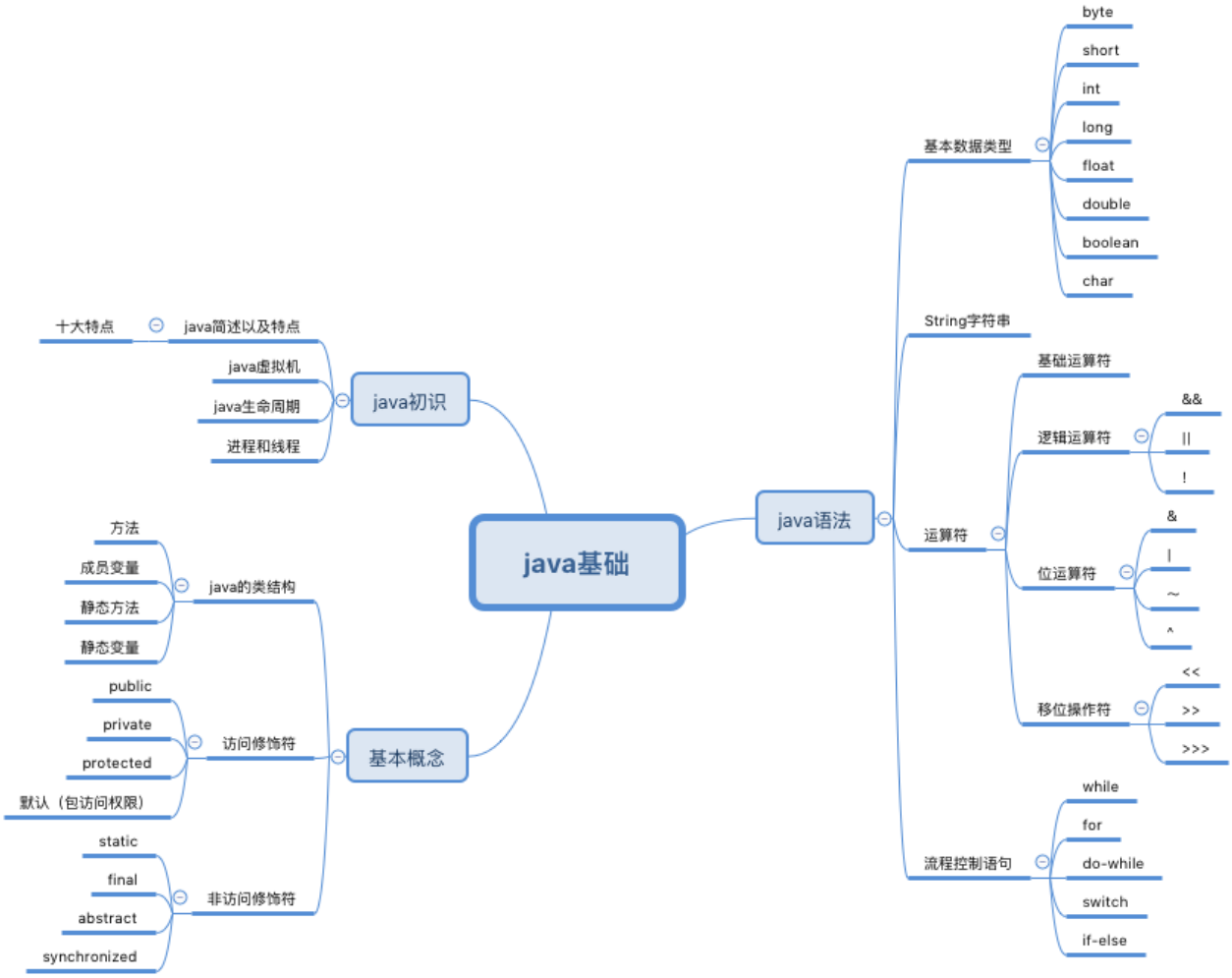
杨兵兵

2017年8月17日

# Java基础




## 一、java初识

### 1、java简述以及特点

Java 是由 Sun Microsystems 在 1995 年首先发布的编程语言和计算平台。有许多应用程序和 Web 站点只有在安装 Java 后才能正常工作，而且这样的应用程序和 Web 站点日益增多。Java 快速、安全、可靠。从笔记本电脑到数据中心，从游戏控制台到科学超级计算机，从手机到互联网，Java 无处不在！

Java是一种编程语言，类似于C++语言，它不仅吸收了C++语言的各种优点，还摒弃了C++里难以理解的多继承、指针等概念。（以下特点摘自百度百科）

1）简单性：Java看起来设计得很像C++，但是为了使语言小和容易熟悉，设计者们把C++语言中许多可用的特征去掉了，这些特征是一般程序员很少使用的。例如，Java不支持go to语句，代之以提供break和continue语句以及异常处理。Java还剔除了C++的操作符过载和多继承特征，并且不使用主文件，免去了预处理程序。Java能够自动处理对象的引用和间接引用，实现自动的收集，使用户不必为存储管理问题烦恼，能更多的时间和精力花在研发上。

2）面向对象：Java是一个面向对象的语言。对程序员来说，这意味着要注意应中的数据和操纵数据的方法，而不是严格地用过程来思考。在一个面向对象的系统中，类是数据和操作数据的方法的集合。数据和方法一起描述对象的状态和行为。每一对象是其状态和行为的封装。类是按一定体系和层次安排的，使得子类可以从超类继承行为。在这个类层次体系中有一个根类，它是具有一般行为的类。Java程序是用类来组织的。Java还包括一个类的扩展集合，分别组成各种程序包，用户可以在自己的程序中使用。例如，Java提供产生图形用户接口部件的类（*java.awt*包），处理输入输出的类（*java.io*包）和支持网络功能的类（*java.net*包）。

3）分布性：java设计成支持在网络上应用，它是分布式语言。Java既支持各种层次的网络连接，又以Socket类支持可靠的流网络连接，所以用户可以产生分布式的客户机和服务器。

网络变成软件应用的分布运载工具。Java程序只要编写一次，就可到处运行。

4）编译和解释性：java编译程序生成字节码，而不是通常的机器码。Java字节码提供对体系结构中性的目标文件格式，代码设计成可有效地传送程序到多个平台。Java程序可以在任何实现了Java解释程序和运行系统的系统上运行。在一个解释性的环境中，程序开发的标准“链接”阶段大大消失了。如果说Java还有一个链接阶段，它只是把新类装进环境的过程，它是增量式的、轻量级的过程。因此，Java支持快速原型和容易试验，它将导致快速程序开发。这是一个与传统的、耗时的“编译、链接和测试”形成鲜明对比的精巧的开发过程。

5）稳健性：Java原来是用作编写消费类家用电子产品软件的语言，所以它是被设计成写高可靠和稳健软件的。Java消除了某些编程错误，使得用它写可靠软件相当容易。Java是一个强类型语言，它允许扩展编译时检查潜在类型不匹配问题的功能。Java要求显式的方法声明，它不支持C风格的隐式声明。Java不支持指针，它消除重写存储和讹误数据的可能性。类似地，Java自动的“无用单元收集”预防存储漏泄和其它有关动态存储分配和解除分配的有害错误。异常处理是Java中使得程序更稳健的另一个特征。异常是某种类似于错误的异常条件出现的信号。使用try/catch/finally语句，程序员可以找到出错的处理代码，这就简化了出错处理和恢复的任务。

6）安全性：java的存储分配模型是它防御恶意代码的主要方法之一。Java没有指针，所以程序员不能得到隐蔽起来的内幕和伪造指针去指向存储器。更重要的是，Java编译程序不处理存储安排决策，所以程序员不能通过查看声明去猜测类的实际存储安排。编译的Java代码中的存储引用在运行时由Java解释程序决定实际存储地址。Java运行系统使用字节码验证过程来保证装载到网络上的代码不违背任何Java语言限制。这个安全机制部分包括类如何从网上装

载。例如，装载的类是放在分开的名字空间而不是局部类，预防恶意的小应用程序用它自己的版本来代替标准Java类。

7）可移植性：Java使得语言声明不依赖于实现的方面。例如，Java显式说明每个基本数据类型的大小和它的运算行为。Java环境本身对新的硬件平台和操作系统是可移植的。Java编译程序也用Java编写，而Java运行系统用ANSIC语言编写。

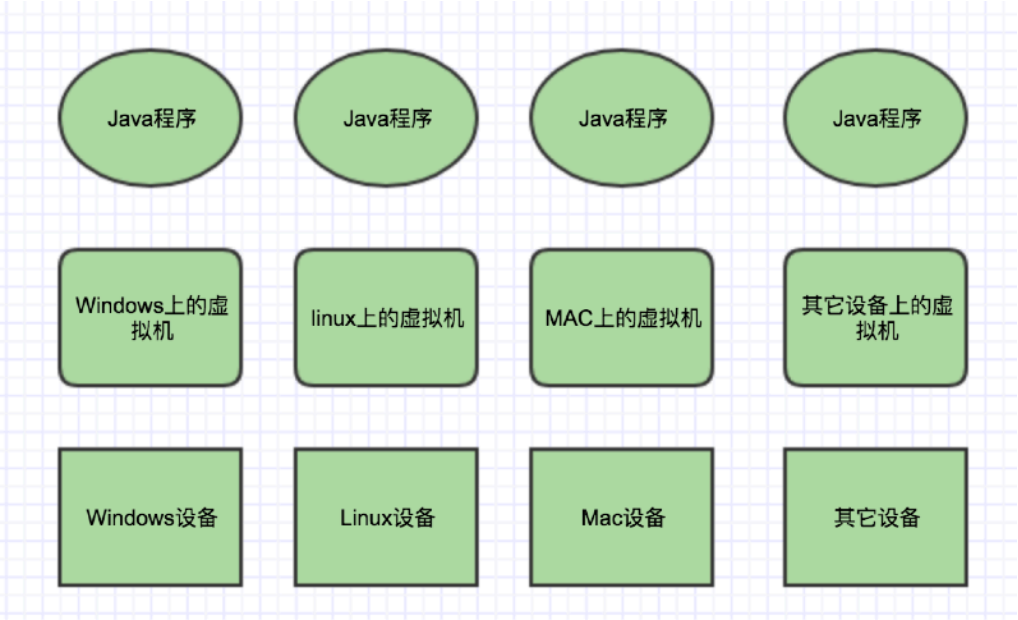
8）高性能：Java是一种先编译后解释的语言，所以它不如全编译性语言快。但是有些情况下性能是很要紧的，为了支持这些情况，Java设计者制作了“及时”编译程序，它能在运行时把Java字节码翻译成特定CPU（中央处理器）的机器代码，也就是实现全编译了。Java字节码格式设计时考虑到这些“及时”编译程序的需要，所以生成机器代码的过程相当简单，它能产生相当好的代码。

9）多线索性：Java是多线索语言，它提供支持多线索的执行（也称为轻便过程），能处理不同任务，使具有线索的程序设计很容易。Java的lang包提供一个Thread类，它支持开始线索、运行线索、停止线索和检查线索状态的方法。Java的线索支持也包括一组同步原语。这些原语是基于监督程序和条件变量风范，由C.A.R.Haore开发的广泛使用的同步化方案。用关键词synchronized，程序员可以说明某些方法在一个类中不能并发地运行。这些方法在监督程序控制之下，确保变量维持在一个一致的状态。

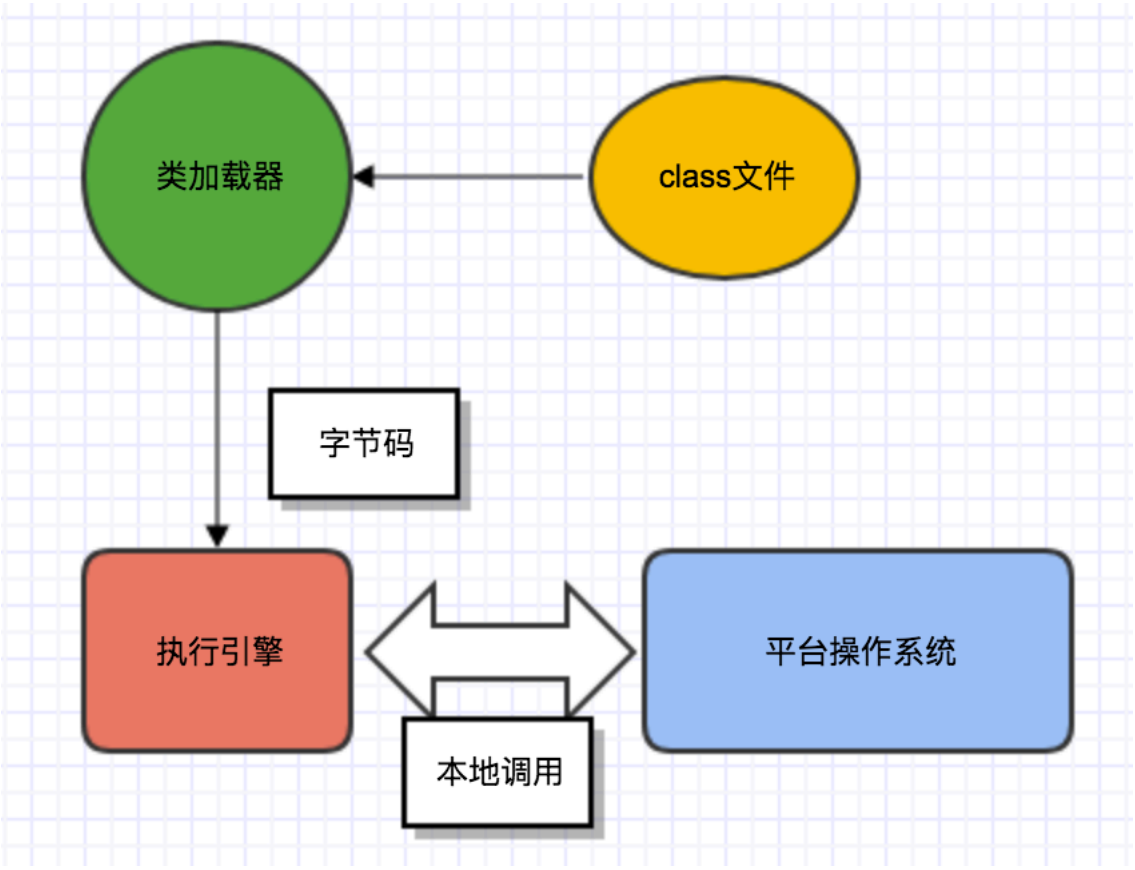
10）动态性：Java语言设计成适应于变化的环境，它是一个动态的语言。例如，Java中的类是根据需要载入的，甚至有些是通过网络获取的。

2、java虚拟机

java之所以能够“编写一次到处运行”，就是因为它引入了Java虚拟机。java程序运行在虚拟机上虚拟机会提供给java程序标准的接口，所有平台的差异化都是由虚拟机解决。在下载JDK的时候也发现了，针对不同的平台有不同的JDK版本，原因就是在不同的平台上要安装不同的虚拟机。如下图所示（程序时相同的，区别在于平台以及运行在其上的虚拟机）



Java虚拟机的主要任务是装载class文件并且执行其中的字节码。Java虚拟机包含一个类装载器（class loader），它可以从程序和API中装载class文件，Java API中只有程序执行时需要的类才会被装载，字节码由执行引擎来执行。（需要记住的是class首先被加载，然后执行引擎来执行）具体的有下图所示：




3、java程序生命周期

1）启动：程序在启动的时候，第一个执行的方法就是main()方法，这个方法和其他的方法有很大的不同，比如方法的名字必须是main，方法必须是public static void 类型的，方法必须接收一个字符串数组的参数。

2）运行：当main方法被执行了以后，可以在在里面开启一些线程来处理一些逻辑，或者是处理用户的操作等等。JVM内部有两种线程:守护线程和非守护线程， main()属于非守护线程，守护线程通常由JVM自己使用，java程序也可以表明自己创建的线程是守护线程。

3）消亡：当程序中的所有非守护线程都终止时，JVM才退出，若安全管理器允许，程序也可以使用Runtime类或者System.exit() 来退出。

4、进程和线程

1）进程：进程是程序的一次动态执行，它对应着从代码加载，执行至执行完毕的一个完整的过程，是一个动态的实体，它有自己的生命周期也即java程序的生命周期。它拥有系统资源（cpu、内存）。

2）线程：线程是指程序在执行过程中，能够执行程序代码的一个执行单元。他是程序执行的最小单元，一个进程可以拥有多个线程，各个线程之间共享程序的内功空间（代码段、数据段和堆空间）及一些进程级的资源（例如打开的文件），但是各个线程都拥有自己的栈空间。
3）关系：一个线程只能属于一个进程，而一个进程可以有多个线程，但至少有一个线程(通常说的主线程)

线程是需要开发人员熟练掌握的，因为在程序中需要多个线程来配合一个功能的实现。

# 二、Java基本概念

1、java的类结构

1）一个java类是由属性（即成员变量和成员常量）和方法组成，从而对数据和操作进行封装。

2）一个java类实例化后就生成一个对象，这时候可以对其中的数据进行操作。一个对象是由状态和行为的。

3）java对象的行为是由其中的方法实现的。是为了执行某个操作的一些语句的组合。

2、访问修饰

1）public共有的，对所有类可见。

2）protected受保护的，对同一包内的类和所有子类可见。

3）private私有的，在同一类内可见。

4）默认的在同一包内可见。默认不使用任何修饰符。

在类继承的时候需要一些继承规则：父类中声明为public的方法在子类中也必须为public。父类中声明为protected的方法在子类中要么声明为protected，要么声明为public。不能声明为private。父类中默认修饰符声明的方法，能够在子类中声明为private。父类中声明为private的方法，不能够被继承。

3、非访问修饰符

1）static 修饰符用于创建类的方法和变量，当修饰变量的时候变量属于是类变量，这个类的所有对象共享；当修饰方法的时候，方法的调用可以不通过对象，直接通过类名.方法名即可调用（这里的限制条件同方法的访问修饰符）。
2）final 修饰符的类最终实现，方法和变量的实现。当用这个方法修饰类的时候，类是不允许被继承；当这个修饰符修饰方法的时候，这个方法是不允许被继承（即不能在子类中覆盖这个方法）；当修饰变量的时候说明这个变量是不允许修改的（即只允许被赋值一次），如果是修饰一个对象的定义则这个对象是不允许修改的，但是对象里的内容可以被修改。
3）abstract 修饰符用于创建抽象类和方法。表示这个类只有被继承并实现了这个方法其子类

才能被实例化，抽象类是不能被实例化。

4）synchronized 和 volatile 修饰符这是用于线程，用于多线程之间访问同一块资源同步问题。避免因为线程不同步导致数据错误。最经典的案例就是网上售票，因为火车站的票个数是一样的，然后不同终端抢票的时候如果不加同步很可能多个人抢到的是同一张票，如果加了同步就会形成一个队列，按序取票，从而避免多个人买到同一张票的情况。

# 三、Java语法

1、基本数据类

1）byte:8个bit，字节类型，取值-128 (-2^7) ~ 127 (2^7 -1)，缺省=0

2）short :16个bit ，短整型，取值-32,768 (-2^15) ~ 32,767 (2^15 -1)，缺省 =0

3） int :32个bit ，整形，取值- 2,147,483,648 (-2^31) ~ 2,147,483,647 (2^31 -1) ，缺省=0

4）long :64个bit ，长整型，取值-9,223,372,036,854,775,808(-2^63) ~ 9,223,372,036,854,775,807 (2^63 -1) ，缺省=0

5）float :32个bit ，浮点型，单精度，数据范围在3.4e-45~1.4e38，直接赋值时必须在数字后加上f或F。缺省=0.0f

6）double:64个bit ，浮点型，双精度，数据范围在4.9e-324~1.8e308，赋值时可以加d或D也可以不加。 缺省=0.0

7）boolean:长度依赖JVM，取值只有true和false，用于条件判断，缺省=false;

8）char:16个bit，Unicode字符，取值’\u0000’(或0)~‘\uffff’(或65，535)，缺 省= ‘\u0000’

java是一种强类型语言，变量的声明或者定义必须要明确其类型，如 int height = 10。int代表类型，height代表变量名，变量的命名规则是：是以$ 、字母、下划线开头，后面的可以是数字、字母、下划线、$，并且大小写敏感，可以随意组合，不过变量名尽量要有一定的意义增加其可读性。变量名也不能是关键字如不能是int，byte....

2、String字符串

1）java中字符串是一个类，里面提供了一些字符串的操作如：equals、indexOf等方法，从而方便开发者对字符串进行操作。

2）String是一个final类，也就是说一个字符串对象是不能进行更改的如String s=“hello world”这里面的hello world就是一个字符串对象而s是一个字符串的引用。当给s重新赋值即s=“hi”的时候其实是新生成了一个字符串对象然后将地址赋给s，之前的hello world如果没有其它字符串对齐引用会被JVM回收掉。

字符串的一些方法大家可以百度一下，里面的方法比较常用，因为字符串操作会经常用到。

3、运算符

1）逻辑运算符共有三种，即“非”、“和”、“或”，分别用"!“、“&&”、“||”。

非运算(!)表示否定，如：!true等于false、!false等于true、!2等于false、!0等于true。

和运算(&&)前后两个条件都为真时，才返回true，否则返回false。

或运算(||)前后两个条件有一个为真是，返回true，都为假时，返回false。

&&运算的时候如果第一个条件为false就不会在往下计算第二个条件就会直接返回false，而||运算是当第一个条件为true的时候就不会再计算第二个条件了。这就是被称为短路运算符的原因了

2）位运算符主要针对二进制进行运算，它包括了：“与”、“非”、“或”、“异或”，分别用"&"、"~"、"|“、"^"。

与运算(&)两个操作数的位都为 1，结果才为 1，否则结果为 0。

非运算(~)的操作数的位如果为 0，结果是 1，如果为 1，结果是 0。

或运算(|)两个操作数的位只要有一个为 1，那么结果就是 1，否则就为 0。

异或运算(^)的两个操作数的位相同时结果为 0，不同时结果为 1。

不同于逻辑运算符，位运算符需要计算每一位的结果。（一个十进制数字进行按位操作需要将十进制转为二进制后按位操作）。

3）移位操作符

<<：左移运算符，num << 1，相当于num乘以2，按二进制形式把所有的数字向左移动对应的位数，高位移出(舍弃)，低位的空位补零。

>>：右移运算符，num >> 1，相当于num除以2，按二进制形式把所有的数字向右移动对应的位数，低位移出(舍弃)，高位的空位补符号位，即正数补零，负数补1。

>>>：无符号右移，忽略符号位，空位都以0补齐，无符号右移运算符只是对32位和64位的值有意义。

4）基础运算符

基础运算符就是大家所熟知的加减乘除运算，不过对于计算机除了加减乘除外还有一些其它的运算符如++，--等。具体的如下图所示已经解释的很清楚了。

| 运算符 | 用法 | 描述 |
| --- | --- | --- |
| + | a + b | 将 a 和 b 相加 |
| + | +a | 如果 a 为 byte、short 或 char，则将它升级为 int |
| - | a - b | 从 a 中减去 b |
| - | -a | 求 a 的算术上的负数 |
| * | a * b | 将 a 和 b 相乘 |
| / | a / b | 将 a 除以 b |
| % | a % b | 返回将 a 除以 b 的余数（取模运算符） |
| ++ | a++ | 将 a 递增 1；计算递增之前 a 的值 |
| ++ | ++a | 将 a 递增 1；计算递增之后 a 的值 |
| -- | a-- | 将 a 递减 1；计算递减之前 a 的值 |
| -- | --a | 将 a 递减 1；计算递减之后 a 的值 |
| += | a += b | a = a + b 的简写形式 |
| -= | a -= b | a = a - b 的简写形式 |
| *= | a *= b | a = a * b 的简写形式 |
| %= | a %= b | a = a % b 的简写形式 |

4、流程控制语句

1）if-else语句

例1:

**if(condition)**{

```
    //执行的代码
}
if(condition){
    //执行的代码
}
```

例2:

```
if(condition){
    //执行的代码
} else if（condition）{
    //执行的代码
} else {
   //执行的代码
}
```

简单来说就是当condition满足的时候就会进入相应的语句块执行相应的代码逻辑。如果是多个if并列如例1，则如果满足条件就都会执行。如果是if-else如上边列2所示只会执行一个语句块，即满足条件的语句块。

2）switch语句

这个语句其实就是一个特殊的if-else语句，也可以说是另外的一种形式。如下所示：

```
switch(condition){
   default:
          System.out.println("打印默认值");
           break;
    case condition1:
           break;
     case condition2:
           break;
      case condition3:
           break;
      ……
 }
```

当condition满足下面的语句的时候就会执行相应case里面的语句，当没有条件满足的时候就会执行default中的语句。condition的类型可以是在java中switch后的表达式的类型只能为以下几种：byte、short、char、int（在Java1.6及以下版本中是这样），在java1.7后支持了对string的判断。break是终止语句说明break以上的代码执行完毕，退出switch语句块。

需要注意的是：在java中如果switch的case语句中少写了break；这个关键字，在编译的时候并不会报错，但是在执行的时候会一直执行所有case条件下的语句并不是去判断，所以会一直执行直到遇到break关键字跳出或者一直执行到defaut语句。这个就是有时候用switch的时候感觉自己写的很对但就是的不到结果的原因了。

3）while循环

循环是在一定条件下反复执行某段程序的流程结构，被反复执行的程序被称为循环体。while语句如下所示

```
while(condition)
  {
    //执行语句
    //改变while里面的那个条件
  }
```

如果在循环体内不改变condition的状态，循环就会一直持续下去，成为死循环。

4）do-while循环

do-while循环是先进行循环体的执行后再对条件进行判断，当条件不满足的时候就结束循环，若条件满足会再执行循环体。do-while语句如下所示

```
 do {
    //执行语句
    //改变while里面的那个条件
  } while(condition)
```

由语句可见，do-while语句最少执行一次语句块。

5）for循环

for循环是可以通过三段表达式来控制循环条件的。并且也可以在循环体里面改变循环条件。在java5的时候javayou 提供了另一种for循环形式。这两种方式如下所示：

例1

```
for (初始值; 判断条件; 状态改变) {
  //循环体
}
```

例2

```
for (循环变量类型 循环变量名称 : 要被遍历的对象) {
  //循环体
}
```

例1:是最开始就支持的写法，在括号内写了一个三段表达式，分别是初始条件，条件判断，还有状态的改变，初始条件就是你循环的开始，首先计算机会计算这个值，然后再和第二个表达式进行对比如果返回true就进入循环体，如果是false就直接退出循环。当循环体执行完后会计算第三个值，计算完后就会再和第二个值进行判断，如果是true就进入循环体……。

例2也叫foreach语句，他在遍历数组、集合方面，为开发人员提供了极大的方便。这种方式是完全顺序遍历，是不允许进行改变状态的（退出等操作是可以的）。foreach仅仅老老实实地遍历数组或者集合一遍。

在循环体中也可以用break，或者continue两个关键字控制循环体执行过程的。break同switch可以直接结束循环，跳出语句执行循环语句后边的语句。而continue是跳过本次循环然后执行下

次循环，也就是说当遇到contiune的时候就不会继续执行了直接会计算下一次判断是否为true然后执行下一次的循环。