目标的。几年前 Sun 用 Java EE（Java Enterprise Edition）扩展了 Java 开发包，Java EE 是用在服务器端开发的。这个版本为数据库访问、消息管理、进程间通信和事务处理增加了一些新的工具。最后是 Java ME（Java Micro Edition），Java 平台微缩版，也是本书所要阐述的重点。Java ME 为运行在嵌入式消费类电子产品的设备（如移动电话、PDA、游戏终端之上的应用程序）提供了一个健壮的、灵活的环境。

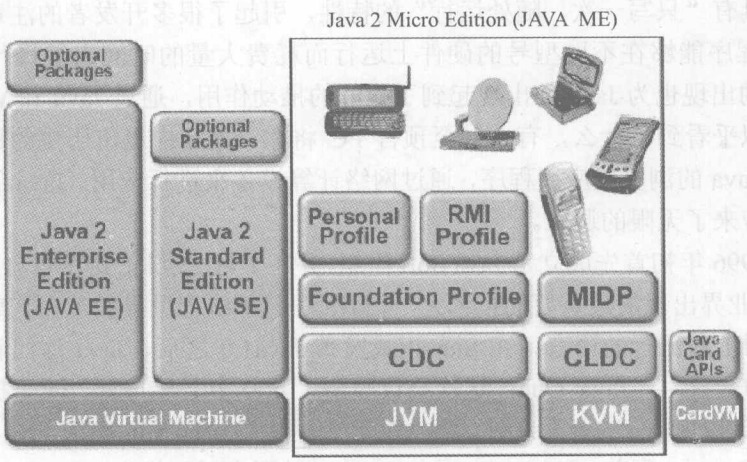


图 1-1　现阶段 Java 3 个主流平台

1.1.1 Java 的由来

Java 最早可以追溯到 20 世纪 90 年代早期。1990 年 12 月，Sun 的工程师 Patrick Naughton 被当时糟糕的 Sun C++工具折磨得快疯了。他大声抱怨，并威胁要离开 Sun 转投当时在 Steve Jobs 领导之下的 NeXT 公司。领导层为了留住他，给他一次机会，启动了一个叫做 Stealth（秘密行动）的项目。随着 James Gosling 等人的加入，这个项目更名为 Green。其目标是使用 C++为嵌入式设备开发一种新的基础平台技术，James Gosling 本人负责开发一个 SGML 编辑器。正如人们事后分析的那样，这位天才的程序员太懒惰——所以没有把 C++学好，开发中碰了一头包；太急躁——所以不愿意停下来读读 Scott Meyers 的新书 *Effective C++*；太傲慢——所以轻易地决定开发一种新的编程语言。他把这种语言命名为 C++++--，意思是 C++“加上一些好东西，减去一些坏东西”。显然这个糟糕的名字不可能长命百岁，很快这种颇受同伴喜爱的小语言被命名为 Oak。

1992 年 9 月，Oak 语言连同 Green OS 和一些应用程序一起发布在称作 Start 7 的小设备上，从而使其有了第一次精彩的亮相。随后，Sun 开了一家名为 First Person 的公司，整个团队被转移到这家公司里研发机顶盒，以投标时代华纳公司的一个项目。这帮天才被技术狂热所鼓舞，开发出了一个高交互性的设备，结果没想到时代华纳公司和有线电视服务商并不愿意用户拥有那么大的控制权，从而在竞标之战中败给了 SGI。Oak 的锋芒之锐，竟然把客户都给吓懵了。Sun 沮丧地关闭了 First Person，召回了整个团队。事实证明，传统行业中那些脑满肠肥的保守主义者是腐朽没落的。回到激情澎湃的 IT 产业，抓住互联网的大潮才是出路。

1994 年，Oak 被命名为 Java，针对互联网的新一轮开发如火如荼地展开。

1995 年，Sun 正式对外公布了 Java，并且发布了 JDK1.0。这种外形酷似 C++，却包含一颗 Smalltalk 般纯洁的面向对象之心的全新程序设计语言及其平台，几乎在一夜之间就成为软件产业的新宠儿。Java 当时仅仅被用来为网站制作一些动态应用，诸如动画图片之类，但这仍然引起了很多 Web 开发者们的注意，他们非常渴望有一种安全的语言，可以在静态的 HTML 网页上制作动画图片。Sun 最终把 Java 集成到 Net Scape 浏览器。同时因为它具有“只写一次、随处运行”的特性，引起了很多开发者的注意，他们可以不用为了使程序能够在不同型号的硬件上运行而耗费大量的时间来编译代码。当时的 Web 浏览器的出现也为 Java 的出现起到了很好的推动作用，通过 Java 和 Web 浏览器的结合，人们似乎看到了什么，有人甚至预言 PC 将在一两年内退出历史的舞台，取而代之的是基于 Java 的浏览器应用程序，通过网络计算设备来进行应用。Java 的出现为当时的软件产业带来了无限的遐想。

Sun 在 1996 年初首先成立了 Java Soft 组织，并在 1 月 23 日正式发布自己的 Java1.0，作为 20 世纪业界出现的最重要的技术之一，Java 引起了编程世界的革命。直到现在，Java 仍然是互联网上最流行的语言。在 Sun 正式发布 Java1.0 之后，Java 这门新生的语言就拥有了自己的会议——Java One，这次会议吸引了 600 多名参与者。除了拥有这么多的积极参与者来进行 Java 的开发之外，各大知名公司也纷纷向 Sun 申请 Java 的使用许可。一时间，Net Scape、惠普、IBM、Oracle、Sybase 甚至当时刚推出 Windows 95 的微软都是 Java 的追随者。Java 的应用就像是世界上的顶级玩家们组成的一个公开联盟，告诉全

世界大家都在用着 Java。也正是因为如此，Java 也找到了自己的归宿。现在的 Java EE 已经成为中大型企业级应用的标准，成为承接数据库和 Web 之间的一个重要桥梁。当年 Java 的机会实在太多，以至于很难选择到底该做什么。最终 Java 在应用服务器市场获得了难以取代的地位，也确定了 Java EE 的发展方向，并且仍将延续下去。

1995 年，正是微软在软件产业地位达到巅峰的时代，Windows 95 发布时的风光场面给人们留下的深刻印象至今难忘。尽管如此，作为最卓越的技术领袖，比尔·盖茨仍然敏锐地注意到 Java 的锋芒。当他了解了 Java 的一些细节之后，给予了这样的评价："Java 是很长时间以来最优秀的程序设计语言。"基于此，微软于 1996 年 3 月申请并获得了 Java 的使用许可。微软对 Java 的这一热情态度在当时大大提高了人们对 Java 的兴趣和信心，但也有不少人担心微软会依靠自己强大的影响力在标准之外另立标准，从而破坏 Java 的纯洁性。果然，从 1997 年发布 Visual J++的第一个版本开始，微软就开始在 Java 中掺入自己的私有扩展，这引起 Sun 的高度重视。

1997 年 10 月，Sun 向美国加州地方法院起诉微软违反两公司就微软使用 Java 技术所签定的合同，指控微软在自己的 Java 产品中做了"不恰当的修改"，违反了合同中承诺向用户提供 Java 兼容产品的条款。这一官司旷日持久，直到 2001 年 1 月双方才达成和解，微软将继续提供采用 Sun 开发的 Java 技术的现有产品（包括测试版）。不过，Sun 有限制地仅对包括 Java1.1.4 的微软产品提供许可。2001 年 7 月，微软公布新版的 Windows XP 将不再支持 Sun 的 JVM，并且推出了.NET 平台与 Java 分庭抗礼。当时的这一场官司对 Java 世界产生了深远的影响。如果没有这一场官司，也许很多 Java 程序员都在使用 Visual J++，基于 WFC 开发 Windows 客户端程序，同时不得不面对被两个不同的事实标准所分裂的 Java 世界。

1998 年，Java 2 平台正式发布。经过了 3 年的发展、热热闹闹的攻关宣传、红红火火的众厂商的热情参与，Sun 终于知道 Java 适合干什么了。对比 Java 刚发明时的技术定位，与 Java 戏剧性触"网"的那段历史，Java 2 平台的发布可真算得上是有的放矢了。根据官方的文档，Java 2 是 Sun 意识到"one size doesn't fit all"之后，把最初的 Java 技术打包成 3 个版本的产物，也就是著名的 Java ME、Java SE、Java EE。之所以说 Java 自从 Java 2 平台发布之后，进入了现代。那是因为之前的历史怎么看都和现在程序员日常开发使用的技术没有什么关系，如 Applet，已经很少有人使用。Java 2 之后的历史就不一样了，至少人们在推崇轻量级开发，猛批 EJB 时还不时会引用 Java EE 这个词是如何诞生的。Java 2 的 3 个版本中，除了 Java EE 得到了长足发展和广泛使用之外，Java ME 也在手机市场上取得了遍地开花的结果。相较之下，Java SE 难免落寞，只剩 SWT 这个血统不纯的家伙在 Rich Client 回归的时代吸引着人们的眼球。无论今天看来当时的 Java 2 有多么的不成熟，至少经过市场和时间的检验，Java 2 规划出来的 3 个方向把 Java 技术指向了光明的方向是毋庸置疑的。JCP 组织成立，并开始把握 Java 的发展方向。JCP 组织的开放性，不但使得所有对 Java 感兴趣的商业公司可以参与 Java 的发展，更重要的是 JCP 允许个人、非盈利组织、学校等加入，这就给 Java 带来了巨大的活力。随之兴起的 Java 开源运动的最大贡献是实现和鼓励了知识共享，在众多热情的开源程序员们的努力和分享下，很多原先只被商业公司掌握的技术、思想和产品可以被所有需要的开发人员免费或者以较低的价格获得使用权,并通过开放源代码更容

易地获得反馈和改进意见从而进一步演化发展。众所周知，知识不是孤立发展的认知，而人们的经验、认识是思考交流和积累的产物。开源运动所带来的开放、反馈、交流的风气正是符合人类社会知识形成和发展规律的，而开源运动起源于西方的发达国家，有其现实背景和文化根源。

Java 语言的出现使得互联网络有了良好的交互性能，但这些很“酷”的技术仅被人们认为是一些小花招，它还无法消除企业级用户对它的怀疑。1998 年，BEA 公司宣布收购 WebLogic 公司，并接着推出由 Sun 公司第一个授权使用 Java EE 许可证的 WebLogic Server 应用服务器，这个 Java 版的 AppServer 一推出就引起业界极大的兴趣。WebLoigc Server 以其对标准的支持、强悍的运算能力和安全的架构设计等特性很快征服了那些怀疑 Java EE 应用的人们。推出市场后不到一年，WebLogic Server 就成为业内第一 Java 应用服务器。这里援引一些当时著名咨询公司的调查数据来说明问题。在 IDC 的报告中，BEA 公司在应用服务器和交易服务器领域市场份额第一；在 Gartner 的报告中，BEA WebLogic Server 拥有业内最广泛的 EJB 应用安装基础；在 Giga Group 的报告中，BEA WebLogic Server 市场份额占 32%。因为应用服务器市场极大的发展潜力，在 WebLogic Server 之后，其他的很多公司也推出了自己的 AppServer，如 IBM 的 WebSphere、Sun 公司的 iPlanet 等，逐渐地应用服务器取代了传统意义上的各类中间件，成为企业应用的基础平台。应用服务器的出现使得 Java 有了真正意义上的发展。

1999 年，JDK1.3 发布，带来了一个重要的新特性：动态代理（Dynamic Proxy）。当所有人都对这项新技术的用途感到迷惑时，Oberg 发现用它便可以轻松攻克 EJB 容器实现中的一些难关。这一发现的产物就是一本书：Mastering RMI，以及大名鼎鼎的 JBoss 应用服务器。但 Oberg 很快又让世人见识了他的玩世不恭。由于和总经理 Marc Fleury 在经营理念上不合，Oberg 抱怨“法国的天空总让我感到压抑”，甩手离开了自己一手打造的 JBoss。此后的几年里，他和老友 Hani Suleiman 不断地对 JBoss 的“专业开源”模式和 Marc Fleury 的商人味道冷嘲热讽，让众人为他的孩子气扼腕叹息。

2003 年 4 月 2 日，Sun 与微软达成 16 亿美元的法律和解。如果不是晚了一天，许多人会以为这是一个在 4 月 1 日愚人节开的玩笑。尽管当时所有人都像是看到“太阳从西边出来了”那样张大了嘴巴，但这的确是事实。根据两家公司达成的版权协议，双方会为采用对方的技术而支付专利费用，微软向 Sun 提前支付 3.5 亿美元使用费，Sun 则承诺，如果 Sun 集成微软的某些技术，也会向微软付款。毫无疑问，“私下了结”的方式对双方而言都是最好的结果。就在协议签署的当天，在美国旧金山由 Sun 和微软为“抛弃十年恩怨、携手合作”举行的新闻发布会上，尽管比尔·盖茨没有到场，但这并没有妨碍现场看起来异常轻松的气氛。麦克尼利和鲍尔默各自穿了一件密歇根州底特律“Red Wings”曲棍球队的运动服，并谈及了一起在哈佛大学读书的经历，麦克尼利还说：“当时我们两人是非常要好的朋友，当然我们也有吵架的时候。”人与人当然可能成为终生的知己，但是公司与公司之间有的只能是利益上的分分合合。

微软跟 Java 不和，世人皆知。跟 Sun 和解了又怎么样？.NET 跟 Java 就是竞争对手。但是有点 IT 常识的人都知道，微软并非一开始就跟 Java 过不去。当年比尔·盖茨盛赞 Java 是“长期以来最好的程序设计语言”，而且很早就购买了 Java 使用许可。但是微软作为 IT 行业的老大，看着其他公司的迅速发展，不由得生了私心杂念，搞起了小动作，

在 Visual J++中加入了一些破坏纯洁性的东西。单独来看，Visual J++是 COM 时代微软最棒的开发工具，用 WFC 写 Windows 应用程序和 COM 组件实在是一种享受。但是放在 Java 大家庭里，就显得多少有点不怀好意。一场官司下来，微软被逐出 Java 大家庭，Visual J++无疾而终。以后的事情尽人皆知，.NET 出笼，利齿直指 Java，经过几年的激烈竞争，二分天下有其一。设想如果当时微软能够摒弃帝国主义心态，正确对待 Java，与其他人一起共建美好的 Java“共产主义社会”，那么今天的软件开发世界应该会美好得多。2004 年，微软与 Sun 实现了和解，但愿到 Java 20 周年的时候，能更正面地描述微软对 Java 发挥的作用。

1.1.2 Java SE 的企业级扩充——Java EE

电子商务和信息技术的快速发展以及对它们的需求给应用程序开发人员带来了新的压力。必须以比以往更少的金钱、更少的资源来更快地设计、开发企业应用程序。为了降低成本，并加快企业应用程序的设计和开发，Java EE 平台提供了一个基于组件的方法，来设计、开发、装配及部署企业应用程序。Java EE 平台提供了多层的分布式应用模型、组件再用、一致化的安全模型以及灵活的事务控制。不仅可以用比以前更快的速度向市场推出创造性的客户解决方案，而且平台独立的、基于组件的 Java EE 解决方案不会被束缚在任何一个厂商的产品和 API 上。

Java EE 规范定义了以下种类的组件。

（1）应用客户组件。

（2）Enterprise JavaBeans 组件。

（3）Servlet 及 JavaServer Pages（JSP 页面）组件（也被称作 Web 组件）。

（4）Applet。

一个多层分布式的应用模型意味着应用逻辑根据功能被划分成组件，并且可以在同一个服务器或不同的服务器上安装组成 Java EE 应用的这些不同的组件。一个应用组件应被安装在什么地方，取决于该应用组件属于该多层 Java EE 环境中的哪一层。这些层是客户层、Web 层、业务层及企业信息系统层（EIS）等。

1.1.3 Java SE 向移动设备的发展——Java ME

Java ME 也就是 Java Micro Edition，最早在 1999 年 6 月的 Java One 大会上被正式提出，是 Sun 专门为小型的、资源受限的、消费型电子设备的应用程序开发所提供的新的 Java 版本，它广泛地应用于（如蜂窝电话 Cell Phone 双向传唤机、Two-way Pager PDA 个人数字助理以及电视机顶盒等）众多小型资源受限设备，相对于通常在 Desktop 上使用的 J2SE 可以粗略地把 Java ME 理解为在微型设备上使用的 Java 平台。

Java ME 这个名词虽然是最近才提出的，但是自从 Java 技术提出以来 Sun 就从未停止过在小型设备应用这方面的研究。从最早的 20 世纪 90 年代初的 Oak 计划（Java 的前身）到 1997 年提出的 Personal Java 1998 的 Java Card，然后是 1999 年的 KVM 计划以及 Java ME 这个名词的提出，Java ME 作为一个独立的平台逐渐成型而且还在不断地发展更新中。目前 Java ME 的体系结构将在后面的章节中介绍。

1.1.4 JCP 与 JSR

JCP（Java Community Process）是一个开放的国际组织，主要由 Java 开发者以及被授权者组成，职能是发展和更新 Java 技术规范、参考实现（RI）、技术兼容包（TCK）。Java 技术和 JCP 两者的原创者都是 Sun 计算机公司。然而，JCP 已经由 Sun 于 1995 年创造 Java 的非正式过程，演变到如今有数百名来自世界各地 Java 代表成员一同监督 Java 发展的正式程序。

JCP 维护的规范包括 Java ME、Java SE、Java EE、XML、OSS、JAIN 等。组织成员可以提交 JSR（Java Specification Requests），通过特定程序以后，进入到下一版本的规范里面。

JSR 是早期提议和最终发布的 Java 平台规范的具体描述。通常，一个新的 JSR 的提出是为了增加或者规范 Java 平台的功能。某个具体的 JSR 由专家组共同来制定，工作由组长协调。例如，CLDC1.0（Connected Limited Device Configuration，JSR30）由 Sun 公司的 Antero Taivalsaari 担任组长，同时专家组的成员还包括 Siemens、Motorola、Nokia、Symbian 等。Java ME 平台规范是在 JSR68 中定义的，规范组长是 Sun 公司的 Jon Courtney。

JSR 完成后，相关的规范及 Java API 会在 JCP 的官方网站（http://jcp.org）发布。设备制造商可以在自己的产品中实现某个 JSR，如 MIDP2.0（JSR118）。但是这些都必须要通过 TCK（Technology Compatibility Kit）测试以确保技术兼容性。

1.2 Java ME 平台体系结构

Java ME 平台由多种配置（Configuration）、简表（Profile）和可选包（Optional Package）组成。平台的实现者和应用程序的开发者可以从中选择并组合出一个完整的 Java 运行环境来满足特定范围内的设备需求。每种组合都应该使这一系列设备的内存、处理器和 I/O 能力达到最优化。Java ME 专家组之所以采取这种灵活的设计结构，主要是为了满足市场上不同种类的嵌入式设备的需求，这些设备在软件和硬件特性上都存在巨大的差异，一种规范很难将它们统一起来。Java ME 平台的体系结构，如图 1-2 所示。

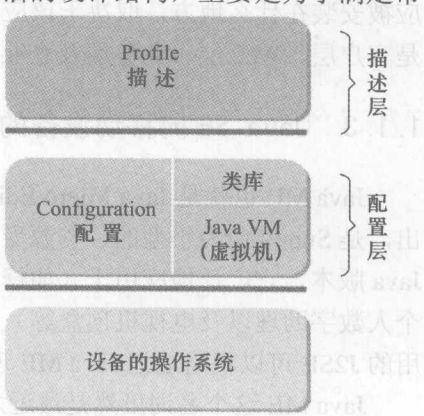


图 1-2 Java ME 平台体系结构

设备操作系统位于 Java ME 运行环境的最底层，操作系统可以是 Linux、Symbian 或者 PalmOS，这充分体现了 Java 语言跨平台的特性。配置由 Java 虚拟机和一系列的 API 集合组成，为某一范围内的嵌入式设备提供基本的功能，这些设备通常在硬件和软件上具有类似的特性。目前，Java ME 平台主要包括两个配置：连接设备配置（Connected Device Configuration，CDC）和连接受限设备配置（Connected Limited Device Configuration，CLDC）。为了给目标设备提供完整的运行环境，配置必须和简表组合。简表位于配置之上，为运行环境提供高层的 API，如应用程序模型和图形用户界面等。目前，CLDC 上

应用最广泛的简表是移动信息设备简表（Mobile Information Device Profile，MIDP），这也是本书介绍的内容。基于 CLDC 上 MIDP 的 Java ME 平台主要面对的目标设备是移动电话。Java ME 平台可以通过添加可选包进行扩展，可选包是针对特殊技术的实现，因此它的定位是特定范围的设备，而不适合作为一项特性定义在 MIDP 中。比较常见的可选包有无线消息 API（Wireless Messaging API，JSR120）、移动多媒体 API（Mobile Media API，JSR135）和 Web 服务 API（Web Service API，JSR172）。随着移动设备内存的增大和处理能力的提高，越来越多的可选包被添加到具体的 Java ME 平台上，这是一件让开发者非常兴奋的事情。

1.3 CLDC

1.3.1 Configuration

设备的配置为设备定义了一个基本的 Java ME 运行环境，其中包括虚拟机和 Java 核心类库，可以把配置理解为一个针对某一族设备的最小的 Java 平台，其中包括满足该族设备的 Java VM 虚拟机功能的最小子集和针对该族设备的 Java 类库的最小集合。需要注意的是 Configuration 主要针对的是系统级的特性，如基本的 Java 语言虚拟机的特性以及系统级的 Java 核心类库等。

Java ME 配置（Configuration）包括两种，分别是连接设备配置（CDC）和连接受限设备配置（CLDC）。正如前面提到过的，配置分别针对的是某一族系的设备，Java ME 所支持的设备主要分为两种，分别通过 CDC 和 CLDC 支持。

个人移动信息设备：指那些可以进行间隙性网络连接的设备，如移动电话、双向寻呼机、PDA 等，这类设备从性能上来讲属于低端设备，由 CLDC 支持。

共享连接信息设备：指那些网络连接固定、不中断的设备，如电视机顶盒、互联网电视、可视电话等，这类设备从性能上来讲属于高端设备，由 CDC 支持。

1）CDC

连接设备配置，目标设备为总内存大于 2MB，其内存可以为 RAM、ROM 或是 Flash 闪存，其虚拟机支持 Java 2 虚拟机的全部功能。CDC 的虚拟机被称为 CVMcompact VM，这里需要注意的是，CVM 虽然在功能上支持 Java 2 虚拟机的全部功能，但并不等同于 Java SE 的虚拟机。为了支持特定的设备，CVM 做了新的设计。所以从严格的意义上讲，图 1-1 中 CDC 下方的虚拟机应写为 CVM 更确切，但从功能的角度来讲，可以把它粗略地理解为 JVM。CDC 包含的类库与 Java SE 之间的包含关系如图 1-3 所示。

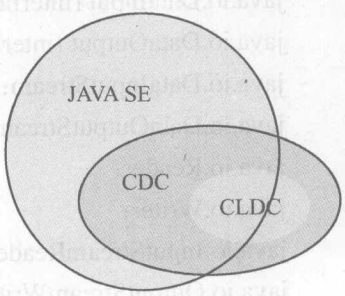


图 1-3　CDC 包含的类库与 Java SE 之间的包含关系

2）CLDC

连接受限设备配置面向的目标设备是小型的、资源有限、连接受限的设备，这些设备的内存在 160KB~512KB 之间，处理器速度较慢，通常是靠电池给设备供电，并且网络连接通常表现为间歇性连接，而且带宽有限。CLDC 为这些设备定义了一个小型的 Java 平台，其实际的表现就是 KVM