第10卷 第4期
1997年12月

盐城工学院学报
Journal of Yancheng Institute of Technology

Vol. 10 No. 4
Dec. 1997

网络中的世界语——Java语言*

周 进
(南京大学,南京,210093)

摘 要 1995年12月,Sun Microsystems Inc推出的Java程序设计语言环境,使人们看到了解决问题的曙光。Java是一种易移植、高效、简捷,面向对象的程序设计语言并支持运行时的动态环境。与C/C++相比,它具有更加完善的结构无关性、真正的面向对象和与Internet的协同工作等特性。介绍了Java的设计思想及其特性,并提出作者的一些看法。

关键词 Java 异构 线程安全 字节代码 结构中立

分类号 TP31

1 Java简介

1.1 Java的设计目标

Java的设计是面向异构的网间分布,这种环境下的应用软件应满足网络的安全性,使用最少的系统资源、硬件和软件环境,并能动态扩充。设计Java最初是为了开发一个小型可靠、易移植、分布式、实时操作环境。开始选用了C++作为开发工具,但在实际中遇到了很大的困难。Eiffel、Smalltalk、Objective C及C++等都曾是选择对象,但最终设计者还是决定开发一个全新的语言环境。

1.2 Java的特性

随着Internet和World Wide Web的迅猛发展,软件开发的概念需要更新。要想开发一个成功的软件,作为开发工具的Java必须是安全高效、适于异构分布式网络环境多平台软件开发的。Java正是具有下述特性才能满足要求的。

简单易学 Java是一个简单易学的语言,但它的功能仍很强。Java脱胎于C++,采用了类似C++的基本语言结构,这样程序员能很快地掌握它。但Java又比C++有了很大改进,抛弃了C++中的非面向对象和容易引起软件错误的内容,且在网络方面增加了相当多的内容。

面向对象 Java从一开始就被设计成面向对象的。面向对象技术现在已被广泛接受并成为程序设计语言的主流。对象式系统中封装、消息传递等基本机制正好与分布式的基本客户/服务器的环境一致。为适应基于网络的复杂性越来越高的软件开发,程序设计语言更应采纳面向对象的概念。

结构中立与易移植 Java开发的软件应面向的是异构分布式网络环境,这样的应用软件应能运行于多种硬件结构上的多种操作平台的多种用户界面。为了达到这一目标,Java编译器根据源代码形成一种字节代码,这是一种结构中立的中间代码,在多种硬件的软件环境中都能被转换成目标代码。中间代码的思想在程序设计语言的发展过程中并不是新鲜的事物,只不

* 收稿日期:1997-06-05

过在网络上用中间代码实现软件开发的结构中立赋予了这种思想紧扣时代的现实意义。

结构中立性只是易移植性的一部分。Java 语言十分注重易移植性，并严格定义它的基本数据类型的概念及基本的运算，用 Java 写的程序在任何平台上都是一样的。

解释执行与动态性 Java 语言的编译器生成字节代码，也即它的中间代码，运行时不再编译而是解释执行，Java 的解释执行缩短了软件原型开发周期。

Java 的编译器在编译时要做严格的静态检查，但 Java 的运行系统却有很强的动态性，类似在需要时被连接，代码可在网络间异地连接。

健壮性与安全性 软件的可靠性是很重要的一个质量因素。Java 语言提供编译时的静态扩充检查和运行时的第二级检查。在语言设施上，Java 舍弃了指针及指针运算，从而消除 C/C++程序中将出现的一大类错误。

多线程 现代的基于网络的应用软件都要求在同一时间可处理多项事物，Java 的多线程能力即满足了这一要求，并提供高度的交互性。Java 在语言设计上支持多线程，它提供了线程类，运行系统中提供管程与条件锁机制。在类库这一级上，Java 也有支持多线程的特性，它提供了线程安全的库。

高效性 效率一直是程序设计语言好坏的重要标准。Java 的解释器通过省去运行环境的检查使应用程序全速运行，它的自动无用单元收集作为一个后台低级线程运行，使内存资料的作用更合理并随之带来高效率。

1.3 基本 Java 系统

Java 既是一个语言，也是一个环境。基本 Java 系统包括一些基本的实用类库和方法库。这些库有：java.lang(基本数据类型库)、java.io(流与随机存取文件)、java.net(支持网络通信与资源定位)、java.util(实用类)、java.awt(抽象窗口工具包)。

2 Java 语言设施的某些特色

2.1 设计目标

简单性是 Java 最重要的设计目标之一。与 C++的“外形”相似从而易于掌握是它的又一个设计目标。对 C/C++，Ade，Eiffel 熟悉的程序员，掌握 Java 将很快。

许多语言都以“Hello World”作为第一个例子，遵循此传统来看看 Java 程序的样子。

```
Class Hello World {
        static public Void main(string args[]){
                        system.out.println("Hello World!");
    }
}
```

上例中定义了一个类 Hello World，其中有一个方法 main，用于显示，而其中又包含一个完成文件输出的 system 类。

2.2 Java 的主要语言设施

基本数据类型 在 Java 中除了基本数据类型，任何数据都是类，而且在需要的情况下，基本数据类型也可看作对象。Java 中只有四类基本数据类型，即数值类型、布尔类型和数组及字符类型。

数值类型包括:8位的byte、16位的short、32位的int、64位的long、32位float、64位的double。字符类型采用16位的Unicode字符,这与C中的8位字符不同。布尔类型只有true和false两种值,不允许进行转换。

算术和关系运算符　由于没有unsigned类型,Java增加了一个运算符>>>表示无符号右移位。另外+表示字符串连接运算。

数组　与C/C++中不同,Java中的数组是真正的对象,有运行时的实例。

字符串　Java中的字符串也是真正的对象,有两种字符串类:String类和StringBuffer类。在Java中,length0是作为类的一个方法出现的,而不是某个系统函数。

多级中断　Java中彻底去除了goto语句,而用多级的break和continue与标号配合可以灵活控制程序走向而又具有良好的结构。

内存管理和无用单元收集　C和C++中有关内存的操作是不明确且危险的。分配内存,回收释放内存和内存的跟踪,所有这些都要求程序精确地描述内存的除理,这正是造成许多错误,系统崩溃,内存丢失,低效的罪魁祸首。

Java完全摒弃了C风格的指针、指针运算、malloc及free函数。自动的内存单元收集是Java语言及其运行系统的一部分。不过,Java中还是保留了new操作符。

Java的内存管理是基于对象及对象的指引的。指引与指针不同,指针是物理地址,而指引只是符号句柄。Java用指引来标记已分配的存储单元,实际上就是标记对象。当一个对象无指引对其进行标记时,就成了无用单位收集的目标。

Java中的自动无用单元收集是一个低优先权的后台线程,当无其它线程占用CPU周期时,Java就通过它进行内存单位的收集整理。

3　Java语言从C与C++中去除的内容

主要有8个方面:没有Typedef、Define及其它预处理设施、不再有结构(struct)及联合(union)、不再有独立于类的函数、没有多重继承、没有goto语句、没有操作符的重载、不再有自动类型转换、没有指针而代之以指引。

4　Java是面向对象的

Java从一开始就是面向对象的,面向对象技术正好适应了基于客户机/服务器模型的分布式环境。区别对象与过程(函数)的一个重要的特征是一个对象可以具有比生成它的对象更长的生存期。在分布式的环境里,这一特征使一系列功能具有实现的潜在可能:在本地生成一个对象,通过网络进行传送,存储于另一处远程结点,在将来某个时刻可通过网络对其检索。

4.1　Java中的对象技术

封装、多态、继承、动态绑定。

4.2　类与对象的实例化

与C++类似。

4.3　构造函数与终结函数

终结函数相当于析构函数,用finalize0表示。在Java中,内存单元是自动收集的,所以终结函数就有了另外的意义,它一般用于关闭该对象所用的输入输出文件流。

4.4 存取控制

Java 的存取控制分为四级:public,protected,privat 和 friendly。

4.5 抽象方法

抽象方法与 C++的虚函数概念类似。抽象基类与 C++的虚基类概念类似。

5 Java 基于异构分布式网络的特性

5.1 字节代码

字节代码是一种高级的独立于机器的代码,只要有 Java 解释器和异构的运行系统,它可运行于异构的环境中。可见字节代码就是一种中间代码,它被设计成易于集成各种平台并解释执行,并且在必要时还可动态翻译成本地机器的目标代码,以获得高效性。

```
Public synchronized void start() {
        if (kicker=null ‖ ! kicker,isAlive()}
                        kicker=new Thread(this);
                        kicker,start();
        }
}
```

5.2 解释执行与动态性

传统的编译和连接过程使得软件开发周期冗长而繁琐。一个软件的运行要经过编辑,编译,连接,装入,调试,修改再回到编辑的循环往复过程。软件一旦经过修改,就必须重新编译。

Java 程序的运行分两个阶段。先由编译器把源代码翻译成字节代码,运行时刻再由解释器解释执行字节代码。Java 的解释执行和运态性有利于加快软件原型开发速度。

结 语

在现在多种环境林立的计算机世界中,Java 实现了在异构分布式网络中的软件开发,并凭借其简单易学,面向对象,易移植,安全健壮,动态解释执行和多线程等特性,成为一种强有力的网络软件开发工具。Java 不仅是一种语言,也是一个环境。

Java 语言继承了不少过去语言中的精髓,同时又紧紧抓住计算机应用技术的热点,将成为计算机网络中的世界语。

参考文献

1 The Java Language Environment:A White Paper James Gosling & Henry McGilton

2 The Hot Java World—Wide Web Browser Sun Microsystem,Inc

3 The Java Programmer's Guide Sun Microsystem,Inc

4 An Introduction to Object—Oriented Programming Timothy Budd