# 一、Java的历史、现状

## 1.Java历史

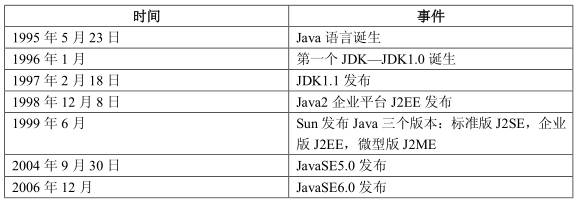
在上世纪90年代初，sun公司有一个叫做Green的项目，目的是为家用消费电子产品开发一个分布式代码系统，这样就可以对家用电器进行控制，和它们进行信息交流。詹姆斯·高斯林(JamesGosling)等人基于C++开发一种新的语言Oak(Java的前身)。Oak是一种用于网络的精巧而安全的语言。Sun公司曾依此投标一个交互式电视项目，但结果是被SGI打败，Sun打算抛弃Oak。随着互联网的发展，Sun看到了Oak在计算机网络上的广阔应用前景，于是改造Oak，在1995年5月以“Java”的名称正式发布，从此Java走上繁荣之路。

当然提到Java历史，不得不提的一个故事就是Java的命名。开始“Oak”的命名是以项目小组办公室外的树而得名，但是Oak商标被其他公司注册了，必须另外取一个名字，传说有一天，几位Java成员组的会员正在讨论给这个新的语言取什么名字，当时他们正在咖啡馆喝着Java(爪哇)咖啡，有一个人灵机一动说就叫Java怎样，得到了其他人的赞同，于是，Java这个名字就这样传开了。当然对于传说，了解一下就好了，不必过于认真。

## 2.Java

作为学习Java的人士，对Java历史上发生的大事件有一个了解是应该的。

JDK(JavaSoftwareDevelopKit))：Java软件开发工具包。JDK是Java的核心，包括了Java运行环境，一系列Java开发工具和Java基础的类库。目前主流的JDK是Sun公司发布的JDK，除了Sun之外，还有很多公司和组织都开发了自己的JDK，例如IBM公司开发的JDK，BEA公司的Jrocket，还有GNU组织开发的JDK等等。



## 3.Java特点

简单地说，Java具有如下特点：简单的、面向对象、平台无关、多线程、分布式、安全、高性能、可靠的、解释型、自动垃圾回收等特点。

这里只解释一下平台无关和分布式，其余的在后面会逐步接触到。

### 3.1平台无关

所谓平台无关指的是：用Java写的程序不用修改就可在不同的软硬件平台上运行。这样就能实现同样的程序既可以在Windows下运行，到了Unix或者Linux环境不用修改就直接可以运行了。Java主要靠Java虚拟机(JVM)实现平台无关性。

平台无关性就是一次编写，到处运行：Write Once, Run Anywhere

### 3.2分布式

分布式指的是：软件由很多个可以独立执行的模块组成，这些模块被分布在多台计算机上，可以同时运行，对外看起来还是一个整体。也就是说，分布式能够把多台计算机集合起来就像一台计算机一样，从而提供更好的性能。

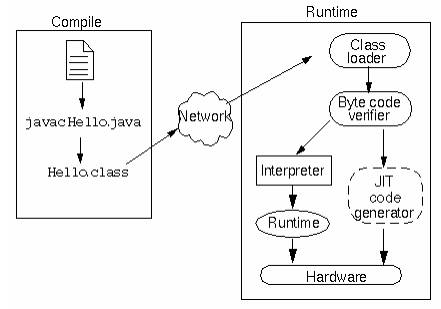
## 4.Java标准组织——JCP

JCP(Java Community Process) 是一个开放的国际组织，成立于 1995 年，主要职能是发展和更新 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。组织成员可以提交 JSR(Java Specification Requests) ，通过讨论、认可、审核以后，将进入到下一版本的规范里面。

也就是说 JCP 是目前 Java 技术发展事实上的控制者和领领导者。

# 二、Java从代码到运行

**用一个图来描述这个过程会比较容易理解：**



## 1.编写代码

首先把我们想要计算机做的事情，通过 Java 表达出来，写成 Java 文件，这个过程就是编写代码的过程。如上图所示的 Hello.java 文件。

## 2.编译

写完 Java 代码后，机器并不认识我们写的 Java 代码，需要进行编译成为字节码，编译后的文件叫做 class 文件。如上图所示的 Hello.class 文件。

## 3.类装载ClassLoader

类装载的功能是为执行程序寻找和装载所需要的类。

ClassLoader能够加强代码的安全性，主要方式是：把本机上的类和网络资源类相分离， 在调入类的时候进行检查，因而可以限制任何“特洛伊木马”的应用。

## 4.字节码(byte-code)校验

功能是对 class 文件的代码进行校验，保证代码的安全性。

Java 软件代码在实际运行之前要经过几次测试。JVM 将代码输入一个字节码校验器以测试代码段格式并进行规则检查--检查伪造指针、 违反对象访问权限或试图改变对象类型的非法代码。

注意----所有源于网络的类文件都要经过字节码校验器字节码校验器对程序代码进行四遍校验， 这可以保证代码符合JVM 规范并且不破坏系统的完整性。如果校验器在完成四遍校验后未返回出错信息，则下列各点可被保证：

- 类符合JVM规范的类文件格式

- 无访问限制异常

- 代码未引起操作数栈上溢或下溢

- 所有操作代码的参数类型将总是正确的

- 无非法数据转换发生，如将整数转换为对象引用

- 对象域访问是合法的

## 5.解释(Interpreter)

可是机器也不能认识 class 文件，还需要被解释器进行解释，机器才能最终理解我们所要表达的东西。

## 6.运行

最后由运行环境中的 Runtime 对代码进行运行，真正实现我们想要机器完成的工作。

## 7.说明

由上面的讲述，大家看到，Java通过一个编译阶段和一个运行阶段，来让机器最终理解我们想要它完成的工作，并按照我们的要求进行运行。

在这两个阶段中，需要我们去完成的就是编译阶段的工作，也就是说：我们需要把我们想要机器完成的工作用 Java 语言表达出来，写成 Java 源文件，然后把源文件进行编译，形成 class 文件， 最后就可以在 Java 运行环境中运行了。 运行阶段的工作由 Java 平台自身提供，我们不需要做什么工作。

# 三、Java程序设计基础

## ****1.Java编程概况****

现在你可以复习一下Java语言的背景材料，它的基本结构象C/C++，但任何用面向过程语言编写过程序的人都可以了解Java语言的大部分结构.

### ****1.1程序结构****

Java语言的源程序代码由一个或多个编译单元(compilationunit)组成，每个编译单元只能包含下列内容(空格和注释除外)：\*一个程序包语句(packagestatement)\*入口语句(importstatements)\*类的声明(classdeclarations)\*界面声明(interfacedeclarations)每个Java的编译单元可包含多个类或界面，但是每个编译单元最多只能有一个类或者界面是公共的。Java的源程序代码被编译后，便产生了Java字节代码。Java的运行系统工作起来如同一台虚拟机。在当前的Java实现中，每个编译单元就是一个以.java为后缀的文件。每个编译单元有若干个类，编译后，每个类生成一个.class文件。.class文件是Java虚拟机能够识别的代码。

### ****1.2注释****

三种类型如下://注释一行/\*一行或多行注释\*//\*\*文档注释\*\*/文档注释一般放在一个变量或函数定义前,指示在任何自动生成文档系统中调入。

### 1.3标识符

变量，函数，类和对象的名称都是标识符，程序员需要标识和使用的东西都需要标识符。在Java语言里,标识符以字符或\_,$开头,后面可以包含数字，标识符是大小写有区别的,没有长度限制。

有效的标识符mynameict\_networkHello\_sys\_path$bill

例子：inta\_number;char\_onechar;float$bill;

### 1.4数据类型

Java使用五种基本类型:integer(整数),floating(浮点数),point(指针),Boolean(布尔变量),CharacterorString(字符或字符串)。

integer整数下边给出的数据表示都是整数的例子：4，15，089，0xAD00整数长度数据类型表示8bitsbyte16bitsshort32bitsint64bitslong

floating浮点数下边给出的数据表示都是浮点数的例子：6.37，3.7E15，3e8

浮点数长度数据类型表示32bitsfloat64bitsdouble

Boolean布尔变量下边是布尔变量的两种可能取值：

truefalseCharacter字符下边给出的都是字符的例子：a\\t(tab)\\u????(unicode)

String字符串下边给出的都是字符串的例子：

\"Thisisastringliteral\"\"中国科学院计算所\"数组你可以定义任意类型的数组.chars[];这是字符型数组;int[]array;这是整型数组;你还可以定义数组的数组.intblock[][]=newint[2][3];数组边界在运行时被检测,避免堆栈溢出和内存崩溃.

在Java里,数组实际上是一个对象,数组有一个成员变量：

length。你可以用这个成员函数来查看任意数组的长度.inta[][]=newint[10][3]a.length/\*10\*/a[0].length/\*3\*/创建数组在Java里创建数组,你可使用两种基本方法：

创建一个空数组：intlist[]=newint[50];或你可以用初始数值填充数组.Stringnames[]={\"Chenji\",\"Yuan\",\"Chun\",\"Yang\"};

相当于下面功能：Stringnames[];names=newString[4];names[0]=newString(\"Chenji\");names[1]

=newString

(\"Yuan\");names[2]=newString(\"Chun\");names[3]=newString(\"Yang\");在编译时你不能象下例那样创建静态数组。intname[50];//将产生一个编译错误你也不能用new操作去填充一个没定义大小的数组。intname[];for(inti=0;i

### 1.5表达式

Java语言的表达式和C语言非常类似。运算符运算符(operator)优先级从高到底排列如下：.[]()++--!～instanceof\*/%+-\\==!=&^&&||?:=op=,整数运算符在整数运算时，如果操作数是long类型，则运算结果是long类型，否则为int类型，绝不会是byte，short或char型。这样，如果变量i被声明为short或byte，i+1的结果会是int。如果结果超过该类型的取值范围，则按该类型的最大值取模。单目整数运算符是：运算符操作-单目非～位补码++加1--减1++运算符用于表示直接加1操作。增量操作也可以用加运算符和赋值操作间接完成。++lvalue(表示lvalue+=1,++lvalue也表示lvalue=lvalue+1(只要lvalue没有副作用)。--运算符用于表示减1操作。++和--运算符既可以作为前缀运算符，也可以做为后缀运算符。双目整数运算符是：运算符操作\*\*+加-减\*乘/除%取模&位与|位或^位异或=，==和!=产生boolean类型的数据。

布尔运算符布尔(boolean)变量或表达式的组合运算可以产生新的boolean值。单目运算符!是布尔非。双目运算符&，|和^是逻辑AND，OR和XOR运算符，它们强制两个操作数求布尔值。为避免右侧操作数冗余求值，用户可以使用短路求值运算符&&和||。用户可以使用==和!=，赋值运算符也可以用&=、|=、^=。三元条件操作符?:和C语言中的一样。

浮点运算符浮点运算符可以使用常规运算符的组合：如单目运算符++、--，双目运算符+、-、\*和/，以及赋值运算符+=，-=，\*=，和/=。此外，还有取模运算：%和%=也可以作用于浮点数，例如：a%b和a-((int)(a/b)\*b)的语义相同。这表示a%b的结果是除完后剩下的浮点数部分。只有单精度操作数的浮点表达式按照单精度运算求值，产生单精度结果。如果浮点表达式中含有一个或一个以上的双精度操作数，则按双精度运算，结果是双精度浮点数。

数组运算符数组运算符形式如下：

[]可给出数组中某个元素的值。合法的取值范围是从0到数组的长度减1。取值范围的检查只在运行时刻实。运算符以String对象实现。运算符\"+\"完成并操作，如果必要则自动把操作数转换为String型。如果操作数是一个对象，它可定义一个方法toString()返回该对象的String方式，例如floata=1.0print(\"Thevalueofais\"+a+\"\\n\");+运算符用到例子Strings=\"a=\"+a;+=运算符也可以用于String。注意，左边(下例中的s1)仅求值一次。s1+=a;//s1=s1+a//若a非String型，自动转换为String型。对象运算符双目运算符instanceof测试某个对象是否是指定类或其子类的实例。例如：if(myObjectinstanceofMyClass){MyClassanothermyObject=(MyClass)myObject;…}是判定myObject是否是MyClass的实例或是其子类的实例。强制和转换Java语言和解释器限制使用强制和转换，以防止出错导致系统崩溃。整数和浮点数之间可以来回

强制转换，但整数不能强制转换成数组或对象。对象不能被强制为基本类型。

### 1.6Java流控制

下面几个控制结构是从C语言借鉴的。分支结构

if/else分支结构if(Boolean){statemanets;}else{statements;}

switch分支结构switch(expr1){caseexpr2:statements;break;caseexpr3:statements;break;default:statements;break;}

循环结构for循环结构for(initexpr1;testexpr2;incrementexpr3){statements;}

While循环结构While(Boolean){statements;}

Do循环结构do{statements;}while(Boolean);

一般顺序控制break[label]continue[label]reutrnexpr;label:statement;

for循环例子下面是一个程序例子，画几条线，分别用红,绿,蓝颜色，这段程序可能是Java函数的一部分：intcount;for(count=1;count

## 2.Java变量和函数的实例

Java的类包含变量和函数。数据变量可以是一些原始的类型,如int,char等。成员函数是一些可执行的过程。例如,下面程序里：publicclassClassOne{inti;publicClassOne(){i=10;}publicvoidAdd\_i(intj){i=i+j;}}ClassOne包含一个变量i和两个成员函数,ClassOne(intfirst)和Add\_i(intj)。

成员函数成员函数是一些可被其它类或自己类调用的处理子程序。一个特殊的成员函数叫构造函数,这个函数名称一般与本类名程相同。它没有返回值。构造函数和成员函数当你在Java里定义一个类时,你可定义一个或多个可选的构造函数,当创建本类的一个对象时用某一个构造函数来初始化本对象。用前面的程序例子来说明,当ClassOne类创建一个新实例时,所有成员函数和变量被创建(创建实例)。构造函数被调用。ClassOnemc:mc=newClassOne();

关键词new用来创建一个类的实例,一个类用new初始化前并不占用内存,它只是一个类型定义,当mc对象初始化后,mc对象里的i变量等于10。你可以通过对象名来引用变量i。(有时称为实例变量)mc.i++;//mc实例变量加1因为mc有ClassOne类的所有变量和成员函数，我们可以使用同样的语法来调用成员函数Add\_i：Add\_i(10);现在mc.i变量等于21.

结束函数Java并不支持析构函数(C++里的定义),因为java本身提供对象无用时自动清除的功能,同时它也提供了一个自动拉圾箱的成员函数，在清除对象时被调用：Protectedvoidfinalize(){close();}

## 3.对象

有效范围和废物自动回收对象有一定的生命期并在它的生命期间使用资源,当一个对象不再被使用时,它应释放内存,避免内存溢出。在Java里,收集和释放内存是一个叫自动废品回收站的线程的责任。这个线程监视对象有效范围并给一个走出有效范围的对象作上标识。

例如：Strings;//没有分配内存s=newString(\"oldstring\");//分配内存s=\"newstring\";//重新分配内存(创建新对象)我们将在以后访问String类时将更加明白它的工作过程,但它的快速工作过程是这样的：1.创建一个新的String类对象并填充以\"oldstring\"2.创建另一个String对象并填充以\"newstring\"注意我们创建了两个对象。Stirng对象\"oldstring\"Stirng对象\"newstring\"在第三条语句里,第一个包括\"oldstring\"的叫做s的对象已走出了有效范围,没有任何方法可以再访问他,我们现在有一个新的对象也叫s,包含\"newstring\"。在下一个废品回收线程,前一个对象将被标识并清除。

## 4.子类

子类是利用存在的对象创建一个新对象的机制,比如,如果你有一个Horse类,你可以创建一个Zebra子类,Zebra是Horse的一种。

classZebraextendsHorse{intnumber\_OF\_stripes:}关键词extends来定义对象有的子类.Zebra是Horse的子类。Horse类里的所有特征都将拷贝到Zebra类里,而Zebra类里可以定义自己的成员函数和实例变量。Zebra称为Horse的派生类或继承。另外,你也许还想覆盖基类的成员函数。用ClassOne说明,下面是一个派生类覆盖Add\_i功能的例子.

importClassOne;publicclassNewClassextendsClassOne{publicvoid

Add\_i(intj){i=i+(j/2);}}

当NewClass类的实例创建时,变量i初始化值为10,但调用Add\_i产生不同的结果。NewClassmnc;mnc=newNewClass();mnc.Add\_i(10);访问控制Java里当你创建一个新类时,你可以标明变量和成员函数的访问层次。

publicpublicvoidAnyOneCanAccess(){}public实例变量和成员函数可以任意其它类调用。

protectedprotectedvoidOnlySubClasses(){}protected实例变量和成员函数只能被其子类调用.

privateprivateStringCreditCardNumber;private实例变量和成员函数只能在本类里调用.

friendlyvoidMyPackageMethod(){}缺省的,如果没有定义任何防火控制,实例变量或函数缺省定义成friendly,意味着可以被本包里的任意对象访问,但其它包里的对象不可访问。所有这个类的对象都只有实例变量的同一个拷贝，这种方法的关键词是static，例如：classBlock{staticintnumber=50;}

所有从Block类创建的对象的number变量值都是相同的。无任在哪个对象里改变了number的值,所有对象的number都跟着改变。同样的,你可以定义static成员函数,但这个成员函数不能访问非static函数和变量。classBlock{staticintnumber=50;intlocalvalue;staticvoidadd\_local(){localvalue++;//没有运行}staticvoidadd\_static(){number++;//运行}}

## 5.this和super

访问一个类的实例变量时,this关键词是指向这个类本身的指针,在前面ClassOne例子中,我们可以增加构造函数如下：

publicclassClassOne{inti;publicClassOne(){i=10;}

publicClassOne(intvalue)this.i=value;}

publicvoidAdd\_i(intj){i=i+j;}}

这里,this指向ClassOne类的指针。如果在一个子类里覆盖了父类的某个成员函数,但又想调用父类的成员函数,你可以用super关键词指向父类的成员函数。

importClassOne;publicclassNewClassextendsClassOne{publicvoidAdd\_i(intj){i=i+(j/2);super.Add\_i(j);}}

下面程序里,i变量被构造函数设成10,然后15，最后被父类(ClassOne)设成25。NewClassmnc;mnc=newNewClass();mnc.Add\_i(10);

## 6.类的类型

至今为止,我用在类前面只用了一个public关键词,其实它有下面4种选择：abstract一个abstract类必须至少有一个虚拟函数,一个abstract类不能直接创建对象,必须继承子类后才能。

final一个final类声明了子类链的结尾,用final声明的类不能再派生子类。publicpublic类能被其它的类访问。在其它包里,如果想使用这个类必须先import,则它只能在它定义的package里使用。

## 7.抽象类

面向对象的一个最大优点就是能够定义怎样使用这个类而不必真正定义好成员函数。如果程序由不同的用户实现时是很有用的，这不需用户使用相同的成员函数名。

在java里Graphics类里一个abstract类的例子如下：publicabstractclassGraphics{publicabstractvoiddrawLine(intx1,inty1,intx2,inty2);

publicabstractvoiddrawOval(intx,inty,intwidth,intheight);publicabstractvoiddrawRect

(intx,inty,intwidth,intheight);...}

在Graphics类里声明了几个成员函数,但成员函数的实际代码是在另外一处地方实现的。

publicclassMyClassextendsGraphics{publicvoiddrawLine(intx1,inty1,intx2,inty2){}}

当一个类包含一个abstract成员函数,这个类必须定义为abstract类。然而并不是abstract类的所有的成员函数都是abstract的。Abstract类不能有私有成员函数(它们不能被实现),也不能有静态成员函数。

## 8.包(Packages)

包(Package)由一组类(class)和界面(interface)组成。它是管理大型名字空间，避免名字冲突的工具。每一个类和界面的名字都包含在某个包中。按照一般的习惯，它的名字是由\".\"号分隔的单词构成，第一个单词通常是开发这个包的组织的名称。

定义一个编译单元的包编译单元的包由package语句定义。如果使用package语句，编译单元的第一行必须无空格，也无注释。其格式如下：packagepackageName;若编译单元无package语句，则该单元被置于一个缺省的无名的包中。使用其它包中的类和界面在Java语言里提供;一个包可以使用另一个包中类和界面的定义和实现的机制。用import关键词来标明来自其它包中的类。一个编译单元可以自动把指定的类和界面输入到它自己的包中。在一个包中的代码可以有两种方式来定义来自其它包中的类和界面：\*在每个引用的类和界面前面给出它们所在的包的名字;//前缀包名法acme.project.FooBarobj=newacme.project.FooBar();\*使用import语句，引入一个类或一个界面，或包含它们的包。引入的类和界面的名字在当前的名字空间可用。引入一个包时，则该包所有的公有类和界面均可用。其形式如下：//从acme.project引入所有类importacme.project.\*;这个语句表示acme.project中所有的公有类被引入当前包。以下语句从acme.project包中进入一个类Employec\_List。//从acme.project而引入Employee\_Listimportacme.project.Employee\_list;Employee\_Listobj=newEmployee

\_List();在使用一个外部类或界面时，必须要声明该类或界面所在的包，否则会产生编译错误。import(引用)类包(classpackage)用import关键词调入,指定package名字如路径和类名,用\*匹配符可以调入多于一个类名。

importjava.Date;importjava.awt.\*;

如果java源文件不包含package,它放在缺省的无名package。这与源文件同目录，类可以这样引入:importMyClass。

Java系统包：Java语言提供了一个包含窗口工具箱，实用程序，一般I/O,工具和网络功能的包。

java.applet这个包包含量了设计applet的类,用一个类Applet和三个接口.AppletContext;AppletStub;和AudioClip.

java.awt另一个窗口工具箱包.awt,包含了产生装饰物和GUI成员的类。这个package包括：

Button,Checkbox,Choice,Component,Graphics,Menu,Pane1,TextArea和TextField。

java.ioI/Opackage包含文件输入/输出类，FileInputStream和FileOutputStream.

java.lang这个包包含Java语言类,包含:对象,线程,异常出口,系统,整数,原点,数学,字符等。

java.net这个类支持TCP/IP网络协议，并包含Socket类,URL和URL相联系的类。

java.util这个类包含程序的同步类,它包含Date,Dictionary类等。

# 四、Java程序编写

象其它编程语言一样，Java 编程语言也被用来创建应用程序。一个共同的应用程序范例是在屏幕上显示字串“Hello World!” 。下列代码给出了这个 Java 应用程序。

虽然很多你可能都不明白，没有关系，主要是来体会一下 Java 程序是什么样子，你可以先看看，有个印象，然后可以先模仿着做。

## 1.HelloWorldApp

1.//

2.// HelloWorld 应用示例

3.//

4.public class HelloWorldApp{

5. public static void main (String args[]) {

6. System.out.println ("Hello World!");

7. }

8.}

以上程序行是在你的屏幕上打印“Hello World!”所需的最少代码。

## 2.描述HelloWorldApp

第 1-3 行

程序中的 1-3 行是注释行

1 //

2 // HelloWorld 应用示例

3 //

第 4 行

第 4 行声明类名为 HelloWorldApp。类名(Classname)是在源文件中指明的，它可在与源代码相同的目录上创建一个 classname · class 文件。在本例题中，编译器创建了一个称为HelloWorldApp.class的文件, 它包含了公共类HelloWorldApp的编译代码。

4 public class HelloWorldApp{

第 5 行

第 5 行是程序执行的起始点。Java 解释器必须发现这一严格定义的点，否则将拒绝运行程序。

其它程序语言(特别是 C 和 C++)也采用 main ()声明作为程序执行的起始点。此声明的不同部分将在本课程的后几部分介绍。

如果在程序的命令行中给出了任何自变量(命令行参数) ，它们将被传递给 main()方法中被称作 args 的 String 数组。在本例题中，未使用自变量。

public static void main (String args[]) {

public-方法 main()可被任何程序访问，包括 Java 解释器。

static- 是一个告知编译器 main()是用于类 HelloWorldApp 中的方法的关键字。

为使 main()在程序做其它事之前就开始运行，这一关键字是必要的。

void- 表明 main()不返回任何信息。这一点是重要的，因为 Java 编程语言要进行谨慎的类型检查，包括检查调用的方法确实返回了这些方法所声明的类型。

String args[] - 是一个 String 数组的声明， 它将包含位于类名之后的命令行中的自变量。

java HelloWorldApp args[0] args[1]· · · ·

第 6 行

第 6 行声明如何使用类名、对象名和方法调用。它使用由 System 类的 out 成员引用的 PrintStream 对象的 println()方法，将字串“Hello World!”打印到标准输出上。

6 System.out.println(“Hello World!”);

在这个例子中，println()方法被输入了一个字串自变量并将其写在了标准输出流上。

第 7-8 行

本程序的 7-8 行分别是方法 main()和类 HelloWorldApp 的下括号。

7 ｝

8 ｝

## 3.编译并运行HelloWorldApp

编译

当你创建了 HelloWorldApp.java 源文件后，用下列程序行进行编译：

javac HelloWorldApp.java

如果编译器未返回任何提示信息， 新文件 HelloWorldApp.class 则被存储在与源文件相同的目录中，除非另有指定。

如果在编译中遇到问题，请参阅本模块的查错提示信息部分。

运行

为运行你的 HelloWorldApp 应用程序，需使用 Java 解释器和位于 bin 目录下的 java程序:

java HelloWorldApp

Hello World!

## 4.编译查错

编译时的错误，以下是编译时的常见错误：

javac:Command not found

PATH 变量未正确设置以包括 javac 编译器。javac 编译器位于 JDK 目录下的 bin 目录。

HelloWorldApp.java:6: Method printl(java.lang.String)

not found in class java.io.PrintStream.System.

out.printl ("Hello World!");

方法名 println 出现打印错误。

In class HelloWorldApp:main must be public or static

该错误的出现是因为词 static 或 public 被放在了包含 main 方法的行之外。

运行时的错误 can’t find class HelloWorldApp(这个错误是在打印 java HelloWorldApp时产生的) ， 通常, 它表示在命令行中所指定的类名的拼写与 filename.class 文件的拼写不同。Java 编程语言是一种大小写区别对待的语言。

例如： public class HelloWorldapp {

创建了一个 HellloWorldapp.class ，它不是编译器所预期的类名(HelloWorldApp.class)。

### 4.1.命名

如果 java 文件包括一个公共类，那么它必须使用与那个公共类相同的文件名。例如在前例中的类的定义是

public class HelloWorldapp

源文件名则必须是 HelloWorldapp.java

### 4.2.类计数

在源文件中每次只能定义一个公共类。

### 4.3.源文件布局

一个 Java 源文件可包含三个“顶级”要素：

(1)一个包(package)声明(可选)

(2)任意数量的导入(import)语句

(3)类(class)声明

该三要素必须以上述顺序出现。即，任何导入语句出现在所有类定义之前;如果使用包声明，则包声明必须出现在类和导入语句之前。

# 五、关键字

## 1.什么是关键字

Java 语言保留的，Java 的开发和运行平台认识，并能正确处理的一些单词。

其实就是个约定，就好比我们约定好，我画个勾勾表示去吃饭。那好了，只要我画个勾勾，大家就知道是什么意思，并能够正确执行了。

关键字这个约定在 Java 语言和 Java 的开发和运行平台之间，我们只要按照这个约定使用了某个关键字，Java 的开发和运行平台就能够认识它，并正确地处理。

## 2.Java中有哪些关键字

abstract do implements private throw

boolean double import protected throws

break else instanceof public transient

byte extends int return true

case false interface short try

catch final long static void

char finally native super volatile

class float new switch while

continue for null synchronized enum

default if package this assert

## 3.Java中关键字的基本含义

• abstract：表明类或类中的方法是抽象的;

• boolean：基本数据类型之一，布尔类型;

• break：提前跳出一个块;

• byte：基本数据类型之一，字节类型;

• case：在 switch 语句中，表明其中的一个分支;

• catch：用于处理例外情况，用来捕捉异常;

• char：基本数据类型之一，字符类型;

• class：类;

• continue：回到一个块的开始处;

• default：用在 switch 语句中，表明一个默认的分支;

• do：用在"do while"循环结构中;

• double：基本数据类型之一，双精度浮点数类型;

• else：在条件语句中，表明当条件不成立时的分支;

• extends：用来表明一个类是另一个类的子类;

• final：用来表明一个类不能派生出子类，或类中的方法不能被覆盖，或声明一个变量是常量;

• finally：用于处理异常情况，用来声明一个肯定会被执行到的块;

• float：基本数据类型之一，单精度浮点数类型;

• for：一种循环结构的引导词;

• if：条件语句的引导词;

• implements：表明一个类实现了给定的接口;

• import：表明要访问指定的类或包;

• instanceof：用来测试一个对象是否是一个指定类的实例;

• int：基本数据类型之一，整数类型;

• interface：接口;

• long：基本数据类型之一，长整数类型;

• native：用来声明一个方法是由与机器相关的语言(如 C/C++/FORTRAN 语言)实现的;

• new：用来申请新对象;

• package：包;

• private：一种访问方式：私有模式;

• protected：一种访问方式：保护模式;

• public：一种访问方式：公共模式;

• return：从方法中返回值;

• short：基本数据类型之一，短整数类型;

• static：表明域或方法是静态的，即该域或方法是属于类的;

• strictfp：用来声明 FP-strict(双精度或单精度浮点数)表达式，参见 IEEE 754 算术规范;

• super：当前对象的父类对象的引用;

• switch：分支结构的引导词;

• synchronized：表明一段代码的执行需要同步;

• this：当前对象的引用;

• throw：抛出一个异常;

• throws：声明方法中抛出的所有异常;

• transient：声明不用序列化的域;

• try：尝试一个可能抛出异常的程序块

• void：表明方法不返回值;

• volatile：表明两个或多个变量必须同步地发生变化;

• while：用在循环结构中;

• assert：声明断言;

• enum：声明枚举类型;

## 4.解释几点

(1)：这些关键字的具体含义和使用方法，会在后面用到的地方讲述

(2)：Java 的关键字也是随新的版本发布在不断变动中的，不是一成不变的

(3)：所有关键字都是小写的

(4)：goto 和 const 不是 Java 编程语言中使用的关键字，但是是 Java 的保留字，也就是说 Java 保留了它们，但是没有使用它们。true 和 false 不是关键字，而是 boolean 类型直接量

(5)：表示类的关键字是 class

# 六、数据类型

## 1.什么叫数据类型

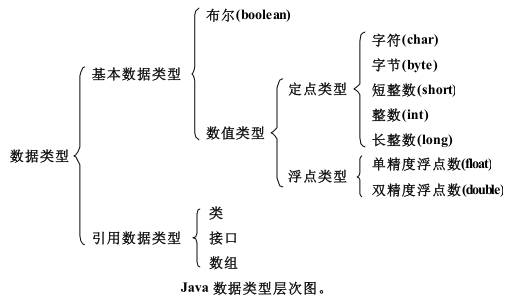
数据类型简单的说就是对数据的分类， 对数据各自的特点进行类别的划分， 划分的每种数据类型都具有区别于其它类型的特征， 每一类数据都有相应的特点和操作功能。 例如数字类型的就能够进行加减乘除的操作。

在现实生活中， 我们通常会对信息进行分类， 从而使得我们能很容易的判断某个数据是表示一个百分数还是一个日期，我们通常是通过判断数字是否带”%”，或者是否是一个我们熟悉的”日期格式”。

类似的在程序中， 计算机也需要某种方式来判断某个数字是什么类型的。 这通常是需要程序员显示来声明某个数据是什么类型的，Java 就是这样的。Java 是一种强类型的语言，凡是使用到的变量，在编译之前一定要被显示的声明。

## 2.Java数据类型的分类

Java 里面的数据类型从大的方面分为两类，一是基本数据类型，一是引用类型，基本 的 Java 数据类型层次图如下：



## 3.Java中的基本数据类型

Java 中的基本数据类型分为八个原始数据类型，原始类型可分为四种：

(1)整数型：byte、short、int、long

(2)浮点型：float、double

(3)字符型：char

(4)逻辑型：boolean

### 3.1.整数型：byte、short、int、long

byte：字节型

short：短整型

int： 整型

long：长整型

在 Java 中，整数型的值都是带符号的数字，可以用十进制、八进制和十六进制来表示。

所谓多少进制， 就是满多少就进位的意思， 如十进制表示逢十进位， 八进制就表示逢八进位。

示例：

15：十进制的 15

015：八进制的 15 ，相当于十进制的 13，计算公式：1\*8 +5 = 13

0x15：十六进制的 15，相当于十进制的 21，计算公式：1\*16+5=21

3.1.1.在Java中的定义示例

示例 1.byte abc = 5;

表示在 Java 中定义一个变量 abc，类型是 byte 类型，值是 5

同理可以定义其它的类型：

比如：

short abc1 = 5;

int abc2 = 5;

long abc3 = 5;

这些都是可以的，如果要明确表示是long型的值，可以在后面直接跟一个字母“L”。L表示一个long值。

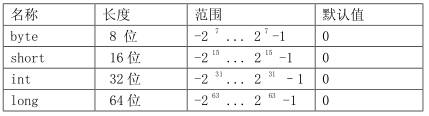
也就是写成：long abc4 = 5L;

请注意，在 Java 编程语言中使用大写或小写 L 同样都是有效的，但由于小写 l 与数字 1 容易混淆，因而，尽量不要使用小写。

注意：整数型的值，如果没有特别指明，默认是是int型

3.1.2取值范围和默认值

取值范围的表示是按 Java 编程语言规范定义的且不依赖于平台



### 3.2.浮点型：float、double

Java 用浮点型来表示实数，简单地说就是带小数的数据。

用 float 或 double 来定义浮点类型，如果一个数字包括小数点或指数部分，或者在数字后带有字母 F 或 f(float) 、D 或 d(double) ，则该数字文字为浮点型的。

示例：

12.3 //简单的浮点型数据

12.3E10 //数据很大的一个浮点数据

3.2.1.在在Java中的定义示例

如下定义都是可以的：

float abc = 5.6F;

float abc = 5.6f;

double abc = 5.6;

double abc = 5.6D;

double abc = 5.6d;

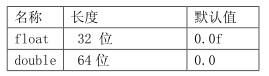
3.2.2.提示

(1)浮点型的值，如果没有特别指明，默认是是double型的

(2)定义float型的时候，一定要指明是float型的，可以通过在数字后面添加”F”或者”f”来表示。

(3)定义double型的时候，可以不用指明，默认就是double型的，也可以通过在数字后面添加”D”或者”d”来表示。

3.2.3 取值范围 和 默认值



Java 技术规范的浮点数的格式是由电力电子工程师学会(IEEE)754 定义的，是独立于平台的。 可以通过 Float.MAX\_VALUE 和 Float.MIN\_VALUE 取得 Float 的最大最小值; 可以通过 Double.MAX\_VALUE 和 Double.MIN\_VALUE 来取得 Double 的最大最小值。

### 3.3.字符型：char

char类型用来表示单个字符。一个char代表一个16-bit无符号的(不分正负的)

Unicode字符，一个char字符必须包含在单引号内。

示例：

‘a’//表示简单的字符

‘1’//用数字也可以表示字符

下面就错了，只能使用单个字符

‘ab’//错误

‘12’//错误

3.3.1.什么是Unicode编码

Unicode 编码又叫统一码、万国码或单一码，是一种在计算机上使用的字符编码。它为每种语言中的每个字符设定了统一并且唯一的二进制编码， 以满足跨语言、 跨平台进行文本转换、 处理的要求。 1990 年开始研发， 1994 年正式公布。 随着计算机工作能力的增强， Unicode也在面世以来的十多年里得到普及。unicode 的表达如下：

‘\u????’ 一个 Unicode 字符。????应严格按照四个 16 进制数进行替换。

3.3.2.在在Java中的定义示例

char c = ‘a’;

char c = ‘1’;

3.3.3.取值范围和默认值

http://cms2.51edu.com/uploadfile/2016/0624/20160624023145969.jpg

3.3.4.Java里面的转义字符

转义字符是指， 用一些普通字符的组合来代替一些特殊字符， 由于其组合改变了原来字符表示的含义，因此称为“转义”。 常见的转义字符：

\n 回车(\u000a)

\t 水平制表符(\u0009)

\b 空格(\u0008)

\r 换行(\u000d)

\f 换页(\u000c)

\' 单引号(\u0027)

\" 双引号(\u0022)

\\ 反斜杠(\u005c)

### 3.4.逻辑型：boolean

逻辑值有两种状态， 即人们经常使用的 “on” 和 “off” 或 “true” 和 “false” 或 “yes”

和“no” ，这样的值是用 boolean 类型来表示的。boolean 有两个文字值，即 true 和 false。

以下是一个有关 boolean 类型变量的声明和初始化：

boolean truth = true; //声明变量 truth

注意──在整数类型和 boolean 类型之间无转换计算。有些语言(特别值得强调的是 C和 C++)允许将数字值转换成逻辑值(所谓“非零即真” ),这在 Java 编程语言中是不允许的;boolean 类型只允许使用 boolean 值(true 或 false) 。

注意：Java中不可以直接将整数类型和逻辑类型转换.

### 3.5.声明和赋值

3.5.1.什么是声明

声明为 Java 程序实体引入标识符， 能够使用这些名字访问到这些实体， 声明实体包括：类名、属性名、方法名、变量名、参数名、接口名等等。其实简单点说就是定义某个东西并对外宣称它。

3.5.2.什么是赋值

赋值就是为一个声明的变量或者常量赋予具体的值， 也就是赋予值的意思。 使用一个等号”=”来表示。

示例：

int a = 5;

这句话的意思就是，声明一个类型为int的变量a，并将它赋值为5。

### 3.6.特别介绍：字符串型String

字符型只能表示一个字符，那么多个字符怎么表示呢？

Java 中使用 String 这个类来表示多个字符， 表示方式是用双引号把要表示的字符串引起来，字符串里面的字符数量是任意多个。字符本身符合 Unicode 标准，且上述 char 类型的反斜线符号 (转义字符) 适用于 String。 与 C 和 C++不同， String 不能用 \0 作为结束。 String的文字应用双引号封闭，如下所示：

“The quick brown fox jumped over the lazy dog.”

char 和 String 类型变量的声明和初始化如下所示：

char ch = ‘A'; // 声明并初始化一个字符变量

char ch1,ch2 ; //声明两个字符变量

// 声明两个字符串变量并初始化它们

String greeting = "Good Morning !! \n" ;

String err\_msg = "Record Not Found !" ;

String str1,str2 ; // 声明两个字符串变量

String s = “12abc”; //基本的字符串型

String s = “”; //表示空串

注意：

(1)：String不是原始的数据类型，而是一个类(class)

(2)：String包含的字符数量是任意多个，而字符类型只能是一个。

要特别注意：”a”表示的是字符串，而’a’表示的是字符类型，它们具有不同的功能。

(3)：String的默认值是null

### 3.7.示例

下列程序显示了如何为整数、浮点数、boolean、字符和 String 类型变量声明和赋值

1. public class Assign {

2. public static void main(String args []) {

3. int x, y; // 声明 int 变量

4.float z = 3.414f; // 声明并赋值 float

5.double w = 3.1415; //声明并赋值 double

6.boolean truth = true; //声明并赋值 boolean

7.char c; // 声明字符变量

8.String str; // 声明 String 字符串

9.String str1 = "bye"; //声明并赋值 String 变量

10.c = 'A'; // 给字符变量赋值

11.str = "Hi out there!"; // 给 String 变量赋值

12.x = 6;

13.y = 1000; // 给 int 变量赋值

14....

15.}

16.}

非法赋值举例

y = 3.1415926; // 3.1415926 不是一个 int.

// 需要类型转换并且小数位要截掉

w = 175,000; // 逗号( , ) 不能够出现

truth = 1; // 一个优秀的 C/C++程序员常犯的错误，在 Java 语言中 boolean 型变量只能为 true 或 false

z = 3.14156 ; // double 型的值不能赋给 float 变量，需要类型转换

# 七、面向对象三大特征

## 1.封装

封装这个词听起来好象是将什么东西包裹起来不要别人看见一样，就好象是把东西装进箱子里面，这样别人就不知道箱子里面装的是什么东西了。其实 JAVA 中的封装这个概念也就和这个是差不多的意思。

封装是 JAVA 面向对象的特点的表现，封装是一种信息隐蔽技术。它有两个含义：即把对象的全部属性和全部服务结合在一起，形成一个不可分割的独立单位; 以及尽可能隐藏对象的内部结构。也就是说，如果我们使用了封装技术的话，别人就只能用我们做出来的东西而看不见我们做的这个东西的内部结构了。

封装的功能

- 隐藏对象的实现细节

- 迫使用户去使用一个界面访问数据

- 使代码更好维护

封装迫使用户通过方法访问数据能保护对象的数据不被误修改，还能使对象的重用变得更简单。数据隐藏通常指的就是封装。它将对象的外部界面与对象的实现区分开来，隐藏实现细节。迫使用户去使用外部界面，即使实现细节改变，还可通过界面承担其功能而保留原样，确保调用它的代码还继续工作。封装使代码维护更简单。

## 2.继承

is a 关系—— 子对象

在面向对象世界里面，常常要创建某对象(如：一个职员对象)，然后需要一个该基本对象的更专业化的版本，比如，可能需要一个经理的对象。显然经理实际上是一个职员，经理和职员具有 is a 的关系，经理只是一个带有附加特征的职员。因此，需要有一种办法从现有对象来创建一个新对象。这个方式就是继承。

“继承”是面向对象软件技术当中的一个概念。 如果一个 对象 A 继承自另一个 对象 B， 就把这个 A 称为"B 的子 对象 "，而把 B 称为"A 的父 对象 "。继承可以使得子 对象 具有父 对象 的各种属性和方法，而不需要再次编写相同的代码。在令子 对象 继承父 对象 的同时，可以重新定义某些属性，并重写某些方法，即覆盖父 对象 的原有属性和方法，使其获得与父对象不同的功能。

## 3.多态

同一行为的多种不同表达，或者同一行为的多种不同实现就叫做多态。

还是用刚才经理和职员这个例子来举例：人事部门需要对公司所有职员统一制作胸卡(一般也就是门禁卡，进出公司证明身份使用)，制作的师傅说，只要告诉我一个人员的信息，就可以制作出一份胸卡，简化一下就是：一位职员的信息对应一份胸卡。

这个时候，对胸卡制作的师傅而言，所有的人都是职员，无所谓是经理还是普通职员。

也就是说，对于传递职员信息这样一个行为，存在多种不同的实现，既可以传递经理的信息， 也可以传递普通职员的信息。这就是多态的表现。

再举一个例子：比如我们说“笔”这个对象，它就有很多不同的表达或实现，比如有钢笔、铅笔、圆珠笔等等。那么我说“请给我一支笔”，你给我钢笔、铅笔或者圆珠笔都可以，这里的“笔”这个对象就具备多态。

# 八、Java类的基本构成

## 1.Java类的定义形式

一个完整的 Java 类通常由下面六个部分组成：

包定义语句

import 语句

类定义{

成员变量

构造方法

成员方法

}

其中：只有类定义和“{}”是不可或缺的，其余部分都可以根据需要来定义。

下面分别来学习各个部分的基本规则，看看如何写 Java 的类。

## 2.包

### 2.1.包是什么

在 Java 中，包是类、接口或其它包的集合，包主要用来将类组织起来成为组，从而对类进行管理。

### 2.2.包能干什么

包对于下列工作非常有用：

(1)：包允许您将包含类代码的文件组织起来，易于查找和使用适当的类。

(2)：包不止是包含类和接口，还能够包含其它包。形成层次的包空间。

(3)：它有助于避免命名冲突。当您使用很多类时，确保类和方法名称的唯一性是非常困难的。包能够形成层次命名空间，缩小了名称冲突的范围，易于管理名称。

为便于管理数目众多的类，Java 语言中引入了“包”的概念，可以说是对定义的 Java类进行“分组” ，将多个功能相关的类定义到一个“包”中，以解决命名冲突、引用不方便、安全性等问题。

就好似当今的户籍制度，每个公民除有自己的名字“张三” 、 “李四”外还被规定了他的户籍地。假定有两个人都叫张三，只称呼名字就无法区分他们，但如果事先登记他们的户籍分别在北京和上海，就可以很容易的用“北京的张三” 、 “上海的张三”将他们区分开来。如果北京市仍有多个张三，还可以细分为“北京市.海淀区的张三” 、 “北京市.西城区.平安大街的张三”等等，直到能惟一标识每个“张三”为止。

JDK 中定义的类就采用了“包”机制进行层次式管理，下图显示了其组织结构的一部分：

从图中可以看出，一个名为 java 的包中又包含了两个子包：io 包和 lang 包。lang 包中包含了 System, String, Object 三个类的定义。事实上，Java 包中既可以包含类的定义，也可以包含子包，或同时包含两者。

简而言之：从逻辑上讲，包是一组相关类的集合;从物理上讲，同包即同目录。

### 2.1.JDK中常用的包

java.lang----包含一些 Java 语言的核心类，包含构成 Java 语言设计基础的类。在此包中定义的最重要的一个类是“Object” ，代表类层次的根，Java 是一个单根系统，最终的根就是“Object” ，这个类会在后面讲到。

Java 并不具有“自由”的方法，例如，不属于任何类的方法，Java 中的所有方法必须始终属于某个类。经常需要使用数据类型转换方法。Java 在 Java.lang 包中定义了“包装对象”类，使我们能够实现数据类型转换。如 Boolean、Character、Integer、Long、Float 和Double，这些在后面会讲到。

此包中的其它类包括：

? Math——封装最常用的数学方法，如正弦、余弦和平方根。

? String，StringBuffer——封装最常用的字符串操作。

你不必显示导入该包，该 Java 包通常已经导入。

java.awt----包含了构成抽象窗口工具集(abstract window toolkits)的多个类，这些类被用来构建和管理应用程序的图形用户界面(GUI)。

javax.swing----完全 Java 版的图形用户界面(GUI)解决方案，提供了很多完备的组件，可以应对复杂的桌面系统构建。

java.net----包含执行与网络相关的操作的类，如 URL, Socket, ServerSocket 等。

java.io----包含能提供多种输入/输出功能的类。

java.util----包含一些实用工具类，如定义系统特性、使用与日期日历相关的方法。还有重要的集合框架。

### 2.2.Java中如何表达包——package语句

Java 语言使用 package 语句来实现包的定义。package 语句必须作为 Java 源文件的第一条语句， 指明该文件中定义的类所在的包。 若缺省该语句， 则指定为无名包， 其语法格式为：

package pkg1[.pkg2[.pkg3…]]; //“[]”表示可选Java 编译 器 把包对 应 于 文 件系 统 的目 录管 理，因此包也可以嵌套使用，即一个包中可以含有类的定义也可以含有子包，其嵌套层数没有限制。package 语句中，用‘.’来指明包的层次;

Java 语言要求包声明的层次和实际保存类的字节码文件的目录结构存在对应关系， 以便将来使用该类时能通过包名(也就是目录名)查找到所需要的类文件。简单地说就是包的层次结构需要和文件夹的层次对应。

注意：每个源文件只有一个包的声明，而且包名应该全部小写。

具体来说，程序员要做以下工作：

### 2.3.编译和生成包

如果在程序 Test.java 中已定义了包 p1，就必须将编译生成的字节码文件 Test.class 保存在与包名同名的子目录中，可以选用下述两种方式之一：

采用下述命令编译：

javac Test.java

则编译器会在当前目录下生成 Test.class 文件， 再在适合位置手动创建一个名为 p1 的子目录，将 Test.class 复制到该 p1 目录下。

采用简化的编译命令，就是可以带包编译

javac -d destpath Test.java

归入该包的类的字节代码文件应放在 java 的类库所在路径的 destpath 子目录下。 现在包的相对位置已经决定了，但 java 类库的路径还是不定的。事实上，java 可以有多个存放类库的目录，其中的缺省路径为 java 目录下的 lib 子目录，你可以通过使用-classpath 选项来确定你当前想选择的类库路径。 除此之外， 你还可以在 CLASSPATH 环境变量中设置类库路径。 destpath 为目标路径， 可以是本地的任何绝对或相对路径。 则编译器会自动在 destpath目录下建立一个子目录 p1，并将生成的.class 文件自动保存到 destpath/p1 下。例如：

javac -d .\ Test.java

javac -d C:\test\ Test.java

### 2.4.带包运行

运行带包的程序，需要使用类的全路径，也就是带包的路径，比如上面的那个程序，就使用如下的代码进行运行：

java p1.Test

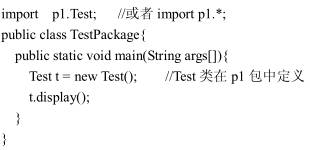
## 3.import

为了能够使用某一个包的成员，我们需要在 Java 程序中明确导入该包。使用“import”语句可完成此功能。在 java 源文件中 import 语句应位于 package 语句之后，所有类的定义之前，可以有 0~多条，其语法格式为：

import package1[.package2…].(classname|\*);

java 运行时环境将到 CLASSPATH + package1.[package2…]路径下寻找并载入相应的字节码文件 classname.class。 “\*”号为通配符，代表所有的类。也就是说 import 语句为编译器指明了寻找类的途径。

例，使用 import 语句引入类程序：TestPackage.java



java 编译器默认为所有的 java 程序引入了 JDK 的 java.lang 包中所有的类(import java.lang.\*;) ，其中定义了一些常用类：System、String、Object、Math 等。因此我们可以直接使用这些类而不必显式引入。但使用其它非无名包中的类则必须先引入、后使用。

### 3.1.Java类搜寻方式

程序中的 import 语句标明要引入 p1 包中的 Test 类， 假定环境变量 CLASSPATH 的值为

“.;C:\jdk6\lib;D:\ex” ， java 运行环境将依次到下述可能的位置寻找并载入该字节码

文件 Test.class：

.\p1\Test.class

C:\jdk6\lib\p1\Test.class

D:\ex\p1\Test.class

其中， “.”代表当前路径，如果在第一个路径下就找到了所需的类文件，则停止搜索。

否则依次搜索后续路径，如果在所有的路径中都未找到所需的类文件，则编译或运行出错。

## 4.访问修饰符

Java 语言允许对类中定义的各种属性和方法进行访问控制， 即规定不同的保护等级来限制对它们的使用。为什么要这样做?Java 语言引入类似访问控制机制的 目的 在 于实 现 信息的 封 装 和 隐藏。 Java 语言为对类中的属性和方法进行有效地访问控制， 将它们分为四个等级：private, 无修饰符, protected, public，具体规则如下：



变量和方法可以处于四个访问级别中的一个：公共，受保护，无修饰符或私有。类可以在公共或无修饰级别。

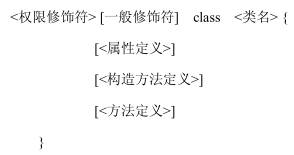
变量、方法或类有缺省(无修饰符)访问性，如果它没有显式受保护修饰符作为它的声明的一部分的话。这种访问性意味着，访问可以来自任何方法，当然这些方法只能在作为对象的同一个包中的成员类当中。

以修饰符 protected 标记的变量或方法实际上比以缺省访问控制标记的更易访问。 一个protected 方法或变量可以从同一个包中的类当中的任何方法进行访问， 也可以是从任何子类中的任何方法进行访问。 当它适合于一个类的子类但不是不相关的类时， 就可以使用这种受保护访问来访问成员。

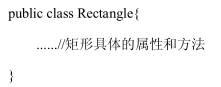
## 5.类定义

Java 程序的基本单位是类，你建立类之后，就可用它来建立许多你需要的对象。Java把每一个可执行的成分都变成类。

类的定义形式如下：



这里，类名要是合法的标识符。在类定义的开始与结束处必须使用花括号。你也许想建立一个矩形类，那么可以用如下代码：



## 6.构造方法

### 6.1.什么是构造方法

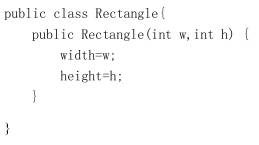
类有一个特殊的成员方法叫作构造方法， 它的作用是创建对象并初始化成员变量。 在创建对象时，会自动调用类的构造方法。

### 6.2.构造方法定义规则

Java 中的构造方法必须与该类具有相同的名字，并且没有方法的返回类型(包括没有void) 。另外，构造方法一般都应用 public 类型来说明，这样才能在程序任意的位置创建类的实例--对象。

### 6.3.示例

下面是一个 Rectangle 类的构造方法，它带有两个参数，分别表示矩形的长和宽：



### 6.4.说明

每个类至少有一个构造方法。 如果不写一个构造方法， Java 编程语言将提供一个默认的，该构造方法没有参数，而且方法体为空。

注意：如果一个类中已经定义了构造方法则系统不再提供默认的构造方

## 7.析构方法

析构方法 finalize 的功能是： 当对象被从内存中删除时， 该成员方法将会被 自动调用。通常，在析构方法内，你可以填写用来回收对象内部的动态空间的代码。

特别注意：当我们去调用析构方法的时候，并不会引起该对象实例从内存中删除，而是不会起到任何作用。

在 Java 编程里面，一般不需要我们去写析构方法，这里只是了解一下就可以了。

## 8.属性

### 8.1.属性是什么

简单点说，属性就是对象所具有的静态属性。

### 8.2.定义规则

Java 类中属性的声明采用如下格式：

访问修饰符 修饰符 类型 属性名称=初始值;

访问修饰符：可以使用四种不同的访问修饰符中的一种，包括 public(公共的)、protected(受保护的) ，无修饰符和 private(私有的) 。public 访问修饰符表示属性可以从任何其它代码调用。private 表示属性只可以由该类中的其它方法来调用。protected 将在以后的课程中讨论。

修饰符：是对属性特性的描述，例如后面会学习到的：static、final 等等。

类型：属性的数据类型，可以是任意的类型。

属性名称：任何合法标识符

初始值：赋值给属性的初始值。如果不设置，那么会自动进行初始化，基本类型使用缺省值，对象类型自动初始化为 null。

### 8.3.说明

属性有时候也被称为成员变量、实例变量、域，它们经常被互换使用。

## 9.方法

### 9.1.方法是什么

方法就是对象所具有的动态功能。

### 9.2.定义规则

Java 类中方法的声明采用以下格式：访问修饰符 修饰符 返回值类型 方法名称 (参数列表) throws 异常列表 {方法体}

访问修饰符：可以使用四种不同的访问修饰符中的一种，包括 public(公共的)、protected(受保护的) ，无修饰符和 private(私有的) 。public 访问修饰符表示方法可以从任何其它代码调用。private 表示方法只可以由该类中的其它方法来调用。protected 将在以后的课程中讨论。

修饰符：是对方法特性的描述，例如后面会学习到的：static、final、abstract、synchronized 等等。

返回值类型：：表示方法返回值的类型。 如果方法不返回任何值， 它必须声明为 void(空)。

Java 技术对返回值是很严格的，例如，如果声明某方法返回一个 int 值，那么方法必须从所有可能的返回路径中返回一个 int 值(只能在等待返回该 int 值的上下文中被调用。 )

方法名称：可以是任何合法标识符，并带有用已经使用的名称为基础的某些限制条件。

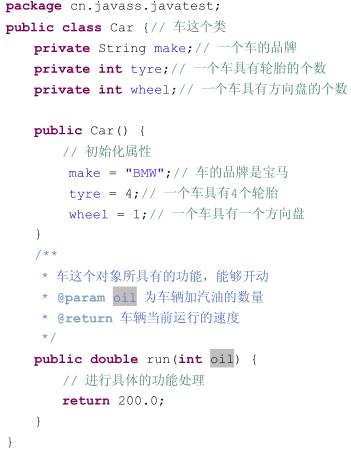
参数列表：允许将参数值传递到方法中。列举的元素由逗号分开，而每一个元素包含一个类型和一个标识符。在下面的方法中只有一个形式参数，用 int 类型和标识符 days 来声明：public void test(int days){}

throws 异常列表：子句导致一个运行时错误(异常)被报告到调用的方法中，以便以合适的方式处理它。异常在后面的课程中介绍。

花括号内是方法体，即方法的具体语句序列。

### 9.3.示例

比如现在有一个“车”的类——Car， “车”具有一些基本的属性，比如四个轮子，一个方向盘，车的品牌等等。当然，车也具有自己的功能，也就是方法，比如车能够“开动”——run。要想车子能够开动，需要给车子添加汽油，也就是说，需要为 run 方法传递一些参数“油”进去。车子跑起来过后，我们需要知道当前车辆运行的速度，就需要 run 方法具有返回值“当前的速度” 。



### 9.4.形参和实参

形参：就是形式参数的意思。是在定义方法名的时候使用的参数，用来标识方法接收的参数类型，在调用该方法时传入。

实参：就是实际参数的意思。是在调用方法时传递给该方法的实际参数。

比如：上面的例子中“int oil”就是个形式参数，这里只是表示需要加入汽油，这个方法才能正常运行，但具体加入多少，要到真正使用的时候，也就是调用这个方法的时候才具体确定，加入调用的时候传入“80” ，这就是个实际参数。

形参和实参有如下基本规则：

(1)：形参和实参的类型必须要一致，或者要符合隐含转换规则

(2)：形参类型不是引用类型时，在调用该方法时，是按值传递的。在该方法运行时，形参和实参是不同的变量，它们在内存中位于不同的位置，形参将实参的值复制一份，在该方法运行结束的时候形参被释放，而实参内容不会改变。

(3)：形参类型是引用类型时，在调用该方法时，是按引用传递的。运行时，传给方法的是实参的地址， 在方法体内部使用的也是实参的地址， 即使用的就是实参本身对应的内存空间。所以在函数体内部可以改变实参的值。

### 9.5.参数可变的方法

从 JDK5.0 开始，提供了参数可变的方法。

当不能确定一个方法的入口参数的个数时，5.0 以前版本的 Java 中，通常的做法是将多个参数放在一个数组或者对象集合中作为参数来传递，5.0 版本以前的写法是：

int sum(Integer[] numbers){…}

//在别处调用该方法

sum(new Integer[] {12,13,20});

而在 5.0 版本中可以写为：

int sum(Integer... numbers){//方法内的操作}

注意：方法定义中是三个点

//在别处调用该方法

sum(12,13,20);//正确

sum(10,11); //正确

也就是说，传入参数的个数并不确定。但请注意：传入参数的类型必须是一致的，究其本质，就是一个数组。

显然，JDK5.0 版本的写法更为简易，也更为直观，尤其是方法的调用语句，不仅简化很多，而且更符合通常的思维方式，更易于理解。