目　录

第1章

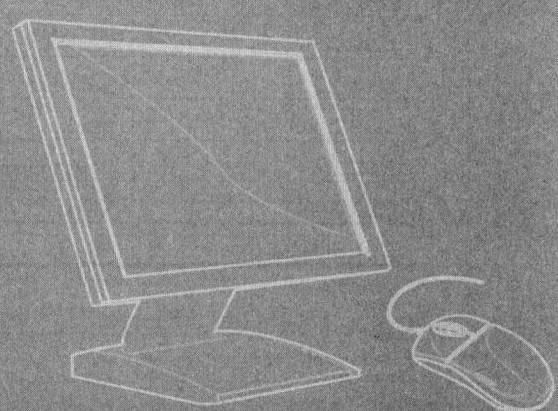
Java 前 奏

学前提示

Java是目前最流行的一门编程语言，要学习Java语言，必须先了解Java的整体概况。本章主要介绍了Java语言的发展历史，体系结构，安装环境和主流IDE集成开发工具等。通过这一章的学习读者会对Java语言有个整体的认识。

知识要点

- Java语言发展简史
- 认识Java语言
- Java平台的体系结构
- Java SE环境安装和配置
- MyEclipse 7.1开发环境的安装和配置



1.1 Java 语言发展简史

Java 作为目前最流行的一门编程语言，它的名字被所有与程序相关的人们所熟知。读者可能会猜测，这个伟大的名字是如何想出来的呢？据说 Java 程序设计语言最早被称为“oak”，但由于当时已经存在一种命名为 oak 的语言。所以不得不放弃 oak 这个名称，在包括一个起名专家在内的众多人员进行的一系列的讨论之后终于选择了“java”这个名称，于是 Java 就这样在一片混乱中诞生了。这个说法在 Java 技术之父 James Gosling 的博客中得到了证实。

Java 语言最早诞生于 1991 年，刚开始它只是 Sun 公司为一些消费性电子产品所设计的通用环境。因为当时 Java 的应用对象只限于 PDA、电子游戏机、电视机顶盒之类的消费性电子产品，所以并未被众多的编程技术人员所接受。

在 Java 出现以前，Internet 上的信息内容都是一些静态的 HTML 文档。正是因为在 Web 中看不到交互式的内容，所以人们很不满意当时的 Web 浏览器，他们迫切希望能够在 Web 上创建一类无须考虑软、硬件平台就可以执行的应用程序，并且这些程序还要有极大的安全保障。正是由于这种需求给 Java 带来了前所未有的施展舞台。

Sun 的工程师从 1994 年起把 Java 技术应用于 Web 上，并且开发出了 HotJava 的第一个版本。从此 Java 的名字逐渐变得广为人知。

到 2009 年年中为止，Java 已经发布了一系列的版本，并且它每发布一个版本都有其自己特有的名字，如表 1.1 所示。

表 1.1 JDK 已发布版本

JDK 版本	名 字	中 文 名	发布时间
JDK 1.1.4	Sparkler	宝石	1997-09-12
JDK 1.1.5	Pumpkin	南瓜	1997-12-13
JDK 1.1.6	Abigail	阿比盖尔(女子名)	1998-04-24
JDK 1.1.7	Brutus	布鲁图(古罗马政治家和将军)	1998-09-28
JDK 1.1.8	Chelsea	切尔西(城市名)	1999-04-08
J2SE 1.2	Playground	运动场	1998-12-04
J2SE 1.2.1	none	无	1999-03-30
J2SE 1.2.2	Cricket	蟋蟀	1999-07-08
J2SE 1.3	Kestrel	美洲红隼	2000-05-08
J2SE 1.3.1	Ladybird	瓢虫	2001-05-17
J2SE 1.4.0	Merlin	灰背隼	2002-02-13
J2SE 1.4.1	grasshopper	蚱蜢	2002-09-16
J2SE 1.4.2	Mantis	螳螂	2003-06-26
J2SE 5.0 (1.5.0)	Tiger	老虎	2004

续表

JDK 版本	名　字	中 文 名	发布时间
J2SE 5.1 (1.5.1)	Dragonfly	蜻蜓	2005
J2SE 6.0 (1.6.0)	Mustang	野马	2006
Java SE 7.0	Dolphin	海豚	开发中

1.2　认识 Java 语言

作为一种程序设计语言，Java 语言具有简单高效、面向对象、不依赖于机器的结构、可移植性、安全性等特点，并且提供了并发机制，具有很高的性能。其次，Java 语言最大限度地利用了网络，Java 的小应用程序(Applet)可在网络上传输而不受 CPU 和环境的限制。另外，Java 还提供了丰富的类库，使程序设计者可以很方便地建立自己的系统。

下面分别从语言特性、Applet 和类库三个方面来讨论 Java 的特点，然后通过把 Java 与它的竞争对手 C、C++、C#进行比较进一步指出它所具有的优点。

1.2.1　Java 语言特性

Java 语言主要有简单高效、面向对象、网络分布计算、健壮性、安全性、跨平台、并发性以及动态扩展等一些特点。Java 语言特性的具体说明如下。

1. 简单高效

Java 语言最初是应用于电子产品的，如冰箱，只需要控制开和关即可完成制冷工作，所以相对来说比较简单。Java 语言提供了很多的功能实现类库，很多代码只需要简单修改便可以很轻松地应用到其他的软件产品中，大大提高了代码的重用率，缩短了开发时间，提高了开发软件的效率。

2. 面向对象

世界万事万物皆是对象。程序员如果要对现实生活中的各种对象进行模拟并编写出大型程序，那么 Java 语言是最好的选择。在面向对象方面，Java 语言比其他面向对象的编程语言更“纯”，所有的数据类型都有相应的类，完全可以用面向对象的方式来编写。

在很多应用中，Java 语言的设计主要集中于对象及其接口，Java 提供了简单的类机制以及动态的接口模型。对象中封装了对象的状态变量以及相应的方法，实现了模块化和信息隐藏；而类则提供了对象的原型，并且通过继承机制，子类可以使用父类所提供的方法，实现了代码的复用。

3. 网络分布计算

Internet 的出现，为网络计算提供了一个良好的信息共享和信息交流的平台。然而，要充分利用网络来处理各种信息，不同操作系统平台具有的不同的运行环境是一个严重的制约，Java 技术的出现为解决网络分布计算提供了最佳途径。Java 语言是面向网络的编程语

言，通过它提供的相应类库可以很方便地处理分布在不同计算机上的对象。

4. 健壮性

Java 程序一般不可能使计算机系统崩溃。因为 Java 虚拟机系统会在编译时对每个 Java 程序进行合法检查，以消除错误的产生。在运行时如果碰到出乎意料的事情，Java 也可以通过异常处理机制，将异常抛出，并由相应的程序进行处理。

5. 安全性

用于网络分布环境下的 Java 产品必须要防止病毒的入侵。Java 语言之所以安全是因为它不支持指针，并提供了字节码校验机制，禁止在自己的处理空间之外破坏内存。

6. 跨平台

Java 源程序通过 Java 解释器解释后会产生与源程序对应的字节码指令，只要在不同的平台上安装配置好相应的 Java 运行环境，Java 程序就可以随处运行。

7. 并发性

Java 内建了对多线程的支持，多线程机制的引入使 Java 程序的运行效率大大提高。同时也保证了对共享数据的正确操作。通过使用多线程，程序设计者可以分别用不同的线程完成特定的功能，而不需要采用全局的事件循环机制，这样就可以很容易地实现网络上的实时交互行为。

8. 动态扩展

Java 语言是一个不断发展的优秀编程语言。Java 语言的类库可以自由地加入新的方法和实例变量而不会影响用户程序的执行，并且通过接口机制改进了传统多继承的缺点，使之比严格的类继承具有更灵活的方式和动态扩展性等。

1.2.2 Java Applet

Java 语言的特性是它可以最大限度地利用网络。Applet 是 Java 的小应用程序，它是动态、安全、跨平台的网络应用程序。Applet 可以嵌入到 HTML 语言中，通过主页发布到 Internet 上。网络用户访问服务器的 Applet 时，这些 Applet 从网络上进行传输，然后在支持 Java 的浏览器中运行。由于 Java 语言的安全机制，用户一旦载入 Applet，就可以放心地生成多媒体的用户界面或完成复杂的计算而不必担心病毒的入侵。虽然 Applet 可以和图像、声音、动画等一样从网络上下载，但它并不同于这些多媒体的文件格式，它可以接收用户的输入，动态地进行改变，而不仅仅是动画的显示和声音的播放。Applet 在早期 Internet 上的应用比较广泛，但当前的应用比较少。

1.2.3 丰富的类库

Java 提供了大量的类以满足网络化、多线程、面向对象系统的需要，类的主要应用领域包括以下几方面。

(1) 语言包提供的支持包括字符串处理、多线程处理、异常处理、数学函数处理等，可以用它简单地实现 Java 程序的运行平台。

(2) 实用程序包提供的支持包括散列表、堆栈、可变数组、时间和日期等。

(3) 输入输出包用统一的“流”模型来实现所有格式的 I/O(输入/输入)，包括文件系统、网络、输入。

(4) 低级网络包用于实现 Socket 编程。

(5) 抽象图形用户接口包实现了不同平台的计算机的图形用户接口部件，包括窗口、菜单、滚动条、对话框等，使得 Java 可以移植到不同平台的机器上。

(6) 网络包支持 Internet 的 TCP/IP 协议，提供了与 Internet 的接口。网络包支持 URL 连接和 WWW 的即时访问，并且简化了用户/服务器模型的程序设计。

(7) 为了适应新的形势，在 JDK5 以后陆续加入了很多新的特性，如注解、泛型、反射等类。

1.2.4　Java 的竞争对手

Java 的不断发展要归功于 C、C++ 和 C# 等编程语言的不断挑战。C++、C#和 Java 等编程语言基本上都来源于 C 语言但又有很多区别。业内人士经常将 C 比作爷爷，C++比做儿子，C# 和 Java 等语言比作孙子。对于变量声明、参数传递、操作符、流控制等，Java 使用了和 C、C++、C# 相同的传统，而 C++ 主要是对 C 的扩展并融入了面向对象的思想，C# 和 Java 语言是纯粹的面向对象的编程语言并吸收了 C、C++ 语言的很多优点，摒弃了很多缺点，但 C#编程语言的运行依赖于 Windows 平台，而 Java 语言不依赖于任何平台，因此使得熟悉 C、C++、C#的程序员能够很方便地转向 Java 编程。具体描述有如下几点。

1. Java 与 C、C++对比

1) 全局变量

在 Java 编程的过程中，不能在类之外定义全局变量，如：

```
public String name;                    //错，不能在类之外定义全局变量
public class GlobalVar{
   public static global_var;           //全局变量也叫成员变量或成员属性
}
```

要定义全局变量，只能通过在一个类中定义公用、静态的变量来实现一个全局变量。在类 GlobalVar 中定义变量 global_var 为 public static，使得其他类可以访问和修改该变量。Java 对全局变量进行了更好的封装。而在 C 和 C++ 中，依赖于不加封装的全局变量常常造成系统的崩溃。

2) 剔除 goto 关键字

虽然在 Java 中将关键字 goto 保留了，但是 Java 不支持 C、C++ 中的 goto 语句，而是通过异常处理语句 try、Catch、final 等来代替 C、C++ 中用 goto 语句来处理遇到错误时跳转的情况，使程序更易读且更结构化。

3) 良好的指针控制

指针是 C、C++ 编程语言中最有魅力的特性，但它的超高使用难度加上超高灵活性，

使得大部分程序员望而止步，在学习 C、C++ 语言进行编程的过程中，通过指针所进行的内存地址操作常常会造成不可预知的错误，同时通过指针对某个内存地址进行显式类型转换后，可以访问一个 C 或 C++ 中的私有成员，从而破坏安全性，造成系统的崩溃。而 Java 语言对指针进行完全的控制，程序员不能直接进行任何指针操作，例如把整数转化为指针，或者通过指针释放某一内存地址等。同时，数组作为类在 Java 中实现，较好地解决了数组访问越界这一问题。

4) 自动内存回收

一般内存资源有限，很容易被程序破坏。在 C 中，程序员通过库函数 malloc()和 free() 来分配和释放内存，在 C++ 中则通过运算符 new 和 delete 来分配和释放内存。再次释放已释放的内存块或未被分配的内存块，会造成系统的崩溃；同样，忘记释放不再使用的内存块也会逐渐耗尽系统资源。而在 Java 中，所有的数据结构都是对象，通过运算符 new 为它们分配内存堆。通过运算符 new 可以得到对象的处理权，而实际分配给对象的内存可能随程序运行而改变，Java 对此自动地进行管理并且进行垃圾收集，有效防止了由于程序员的误操作而导致的错误，并且更好地利用了系统资源。

5) 固定的数据类型

在 C、C++ 语言中不同数据类型在不同的平台上所占的位数不一样，例如，int 类型的数据在 IBM PC 中占 16 位，在 VAX-II 中占 32 位，这就导致了代码的不可移植性。但在 Java 中，对于这些数据类型都采用国际统一字符编码，即分配固定长度的位数，例如，对 int 类型的数据，它在任何机器上都占 32 位，这就保证了 Java 的平台无关性。

6) 严格控制数据类型转换

一种数据类型的数据转换成另外一种数据类型的数据时，常常会出现数据精度丢失的问题，在 C、C++ 中，通过指针进行任意的数据类型转换极不安全，而在 Java 中，运行时系统对对象的处理要进行类型相容性检查，以防止不安全的转换。

7) 库文件

编程语言中丰富的库文件能快速地开发出各种应用软件。C、C++ 中用头文件来声明类的原型以及全局变量、库函数等，在大的系统中，维护这些头文件是很困难的。而 Java 不支持头文件，类成员的类型和访问权限都封装在一个类中，运行时系统对访问进行控制，防止对私有成员的操作。同时，Java 中用 import 语句来与其他类进行通信，以便使用它们的方法。

8) 类与结构体和联合体

安全是一个永恒的话题。C、C++ 中的结构体和联合体中的所有成员均为公有，这就带来了安全性问题。Java 中不包含结构体和联合体，所有的内容都封装在类中。

其实 Java 与 C、C++ 编程语言还有很多的差别如：速度、内部类、方法嵌入等，但总的来说 Java 提取了很多其他编程语言的优点，使它更适合于大众程序员的需求。

2. Java 与 C# 对比

Java 语言是开放式的世界语言，基本源代码都公开，而 C# 作为 Microsoft 的一门主打语言也不甘示弱。一个开源，一个收费，它们两者基本上都对 C、C++ 深涩的语法和语义进行了改进。在语法方面，两者都摒弃了 const 修饰、宏替换等；在继承方面，两者都采用

更易于理解的单继承和多接口实现方案；在源代码组织方面，两者都提出了声明与实现于一体的逻辑封装。

Java 与 C# 的不同点主要体现在：C# 在 Microsoft 的支撑下提供了强大的 Visual Studio 开发平台，可以极好地提高 C# 程序的开发效率。而且 C#更善于利用 Windows 平台。Java 的设计宗旨是独立于任何平台，因此自然不会提供太多的 Windows 特性。但这也正体现了 Java 语言的跨平台优势。一般企业级应用，无法确定这个应用是在怎样的平台上运行。因而企业级开发一般选择 Java 作为开发语言。

1.2.5　Java 在应用领域的优势

图 1.1 所示为目前各种编程语言的使用排名。

Position Feb 2009	Position Feb 2008	Delta in Position	Programming Language	Ratings Feb 2009	Delta Feb 2008	Status
1	1	=	Java	19.401%	-2.08%	A
2	2	=	C	15.837%	+0.98%	A
3	5	↑↑	C++	9.633%	+0.36%	A
4	3	↓	(Visual) Basic	8.843%	-2.76%	A
5	4	↓	PHP	8.779%	-1.11%	A
6	8	↑↑	C#	5.062%	+0.55%	A
7	7	=	Python	4.567%	-0.20%	A
8	6	↓↓	Perl	4.117%	-2.09%	A
9	9	=	Delphi	3.624%	+0.83%	A
10	10	=	JavaScript	3.540%	+1.21%	A
11	11	=	Ruby	3.278%	+1.42%	A
12	12	=	D	1.259%	+0.07%	A
13	13	=	PL/SQL	0.988%	+0.01%	A
14	14	=	SAS	0.835%	-0.11%	A
15	22	↑↑↑↑↑↑↑	Logo	0.813%	+0.50%	A–
16	17	↑	Pascal	0.689%	+0.24%	B
17	29	↑↑↑↑↑↑↑↑↑↑↑↑	ABAP	0.574%	+0.42%	B
18	21	↑↑↑	ActionScript	0.539%	+0.22%	B
19	26	↑↑↑↑↑↑↑	RPG (AS/400)	0.505%	+0.33%	B
20	18	↓↓	Lua	0.487%	+0.10%	B

图 1.1　各种编程语言的使用排名

从图 1.1 中可以看出 Java 语言依然是排名第一的语言，应用非常广泛。Java 语言在应用领域占有较大优势，具体体现在以下几个方面。

(1) 开发桌面应用程序，如：银行软件、商场结算软件等。

(2) 开发面向 Internet 的 Web 应用程序，如：门户网站(工商银行)、网上商城、阿里巴巴、电子商务网站等。

(3) 提供各行业的数据移动、数据安全等方面的解决方案，如：金融、电信、电力等。

Java 语言目前已发展成为最优秀的应用软件开发语言，它有着众多的开源工具，另外，Java 为了实现极度的伸缩能力，增加了产品的复杂性。C、C++ 开发的程序运行速度快，但缺点是没什么开源工具，学习难度大；C# 虽然封装得较好，但它开发的程序不能跨平台运行，并且需要安装大量的运行环境。基于以上原因，Java 的使用者越来越多，Java 在应用程序开发领域所占的份额也越来越大。

1.3 Java 平台的体系结构

作为功能强大的编程语言，Java 发展到今天按其应用来分可以分为三个版本，分别是 JavaSE、JavaEE 和 JavaME，这也就构成了 Java 平台体系结构。Java 平台的体系结构基本上囊括了不同 Java 开发人员对特定市场的需求，下面具体介绍 Java 的这三个版本。

1.3.1 Java SE 标准版

Java SE(Java Platform，Standard Edition)标准版是各种应用平台的基础，主要应用于桌面开发和低端商务应用的解决方案。Java SE 也包含了支持 Java Web 服务开发的类库，并为 Java EE 提供了基础。Java SE 1.4 与 1.5 以后的版本有很大的差别，大多数开发人员都使用 1.6 版本。目前虽然官方没有正式发布 Java SE 7.0，但是开源组织陆续采集了很多高级特性归纳到 Java SE 7.0 中。Java SE 7.0 的组成如图 1.2 所示。

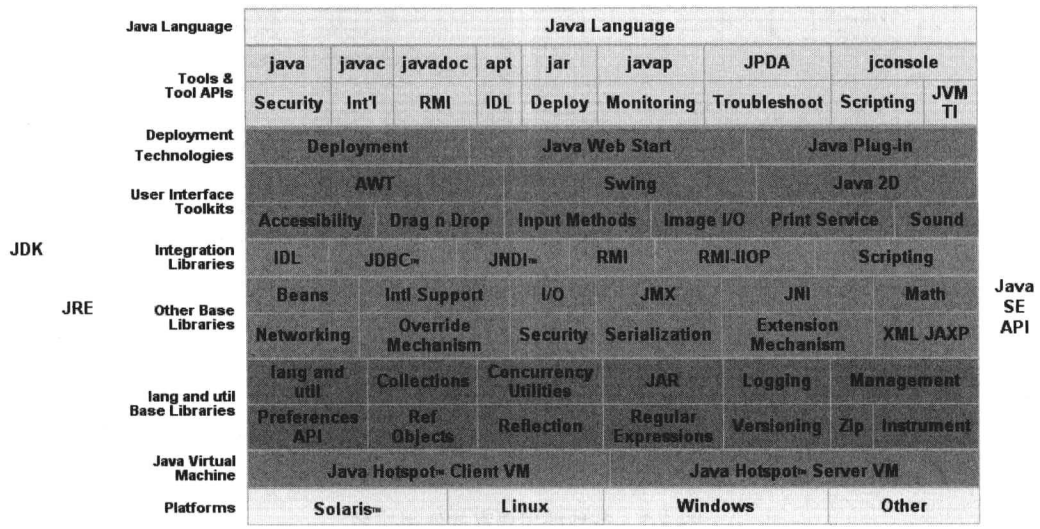


图 1.2 Java SE 7.0 的组成示意图

Java SE 中包含的主要技术如下。

(1) Java Beans Component Architecture 是一个为 Java 平台定义可重用软件组件的框架，可以在图形化构建工具中设计这些组件。

(2) Java Foundation Classes(Swing)(JFC)是一套 Java 类库，支持为基于 Java 的客户机应用程序构建 GUI(Graphical User Interface，图形用户界面)和图形化功能。

(3) Java Help 是一个独立于平台的可扩展的帮助系统，开发人员可以使用它将在线帮助集成到 Applet、组件、应用程序、操作系统和设备中，还可以提供基于 Web 的在线文档。

(4) Java Native Interface(JNI)是 JVM 中运行的 Java 代码，可以与用其他编程语言编写的应用程序和库进行互操作。

(5) Java Platform Debugger Architecture(JPDA)是用于 Java SE 调试支持的基础结构。

(6) Java 2D API 是一套用于高级 2D 图形和图像的类(为图像组合和 alpha 通道图像提供丰富的支持)，一套提供精确的颜色空间定义和转换的类以及一套面向显示的图像操作符。

(7) Java Web Start 允许用户通过一次单击操作下载并启动特性完整的应用程序(比如电子表格)，而不需要进行安装，从而简化了 Java 应用程序的部署。

(8) Certification Path API 提供了一套用于创建、构建和检验认证路径(也称为“认证链”)的 API，可以安全地建立公共密钥到主体的映射。

(9) Java Database Connectivity(JDBC)是一个 API，它使用户能够从 Java 代码中访问大多数表格式数据源，提供了对许多 SQL 数据库的跨 DBMS 连接能力，并可以访问其他表格式数据源，比如电子表格或平面文件。

(10) Java Advanced Imaging(JAI)是一个 API，提供了一套面向对象的接口，这些接口支持一个简单的高级编程模型，使开发人员能够轻松地操作图像。

(11) Java Authentication and Authorization Service(JAAS)是一个包，实现了标准的 Pluggable Authentication Module(PAM)框架的 Java 版本并支持基于用户的授权，能够对用户进行身份验证和访问控制。

(12) Java Cryptography Extension(JCE)是一组包，提供了用于加密、密钥生成和协商以及 Message Authentication Code(MAC)算法的框架和实现。JCE 给对称、不对称、块和流密码提供加密支持，它还支持安全流和密封的对象。

(13) Java Data Objects(JDO)是一种基于标准接口的持久化 Java 模型抽象，使程序员能够将 Java 领域模型实例直接保存到数据库(持久化存储器)中，这可以替代直接文件 I/O、串行化、JDBC 以及 EJB、BMP(Bean Managed Persistence)或 CMP(Container Managed Persistence)实体 Bean 等方法。

(14) Java Management Extensions(JME)提供了用于构建分布式、基于 Web、模块化且动态的应用程序的工具，这些应用程序可以用来管理和监视设备、应用程序和服务驱动的网络。

(15) Java Media Framework(JMF)可以将音频、视频和其他基于时间的媒体添加到 Java 应用程序和 Applet 中。

(16) Java Naming and Directory Interface(JNDI)为 Java 应用程序提供一个连接到企业中的多个命名和目录服务的统一接口，可以无缝地连接结构不同的企业命名和目录服务。

(17) Java Secure Socket Extensions(JSSE)是一组包，它们支持安全的互联网通信，实现了 SSL(Secure Sockets Layer)和 TLS(Transport Layer Security)的 Java 版本，包含了数据加密、服务器身份验证、消息完整性和可选的客户机身份验证等功能。

(18) Java Speech API(JSAPI)包含 Java Speech Grammar Format(JSGF)和 Java Speech Markup Language(JSML)规范，使 Java 应用程序能够将语音技术集成到用户界面中。JSAPI 定义了一个跨平台的 API，支持命令和控制识别器、听写系统及语音识别器。

(19) Java 3D 是一个 API，它提供了一套面向对象的接口，这些接口支持一个简单的高级编程模型，开发人员可以使用这个 API 轻松地将可伸缩的独立于平台的 3D 图形集成到 Java 应用程序中。

(20) Metadata Facility 允许给类、接口、字段和方法标上特定的属性，从而使开发工具、部署工具和运行时能够以特殊方式处理它们。

(21) Java Content Repository API 是一个用于访问 Java SE 中独立于实现的内容存储库的 API。内容存储库是一个高级信息管理系统，是传统数据存储库的超集。

(22) Enumerations(枚举)是一种类型，允许以类型安全的方式将特定的数据表示为常量。

(23) Generics(泛型)允许定义具有抽象类型参数的类，可以在实例化时指定这些参数。

(24) Concurrency Utilities 是一套中级实用程序，提供了并发程序中常用的功能。

(25) Java API for XML Processing(JAXP)允许 Java 应用程序独立于特定的 XML 处理，实现对 XML 文档进行解析和转换，允许灵活地在 XML 处理程序之间进行切换，而不需要修改应用程序代码。Java API for XML Binding(JAXB)允许在 XML 文档和 Java 对象之间进行自动的映射。

(26) SOAP with Attachments API for Java(SAAJ)使开发人员能够按照 SOAP1.1 规范和 SOAP with Attachments note 生成和消费消息。

1.3.2 Java EE 企业版

Java EE(Java Platform，Enterprise Edition)企业版是以企业为环境而开发应用程序的解决方案，这个版本以前称为 J2EE。企业版本帮助开发和部署可移植、健壮、可伸缩且安全的服务器端 Java 应用程序。Java EE 是在 Java SE 的基础上构建的，它提供了 Web 服务、组件模型、管理和通信 API，可以用来实现企业级的面向服务体系结构(Service Oriented Architecture，SOA)和 Web 2.0 应用程序。

Java EE 中包含的主要技术如下。

(1) Enterprise Java Beans(EJB)技术使用一个组件模型来简化中间件应用程序的开发，提供了对事务、安全性和数据库连接等服务的自动支持。

(2) Portlet Specification 定义了一套用于 Java 门户计算的 API，可以解决聚合、个人化、表示和安全性方面的问题。

(3) Java Mail 是一个 API，提供了一套对邮件系统进行建模的抽象类。

(4) Java Message Service(JMS)是一个 API，它为所有与 JMS 技术兼容的消息传递系统定义一套通用的消息概念和编程策略，从而支持开发可移植的基于消息的 Java 应用程序。

(5) Java Server Faces(JSF)提供一个编程模型，帮助开发人员将可重用 UI 组件组合在页面中，将这些组件连接到应用程序数据源，将客户机生成的事件连接到服务器端的事件处理程序，从而轻松地组建 Web 应用程序。

(6) Java Server Pages(JSP)允许 Web 开发人员快速地开发和轻松地维护动态的独立于平台的 Web 页面，并将用户界面和内容生成隔离开，这样设计人员就能够修改页面布局而不必修改动态内容。这种技术使用类似 XML 的标记来封装为页面生成内容的逻辑。

(7) Standard Tag Library for Java Server Pages(JSTL)是一个定制标记集合，它以一种标准化的格式启用许多常见的 Web 站点功能。

(8) Java Servlets 提供了一种基于组件的独立于平台的方法，可以构建基于 Web 的应用程序，同时避免 CGI 程序的性能限制，从而扩展并增强了 Web 服务器的功能。

(9) J2EE Connector Architecture(JCA)为将 J2EE 平台连接到各种结构的 Enterprise Information Systems(EIS)定义了一个标准的体系结构，它定义了一套可伸缩的安全的事务性机制，使 EIS 厂商能够提供标准的资源适配器，可以将这些资源适配器插入到应用服务

器中。

(10) J2EE Management Specification(JMX)为 J2EE 平台定义了一个信息管理模型。根据其设计，J2EE Management Model 可与多种管理系统和协议进行互操作；包含模型到 Common Information Model(CIM)的标准映射，CIM 是一个 SNMP Management Information Base(MIB)；还可以通过一个驻留在服务器上的 EJB 组件——J2EE Management EJB Component (MEJB)映射到 Java 对象模型。

(11) Java Transaction API(JTA)是一个独立于实现和协议的高级 API，它使应用程序和应用服务器可以访问事务。Java Transaction Service(JTS)指定了 Transaction Manager 的实现，它支持 JTA 并在这个 API 之下的层上实现 OMG Object Transaction Service(OTS)1.1 规范的 Java 映射。JTS 使用 Internet Inter-ORB Protocol(IIOP)传播事务。

1.3.3　Java ME 微型版

Java ME(Java Platform，Micro Edition)微型版致力于消费产品和嵌入式设备的最佳解决方案，这个版本以前称为 J2ME。Java ME 为在移动设备和嵌入式设备(比如手机、PDA、电视机顶盒和打印机)上运行的应用程序提供一个健壮且灵活的环境。Java ME 包括灵活的用户界面、健壮的安全模型、许多内置的网络协议以及对可以动态下载的联网和离线应用程序的丰富支持。基于 Java ME 规范的应用程序只需编写一次就可以用于许多设备，而且可以利用每个设备的自身功能。

Java ME 中包含的主要技术如下。

(1) Connected Limited Device Configuration(CLDC)描述最基本的库和虚拟机特性，所有包含 K 虚拟机(K Virtual Machine，KVM)的 J2ME 环境实现中都必须提供这些库和特性。

(2) Mobile Information Device Profile(MIDP)提供核心应用程序功能，包括用户界面、网络连接、本地数据存储和应用程序生命周期管理。

(3) Connected Device Configuration(CDC)是一个基于标准的框架，用来构建和交付可以跨许多连接网络的消费类设备和嵌入式设备共享的应用程序。

(4) Mobile 3D Graphics API for J2ME(M3G)是一种轻量的交互式 3D 图形 API，它作为可选的包与 J2ME 和 MIDP 结合使用。

1.4　Java SE 环境安装和配置

用 Java 语言编写出来的程序要在各种平台上运行，必须要预先安装和配置好它的运行环境。对于编程开发者而言最重要的就是要安装 JDK。

1.4.1　什么是 JDK

JDK(Java Development Kits) 即 Java 开发工具箱，JDK 中主要包括：

- JRE(Java Run Time Environment，Java 运行时环境)。
- JVM(Java Virtual Machine，Java 虚拟机)主要作用是进行 Java 程序的运行和维护。
- Java API(应用程序编程接口)主要作用是为编程人员提供已经写好的功能，便于快