Java 在线培训学习资料

更多学习资料，请到 www.rzchina.net 下载

第 1 章　Java 语言概述

本章主要介绍什么是 Java 语言，Java 语言的起源与发展以及 Java 语言具有哪些特点。在讲解的时候，为了能突出 Java 语言的特色，详细的分析了其与 C/C++语言的区别，最后讲解了 Java 虚拟机的运行原理。

1.1　Java 的起源与发展

Java 是 SUN 公司在 1995 年推出的新的编程语言，它是一种跨平台的、应用现在网络高速发展的编程语言。在编程语言中，可以认为 B 语言导致了 C 语言的出现，C 语言导致了 C++的出现，而 C++又导致了 Java 语言的出现。

很有意思的是：SUN 公司是在开发应用在家用电器的软件时，开发出 Java 的，他们的第一个思想就是与平台无关性，这也是 Java 最大的特点和优势。

自 Java 正式推出之后，以特有的优势迅速发展，经过几年的发展，Java 已经在软件开发和动态网站上占有相当大的市场。可以说 Java 语言是编程语言中的一场革命，它的每次版本升级都会带来不小的轰动。

1.2　Java 的各种版本

Java 分为 J2SE、J2EE 和 J2ME 三种。J2SE 是 Java 平台标准版，主要应用于桌面程序和 Java 小应用程序开发。J2EE 主要用于企业级开发和大型网站的开发。J2ME 主要用于手机等移动设备程序的开发。

- **Java SDK Micro Edition（J2ME）：**此版本用来开发掌上电脑、手机等移动通信设备。现阶段，不是所有移动设备都支持 Java，只有具备其运行环境的设备才能运行它。
- **Java SDK Standard Edition（J2SE）：**主要用于开发一般的台式机应用程序，平时说的 JDK 其实就是指 **J2SE**，本书也是围绕着它来讲述。
- **Java SDK Enterprise Edition（J2EE）：**用于开发分布式的企业级大型应用程序，其中的核心被称为 EJB（Enterprise Java Beans）。

另外，从开发运行的角度，Java 又可分为 JRE 和 JDK，JRE 是面向最终用户的，而 JDK

是 Java 面向开发人员的。

- **Java Runtime Environment（JRE）：**JRE 是指 Java 运行环境，所有的 Java 程序都需要安装此环境，它是面向最终用户的。
- **Java Developers Kits（JDK）：**JDK 是指 Java 开发工具箱，其中除了 JRE 外还包括 Java 的开发工具，比如编译、调试环境等，它是面向 Java 开发人员的。

1.3 Java 的特点

Java 是一种跨平台、适合于分布式计算机环境的面向对象编程语言。具体来说，它具有如下特性：简单性、面向对象、分布式、解释性、可靠、安全、平台无关、可移植、高性能、多线程、动态性等。下面将重点介绍简单性、Java 语言的面向对象、平台无关、分布式、多线程、可靠和安全等特性。

1.3.1 简单性

Java 语言是一种面向对象的语言，它通过提供最基本的方法来完成指定的任务，只需要知道一些概念就能够编写出一些应用程序。Java 程序相对而言小，他的代码能够在小机器，例如手机上运行，这应该是大家经常看到的。

Java 删除了 C++中极少被使用、难理解和令人混淆的功能。学过 C++的人肯定知道，C++中有很多这种功能，例如运算符重载、多重继承和广泛的自动强迫同型，这些都是很让人头疼的功能，值得高兴的是 Java 把他们都给删除了。在一些人看来，Java 的语法就是 C++的清错版本。

1.3.2 平台无关性

前面已经提到过，Java 是在家用电器软件中出来的。怎么样才能让这种软件在每个上都能正常的运行呢？这就用到了 Java 的平台无关性。在 Java 出现之前，这个问题是当时每个程序员难解决的问题。Java 出现之后，这个问题就彻底解决了。引用他们的目标就是“只要写一次程序，在任何地方，任何时间该程序永远都能够运行”。

Java 是怎么实现平台无关性的呢？只要安装 Java 运行系统，Java 就可以在任何处理器上运行。Java 解释器生成与体系结构无关的字节码指令。Java 解释器生成与体系无关的字节码指令，这些指令对应与 Java 虚拟机里表示，Java 解释器得到字节码后，对它进行转换，使之能够在不同的平台上运行。

Java 的平台无关性是指用 Java 写的应用程序不用修改，就可以在不同的软硬件平台上运行。平台无关有两种：源代码级和目标代码级。C 和 C++是具有一定程度的源代码级平台无关。源代码级平台无关表明了用 C/C++写的程序无需修改，只需重新编译就可以在不同平台上运行。

Java 主要靠 Java 虚拟机 JVM（Java Virtual Machine），在目标代码级实现平台无关性。JVM 是一种抽象机器，它附着在具体操作系统之上，其本身具有一套虚拟机器指令，并有自己的栈、寄存器组等。JVM 通常是在软件上而不是在硬件上实现的。

目前，Sun 系统公司已经设计实现了 Java 芯片，它主要使用在网络计算机上。另外，Java

芯片的出现也会使 Java 更加容易嵌入到家用电器中。在 JVM 上，有一个 Java 解释器，使用它来解释 Java 编译器编译后的程序。Java 编程人员在编写完软件后，通过 Java 编译器，将 Java 源程序编译为 JVM 的字节代码。任何机器只要配备了 Java 解释器，就可以运行这个程序，而不管这种字节码是在何种平台上生成的。有关 Java 平台无关性的原理如图 1.4 所示。

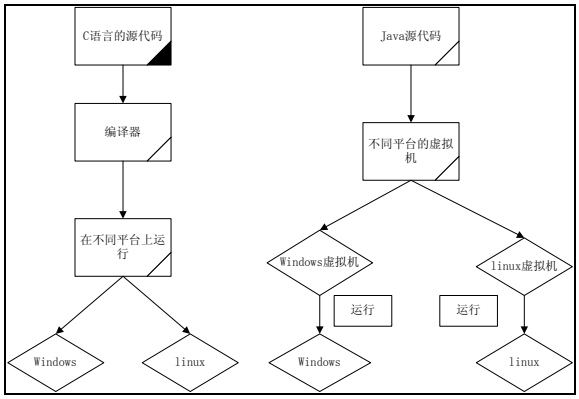

图 1.4　Java 平台无关性示意图

另外，Java 采用的是基于 IEEE 标准的数据类型。通过 JVM 保证数据类型的一致性，也确保了 Java 平台的无关性。

Java 的平台无关性具有深远意义。首先，它的出现使得编程人员所梦寐以求的事情变成了事实，这将大大地加快和促进软件产品的开发。其次 Java 的平台无关性正好迎合了“网络计算机”的思想。

如果常用的应用软件都使用 Java 重新编写，并且放在某个 Internet 服务器上，那么具有网络计算机的用户，将不需要占用大量空间安装软件，他们只需要一个 Java 解释器。每当需要使用某种应用软件时，下载该软件的字节代码即可，其运行结果也可以发回服务器。目前已有数家公司开始使用这种新型的计算机模式，构筑自己的信息系统。

1.3.3　面向对象的特性和多态性

Java 语言是一种纯面向对象语言，可以说它是至今为止最优秀的面向对象语言。Java 的设计集中于对象及其接口，它提供了简单的类机制以及动态的接口模型。对象中封装了它的状态变量和相应的方法，实现了模块化和信息的隐藏；而类则是提供了对象的原型，并且通过继承的机制，子类可以使用父类所提供的方法，实现代码的复用。

面向对象其实是现实世界模型的自然延伸。现实世界中任何实体都可以看作是对象，对象之间通过消息相互作用。另外，现实世界中任何实体都可归属于某类事物，任何对象都是某一类事物的实例。如果说传统的过程式编程语言是以过程为中心，以算法为驱动的话，那面向对象的编程语言就是以对象为中心，以消息为驱动。用公式表示，过程式编程语言为：“程序＝算法＋数据”。面向对象编程语言为：“程序＝对象＋消息”。

所有面向对象编程语言都支持三个概念：封装、多态性和继承，Java 也不例外。现实世界中的对象均有属性和行为映射到计算机程序上。属性则表示对象的数据，行为则表示对象的方法。

封装是用一个自主式的框架，把对象的数据和方法连接在一起，形成一个整体。对象支持封装，是封装的基本单位。Java 语言的封装性较强，那是因为 Java 无全程变量、无主函数。在 Java 中，绝大部分成员是对象，只有简单的数字类型（字符类型和布尔类型除外）。

对于这些类型，Java 提供了相应的对象类型包装，以便与其他对象交互操作。有关封装的原理如图 1.1 所示。

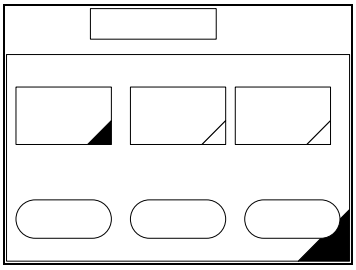

图 1.1　封装的原理示意图

多态性就是多种表现形式。具体来说，可以用“一个对外接口，多个内在实现方法”表示。举一个例子，计算机中的堆栈可以存储各种格式的数据，包括整形、浮点型或字符型，不管存储的是何种数据，堆栈的算法实现都是一样的。针对不同的数据类型，编程人员不必手工选择，只需要使用统一方法名（参数不同），系统可以自动选择。运算符重载一直被认为是一种优秀的多态机制体现。由于考虑到运算符重载会使程序变得难以理解，所以 Java 最后还是把它取消了。有关多态的原理如图 1.2 所示。

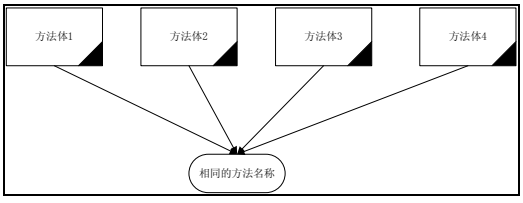

图 1.2　多态的原理示意图

继承是指一个对象直接使用另一个对象的属性和方法。事实上，现实生活中遇到的很多实体，都具有继承的含义。例如，把汽车看成一个实体，它可以分成多个子实体，如：轿车、公交汽车等。以上子实体都具有汽车的特性，因此汽车是它们的“父亲”，而这些子实体则是汽车的“孩子”。子类可以继承父类的属性和方法，与其他面向对象编程语言不同，Java 只支持单一继承。有关继承的原理如图 1.3 所如示。

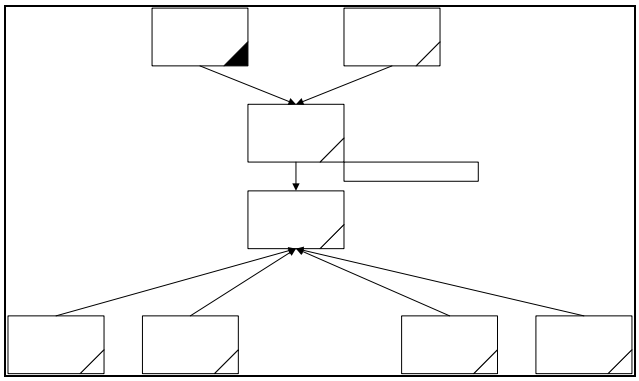

图 1.3　继承的原理示意图

1.3.4　分布式应用

分布式包括数据分布和操作分布。数据分布是指数据可以分散在网络的不同主机上。操

作分布是指把一个计算分散在不同主机上处理。

Java 支持客户机/服务器计算模式，因此它支持这两种分布。对于数据分布，Java 提供了一个叫做 URL 的对象，利用这个对象，可以打开并且访问具有相同 URL 的对象，访问方式与访问本地文件系统相同。对于操作分布，Java 的 Applet 小程序可以从服务器下载到客户端，即部分计算在客户端进行，提高系统执行效率。有关分布式的原理如图 1.5 所示。

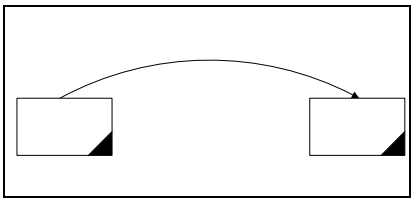

图 1.5　分布式示意图

Java 提供了一整套网络类库，开发人员可以利用这些类库进行网络程序设计，方便的实现 Java 的分布式特性。

1.3.5　多线程

设计 Java 的目标之一就是为了满足人们对创建交互式网上程序的需要。多线程就是因为这个目标设计出来的，它使用 Java 编写出来的应用程序可以同时执行多个任务。多线程机制使应用程序能够并行执行，而且同步机制保证了对共享数据的正确操作。

线程是操作系统的一种新概念，线程又被称作轻量进程，是比传统进程更加小的并且可以并发执行的单位。C 和 C++采用单线程系统结构，而 Java 提供了多线程的支持。

Java 在两方面支持多线程。一方面，Java 环境本身就是多线程的。若干个系统线程运行，负责必要的无用单元回收、系统维护等系统级操作。另一方面，Java 语言内置多线程控制，可以大大简化多线程应用程序的开发。

Java 提供了一个 Thread 类，由它负责启动、运行、终止线程，并且可以检查线程状态。Java 线程还包括一组同步原语，这些原语负责对线程实行并发控制。利用 Java 的多线程编程接口，开发人员可以方便的写出支持多线程的应用程序，从而提供程序执行的效率。Java 的多线程在一定程度上受运行时所在平台的限制，如果操作系统不支持多线程，那么 Java 程序的多线程特性就不能表现出来。

1.3.6　可靠性、安全性

Java 最初的设计目的是电子类消费品，因此要求较高的可靠性。Java 虽然源于 C++，但它消除了许多 C++不可靠的因素，可以防止许多编程错误。

它的可靠性和安全性表现在如下几点：

- Java 是强类型的语言，要求显式的方法声明。这保证了编译器可以发现方法调用错误，保证程序更加可靠。
- Java 不支持指针，这杜绝了内存的非法访问。
- Java 的自动单元收集功能，可以防止内存丢失等动态内存分配导致的问题。
- Java 解释器运行时实施检查，可以发现数组和字符串访问的越界。
- Java 提供了异常处理机制。程序员可以把一组错误代码放在一个地方，这样可简化

错误处理任务，便于恢复。

由于 Java 主要用于网络应用程序开发，因此对安全性有较高的要求。如果没有安全保证，用户从网络下载程序执行就非常危险。Java 通过自己的安全机制，防止了病毒程序的产生，以及下载程序对本地系统的威胁破坏。

当 Java 字节码进入解释器时，首先必须经过字节码校验器的检查，然后 Java 解释器将决定程序中类的内存布局。随后，类装载器负责把来自网络的类装载到单独内存区域，避免应用程序之间相互干扰破坏。最后，客户端用户还可以限制从网络上装载的类只能访问某些文件系统。上述集中机制结合起来，使得 Java 成为安全的编程语言。

1.3.7 小程序和应用程序

用 Java 可以写两种类型的程序：小程序和应用程序。小程序就是嵌入在网页文档中的 Java 程序，而应用程序就是在命令行中运行的程序。对 Java 而言，对小程序的大小和复杂性都没有限制。事实上，Java 小程序有些方面比 Java 应用程序更加强大。目前，由于 Internet 通讯速度有限，因此大多数小程序规模较小。小程序和应用程序之间的技术差别就在于运行环境。

Java 应用程序运行在最简单的环境中，它的惟一外部输入就是命令行参数。另一方面，Java 小程序需要来自 Web 浏览器的大量信息。它需要知道何时启动、何时放入浏览器窗口、何处和何时激活关闭等等。由于这两种不同的执行环境，小程序和应用程序的最低要求不同。

1.4 Java 语言与 C、C++的区别

首先应该清楚，Java 是由 C++发展而来的，保留了 C++的大部分内容，其编程方式类似于 C++。但 Java 的句法更清晰、规模更小、更易学。Sun 公司对多种程序设计语言进行了深入研究，并摒弃了其他语言的不足之处，最终退出了 Java。正是这样，Java 从根本上解决了 C++的固有缺陷，形成了一种新的完全面向对象的语言。

Java 和 C/C++的相似之处多于不同之处，有 C 基础的读者，学习 Java 会更容易。相比较而言，Java 的编程环境更为简单。因篇幅所限，这里不能完全列出不同之处，仅列出一些比较显著的区别。

1.4.1 指针

Java 没有指针的概念，从而有效地防止了在 C / C++语言中，容易出现的指针操作失误（如指针悬空所造成的系统崩溃）。在 C/C++中，指针操作内存时，经常会出现错误。在 Java 中没有指针，更有利于 Java 程序的安全。

1.4.2 多重继承

C++支持多重继承，它允许多父类派生一个子类。也就是说，一个类允许继承多个父类。尽管多重继承功能很强，但使用复杂，而且会引起许多麻烦，编译程序实现它也很不容易。

所以 Java 不支持多重继承，但允许一个类实现多个接口。可见，Java 既实现了 C++多重继承的功能，又避免了 C++的许多缺陷。

1.4.3 数据类型

Java 是完全面向对象的语言，所有方法和数据都必须是类的一部分。除了基本数据类型之外，其余类型的数据都作为对象型数据。例如对象型数据包括字符串和数组。类将数据和方法结合起来，把它们封装在其中，这样每个对象都可实现具有自己特点的行为。而 C++将函数和变量定义为全局的，然后再来调用这些函数和变量，从而增加了程序的负担。此外，Java 还取消了 C／C++中的结构和联合，使编译程序更简洁。

1.4.4 自动内存管理

Java 程序中所有的对象都是用 new 操作符建立在堆栈上的，这个操作符类似于 C++的“new”操作符。Java 自动进行无用内存回收操作，不需要程序员进行删除。当 Java 中一个对象不再被用到时，无须使用内存回收器，只需要给它加上标签以示删除。无用内存的回收器在后台运行，利用空闲时间工作。而 C++中必须由程序释放内存资源，增加了程序设计者的负担。

1.4.5 操作符重载

Java 不支持操作符重载，操作符重载被认为是 C++的突出特征。在 Java 中虽然类可以实现这样的功能，但不支持操作符重载，这样是为了保持 Java 语言尽可能简单。

1.4.6 预处理功能

C／C++在编译过程中都有一个预编译阶段，即预处理器。预处理器为开发人员提供了方便，但增加了编译的复杂性。Java 允许预处理，但不支持预处理器功能，因为 Java 没有预处理器，所以为了实现预处理，它提供了引入语句（import），它与 C++预处理器的功能类似。

1.4.7 Java 不支持缺省函数参数，而 C++支持。

在 C 中，代码组织在函数中，函数可以访问程序的全局变量。C++增加了类，提供了类算法，该算法是与类相连的函数，C++类方法与 Java 类方法十分相似。由于 C++仍然支持 C，所以 C++程序中仍然可以使用 C 的函数，结果导致函数和方法混合使用，使得 C++程序比较混乱。

Java 没有函数，作为一个比 C++更纯的面向对象的语言。Java 强迫开发人员把所有例行程序包括在类中。事实上，用方法实现例行程序可激励开发人员更好地组织编码。

1.4.8 字符串

C 和 C++不支持字符串变量，在 C 和 C++程序中使用“Null”终止符代表字符串的结束，在 Java 中字符串是用类对象（String 和 StringBuffer）来实现的，在整个系统中建立字符串和访问字符串元素的方法是一致的。Java 字符串类是作为 Java 语言的一部分定义的，而不是作为外加的延伸部分。此外，Java 还可以对字符串用“+”进行连接操作。

1.4.9 goto 语句

“可怕”的 goto 语句是 C 和 C++的“遗物”。它是该语言技术上的合法部分，引用 goto 语句造成了程序结构的混乱，不易理解。goto 语句一般用于无条件转移子程序和多结构分支技术。Java 不提供 goto 语句，其虽然指定 goto 作为关键字，但不支持它的使用，这使程序更简洁易读。

1.4.10 类型转换

在 C 和 C++中，有时出现数据类型的隐含转换，这就涉及了自动强制类型转换问题。例如，在 C++中可将一个浮点值赋予整型变量，并去掉其尾数。Java 不支持 C++中的自动强制类型转换，如果需要，必须由程序显式进行强制类型转换。

1.5 Java 虚拟机的运行原理

Java 源程序需要通过编译器编译成为.class 文件（字节码文件），Java 程序的编译和执行过程如图 1.1 所示。

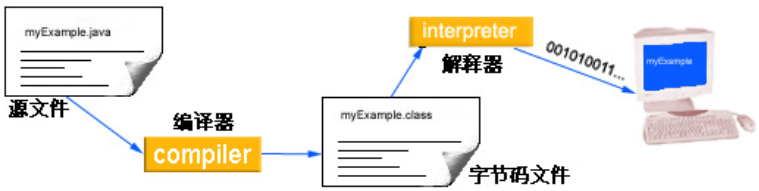

图 1.1 Java 程序的编译和执行过程

Java 虚拟机的建立需要针对不同的软硬件平台做专门的实现，既要考虑处理器的型号，也要考虑操作系统的种类。目前在 SPARC 结构、X86 结构、MIPS 和 PPC 等嵌入式处理芯片上、在 UNIX、Linux、windows 和部分实时操作系统上都有 Java 虚拟机的实现。Java 虚拟机在不同系统和平台上的建立如图 1.2 所示。

1.6 如何才能学好 Java

如何学习 Java，这个问题应该上升到如何学习程序设计这种境界，实际上，学习程序设

计也可以说是接受一种编程思想。每一种语言的程序设计思想大同小异，只是一些由语言特性而带来的细微差别。比如 Java 中的“Interface”，在以前的学习中没有碰到过。以下详细介绍几点：

- 必须明确一个大方向，也就是说在面向对象的编程范畴中，进行学习与研究。目前最流行的面向对象编程语言就是 C++和 Java，所以先锁定这两个目标。
- 掌握 Java 的精华特性，而且一定要知道为什么。比如，nterface 和 Multi-thread。用 Interface 是更好的使用多继承的模型，而多线程则涉及到并发的特性。要完全理解 Interface 是什么、用多线程有几种常用的编程模型等等。
- 理解了语言的特性之后，就可以试着上升到设计这个层次，毕竟学习语言是为了应用。目前，比较好的开发模式是采用自顶向下、结合 MVC 模式的设计。首先要找出最顶层的对象（这往往是最难的），然后一层一层往下递归。所以说，一般有图形用户界面的程序应从界面开始设计。
- 有了基本设计模型后，可以学一些设计模式（Design Pattern）。设计模式有很多种，比如体系结构模式（Layering 分层、Pipe/Filter 管道或过滤器）、设计模式如对象池 Object Pool、缓冲池 Cache 等）、编程模式（比如 Copy-on-Write）。掌握这些模式之后，就会对系统的整体结构有很好的把握。学术上倾向于一个系统完全由各种模式组合而成。
- 学习语言最好的方法就是实践。在一般教科书上的例子并不能算是实践，只能算是掌握了语言的特性。而提倡做实际的 Project 也不是太好，因为还没有熟练的能力去综合各种技术，这样只能是自己越来越迷糊。笔者认为比较好的方法是找一些经典的例子，对其进行进一步的修改。通过修改，找出觉得可以提高性能的地方，加上自己的设计，这样读者才能真正地感到有所收获。

图 1.2 Java 虚拟机的建立

1.7 本章小结

本章首先回顾了 Java 的起源与发展历史，然后阐述了 Java 语言的特点，并且与 C、C++语言进行了比较，。最后又讲述了 Java 虚拟机的运行原理。通过本章的学习，使读者对 Java 语言有了一个大致的了解。