# Java笔记

1. 基本类型：  
   程序的基本功能是处理数据  
   程序用变量来表示数据；  
   程序中必须能使用；  
   定义变量是指设定变量的数据类型和变量的名字，定义变量的基本语法为：  
   数据类型 变量名；
2. Java语言把数据类型分为基本类型和引用类型。

boolean  布尔类型      1位  
byte(8位)<short=char(16位)<int=float(32位)<double=long(64位)  
窄范围转换成宽范围,自动转换；宽范围转换成窄范围,需要强制类型转换当小范围与大范围进行比较时,jvm会先将小范围类型；转换成大范围类型在进行比较.

instanceof:  判断一个引用类型所引用的对象是否是一个类的实例。该操作符左边是一个引用类型，右边是一个类名或接口名。形式如下：  
Student  stu  =  new  Student();  
obj  instanceof  ClassName  
stu  instanceof  Student;  
stu  instanceof  Object;  
stu  instanceof  Teacher;

类型在转换之前,只有instanceof比较的对象返回为ture才能进行类型转换。否则报错类型转换异常或者  
obj  instanceof  InterfaceName

 基本类型比较  ＝＝  比较内容  
 引用类型比较  ＝＝  比较地址

基本类型比较(＝＝比较内容),先将小范围数据类型  
转换成大范围数据类型再进行比较  
既可以是基本类型，也可以是引用类型

OOP中的基本概念  
        Java的编程语言是面向对象的，采用这种语言进行编程称为面向对象编程(Object-Oriented  Programming,  OOP)，  它允许  
        设计者将面向对象设计实现为一个可运行的系统。Java的编程单位是类，对象最后要通过类进行实例化(即“创建”)。

面向对象编程有三个特性：  
.封装：以前我讲过java是以类为基础的，所有的属性和方法都是封装在类中的，不像C++在类外还可以定义函数。  
.多态：表面看是多种状态的意思。同一个行为不同人执行起来不相同  
.继承：不容置疑，从父亲那里继承什么家产、金钱或者产业什么的

抽象数据类型  
在C++中，我们可以用struct  来表示一个类，不了解也不要紧。  
在java中，我们用class  这个关键字来表示一个类，类是一个抽象的数据类型

类和对象  
面向对象的开发方法把软件系统看成各种对象的集合，对象就是最小的子系统，一组相关的对象能够组合成更复杂的子系统。面向对象的开发方法将软件系统看成各种对象的集合，接近人的自然思维方式。  
对象是对问题领域中事件的抽象；类是一组具有相同属性和行为对象的模板。面向对象编程的主要任务就是定义对象模型中的各个类。        
 1)类是一种类型：是引用类型；  
 2)类是元数据：描述数据的数据，数据在面向对象领域里以对象的形式存在，类是对象共有属性和方法的抽象描述。

定义方法形式

修饰符  返回类型  方法名（参数列表  可有可无）  
异常抛出类型(可有可无)  ｛代码块｝

必须有返回类型，如果方法没有返回值，必须用void申明返回类型。  
构造器没有返回类型  
方法中定义的参数我们通常叫做形参，调用有参数的方法时，我们通常会传递一些实参给方法

参数传递  
参数传递分为两种：  
1)对于基本数据类型，参数通过值传递。  
基本类型(包括string类型)作为参数传递时，传递的是这个值的拷贝。无论你怎么改变这个拷贝，原值是不会改变的  
2)对于类类型，参数通过引用(对象的引用)传递。  
对象类型作为参数传递时，传递的是对象的引用，也就是对象的地址  
3)只有引用传递的内容能被改变，而按值传递不会变化。  
无论是基本类型作为参数传递，还是对象作为参数传递，实际上传递的都是值，只是值的的形式不用而已

this关键字  
1.当成员变量和局部变量重名时，在方法中使用this时，表示的是该方法所在类中的成员变量。（this是当前对象自己）2.在构造函数中，通过this可以调用同一类中别的构造函数

构建无参构造器,产生多参构造器的结果.  
值得注意的是：    
1：在构造调用另一个构造函数，调用动作必须置于最起始的位置。    
2：不能在构造函数以外的任何函数内调用构造函数。    
3：在一个构造函数内只能调用一个构造函数

static修饰符  
static修饰的方法或者属性,可以使用类.方法或者类.属性访问.

还可以使用引用.方法或者引用.属性  但是不建议使用引用.的方式访问  
static修饰的方法或者代码块只能操作static修饰的变量  
一般变量不能直接被拿来使用,必须通过引用.属性来获得.  
 static修饰符可以用来修饰类的成员变量、成员方法和代码块。  
.  用static修饰的成员变量表示静态变量，可以直接通过类名来访问；  
.  用static修饰的成员方法表示静态方法，可以直接通过类名来访问；  
.  用static修饰的程序代码表示静态代码块，当Java虚似机加载类时，就会执行该代码块；  
被static所修饰的成员变量和成员方法表明归某个类所有，它不依赖于类的特定实例，被类的所有实例共享。只要这个类被加载，Java虚拟机就能根据类名在运行时数据区的方法区内定位到它们。  
1.static  变量  
成员变量：定义在类里面、方法外面的变量,  分两种：  
a.实例变量;  
b.静态变量；形式和实例变量类似，在实例变量前面加static关键字；  
  
static变量和实例变量的区别：  
 .static变量对于每个类而言在内存中只有一个，能被类的所有实例所共享；实例变量对于每个类的每个实例都有一份，它们之间互不影响；  
.Java虚拟机在加载类的过程中为static变量分配内存，实例变量在加载完类后创建对象时分配内存；  
static变量存在方法区，实例变量存在堆区；  
 .static变量可以直接通过类名访问，实例变量通过引用类型变量访问；

3.static方法  
成员方法分为静态方法和实例方法。用static修饰的方法叫静态方法，或类方法。静态方法也和静态变量一样，不需要创建类的实例，可以直接通过类名来访问。static方法  main可以直接访问所属类的引用的实例变量和类的引用的实例方法，直接访问所属类的静态变量和静态方法；  
注：1)不能使用this关键字，super关键字  
2)静态方法必须被实现。静态方法用来表示某个类所特有的功能，这种功能的实现不依赖于类的具体实例，也不依赖于它的子类。既然如此，当前类必须为静态方法提供实现。  
父类的静态方法不能被子类覆为非静态方法

子类可以定义与父类的静态方法同名的静态方法，以便在子类中隐藏父类的静态方法。

异常的基本概念  
1.  异常产生的条件或者称为异常情况。

1. 整数相除运算中，分母为0;    
   b.  通过一个没有指向任何具体对象的引用去访问对象的方法；  
   c.  使用数组长度作为下标访问数组元素；  
   d.  将一个引用强制转化成不相干的对象；  
   2.  异常会改变正常程序流程;  
   异常产生后，正常的程序流程被打破了，要么程序中止，要么程序被转向异常处理(catch)的语句；  
   throw  new  Exception():如果throw抛出的是exception异常会报错,必须处理  
   如果其他exception子类型,不会显示报错.但是任然需要处理.否则运行出错  
   3.  当一个异常的事件发生后，该异常被虚拟机封装形成异常对象抛出。  
   4.  用来负责处理异常的代码被称为异常处理器  
   5.  通过异常处理器来捕获异常

try...catch语句  
在Java语言中，用try...catch语句来捕获处理异常。格式如下：  
  
                        try  {  
                                可能会出现异常情况的代码；  
                        }  catch(异常类型  异常参数)  {  
                                异常处理代码  
                        }  catch(异常类型  异常参数)  {  
                                异常处理代码  
                        }  
程序如果遇到异常,可以使用两种方式进行处理  
1.throws  抛出异常    实际并没有处理异常    
    如果一致没人处理,最后交付给虚拟机处理  
2.try…catch  处理异常,可能产生异常的代码位于try语句中  
    处理的内容位于catch代码块中.在catch代码块中  
可以使用getMessage()来捕获throw里面抛出的  
自定义内容  
catch语句块中  
1.可以有多个Exception，但是不是每一个都执行.try中出现了那一个  
Exception,它才进入到哪一个catch的exception中进行处理  
2.不允许出现相同Exception  
3.Exception出现的位置是有顺序的,只能越往下范围越大,不能  
越往下范围越小,允许相同等级.  
  
语句执行顺序  
 1.  如果try代码块中没有抛出异常，try代码块中语句会顺序执行完，catch代码块内容不会被执行；  
2.  如果try代码块中抛出catch代码块所声明的异常类型对象，程序跳过try代码块中接下来代码，直接执行catch代码块中对

什么是线程

进程是指运行中的应用程序，每一个进程都有自己独立的内存空间。一个应用程序可以同时启动多个进程。线程就是轻量级的进程。

线程是指进程中的一个执行流程。一个进程可以由多个线程组件。即在一个进程中可以同时运行多个不同的线程，它们分别执行不同的任务，当进程内的多个线程同时运行时，这种运行方式称为并发运行。  
线程与进程的主要区别在于：每个进程都需要操作系统为其分配独立的内存地址空间，而同一进程中的所有线程在同一块地址空间中工作，这些线程可以共享同一块内存和系统资源。比如共享一个对象或者共享已经打开的一个文件。

主线程  
在java虚拟机进程中，执行程序代码的任务是由线程来完成的。每当用java命令启动一个Java虚拟机进程时，Java虚拟机都会创建一个主线程。该线程从程序入口main()方法开始执行。

计算机中机器指令的真正执行者是CPU，线程必须获得CPU的使用权，才能执行一条指令。

线程的创建和启动：从启动类的main()方法开始运行

创建线程有两种方式，如下：  
getName():返回当前调用的线程的名称      
名称默认Thread-0    表示第一个线程    
0这个数字以此类推,表示多个线程  
.  继承java.lang.Thread类;  
.  实现Runnable接口;  
1.    扩展java.lang.Thread类  
Thread类代表线程类，它的最主要的两个方法是：  
.  run()——包含线程运行时所执行的代码；  
.  start()——用于启动线程；  
public  void  run();        //没有抛异常，所以子类重写亦不能抛异常  
1)  主线程与用户自定义的线程并发运行  
a.  Thread类的run()方法是专门被自身的线程执行的，主线程调用Thread类的run()方法，违背了Thread类提供run()方法的初衷；  
b.  Thread  thread  =  Thread.currentThread(); 返回当前正在执行这行代码的线程引用；  
String  name  =  thread.getName();  获得线程名字；  
每个线程都有默认名字，主线程默认的名字为main,  用户创建的第一个线程的默认名字为"Thread-0"，  第二个线程的默认名字为"Thread-1"，  依引类推。Thread类的setName()方法可以显示地设置线程的名字；  
2)  多个线程共享同一个对象的静态变量  
      多个线程各自操作自己的实例变量        
      继承Thread只有操作静态变量时才是共享数据,实例变量不是共享数据  
      实现Runnable操作实例变量就是共享数据(只创建一个runnable对象)  
      多线程多用实现Runnable,少用继承Thread

1. 不要随便覆盖Thread类的start()方法  
   创建了一个线程对象，线程并不自动开始运行，必须调用它的start()方法。对于以下代码：  
   Machine  machine  =  new  Machine();  
    machine.start();  
   当用new语句创建Machine对象时，仅仅在堆区内出现一个包含实例变量Machine对象，此时Machine线程并没有被启动。当主线程执行Machine对象的start()方法时，该方法会启动Machine线程，在Java栈区为它创建相应的方法调用栈。  
   4)  一个线程只能被启动一次  
   Machine  machine  =  new  Machine();  
   machine.start();//抛出IllegalThreadStateException异常  运行时异常
2. 5)主线程也有可能在子线程结束之前结束。并且子线程不受影响，不会因为主线程的结束而结束。  
   2.    实现Runnable接口  
   相同条件下,一般使用Runnable来实现多线程的构建  java支持单继承多实现.  
   Java不允许一个类继承多个类，因此一旦一个类继承了Thread类，就不能再继承其他的类。为了解决这一问题，Java提供了java.lang.Runnable接口，它有一个run()方法，定义如下：  
   public  void  run();  
   启动：Thread(Runnable  runnable)    //当线程启动时，将执行参数runnable所引用对象的run()方法；

线程状态  
线程在它的生命周期中会处于各种不同的状态

流的概念

程序的主要任务是操纵数据。在Java中，把一组有序的数据序列称为流。根据操作的方向，可以把流分为输入流和输出流两种。程序从输入流读取数据，向输出流写出数据。

文件       输入流    输出流        文件

内存  ------------->    Java程序  ------------------>  内存

键盘 控制台

数据源 数据目的地

java  I/O系统负责处理程序的输入和输出，I/O类库位于java.io包中，它对各种常见的输入流和输出流进行了抽象。  
如果数据流中最小的数据单元是字节，那么称这种流为字节流；如果数据流中最小的数据单元是字符，那么称这种流为字符流。在I/O类库中，java.io.InputStream和java.io.OutputStream分别表示字节输入流和字节输出流，  
java.io.Reader和java.io.Writer分别表示字符输入流和字符输出流。在java.io包中，java.io.InputStream表示字节输入流，java.io.OutputStream表示字节输出流，它们都是抽象类，不能被实例化。  
InputStream类提供了一系列和读取数据有关的方法：文件中的数据都是string类型  
  1.  read():  从输入流读取数据,返回一个字母的ascii值：有三种重载形式：　  
              当读不到数据时方法就返回-1，你可以理解为是一个错误信息，提示没有数据可读  
              ascii没有负数,如果有值相当于read()返回值>=0.read()返回的值时整数  
              如果此时(!=-1  表示  >=0)表示有数据  
                              a.  int  read():  从输入流读取一个8位的字节，把它转换为0-255之间的整数，并返回这一整数。例如，如果读到的  
字节为9，则返回9，如果读到的字节为-9，则返回247。如果遇到输入流的结尾，则返回-1;  
       b.  int  read(byte[]  b):  从输入流读取若干个字节，把它们保存到参数b指定的字节数组中。返回的整数表示读取的字节数。如果遇到输入流的结尾，则返回-1;  
      返回的正数,表示一次读取字节数.使用byte[]数组可以降低物理读取次数  
      byte[]长度最大可以取到整个要读取的数据的长度  
      需要注意,数组设定的长度如果不能一次读取完内容.第二次读取时会先将剩下的内容放入到数组中(从第一个位置开始替换直到所有数据都进来,如果数组中还有剩余空间,后面没有替换的不受影响)  
c.  int  read(byte[]  b,  int  off,  int  len):  从输入流读取若干个字节，把它们保存到参数b指定的字节数组中。返回的整数表示读取的字节数。参数off指定在字节数组中开始保存数据的起始下标(默认为0)，参数len指定读取的字节数目。返回的整数表示实现读取的字节数。如果遇到输入流的结尾，则返回-1;需要注意测试如果继续打印保存在数组中的内容,len后面的内容第一次没有读取到.会将他们覆盖到源来的数组中,后面没有覆盖的不影响.len+off的和如果大于数组的长度会抛出数组越界错误  
2.  void  close():  关闭输入流，InputStream类本身的close()方法不执行任何操作。它的一些子类覆盖了close()方法， 在close()方法中释放和流有关的系统资源。  
3.  int  available():  返回可以从输入流中读取的字节数目；此方法是返回这个流中有多少个字节数，可以把数组长度定为这个  
4.  skip(long  n):  从输入流中跳过参数n指定数目的字节.  
5.  boolean  markSupported()，void  mark(int)，void  reset():  
mark就像书签一样，在这个BufferedReader对应的buffer里作个标记，以后再调用reset时就可以再回到这个mark过的地方。mark方法有个参数，通过这个整型参数，你告诉系统，希望在读出这么多个字符之前，这个mark保持有效。读过这么多字符之后，系统可以使mark不再有效，  
需要注意还和buffer的缓冲区大小有关.  
BufferedInputStream类调用mark(int  readlimit)方法后读取多少字节标记才失效，是取readlimit和BufferedInputStream类的缓冲区大小两者中的最大值，而并非完全由readlimit确定。  
1.  write():  向输出流写入数据：有三种重载形式：  
a.  void  write(int  b):　向输出流写入一个字节；read()  
b.  void  write(byte[]  b):  把参数b指定的字节数组中的所有字节写到输出流；read(byte[])  
c.  void  write(byte[]  b,  int  off,  int  len):  把参数b指定的字节数组中的所有字节写到输出流，参数off指定字节数组的起始下标，从这个位置开始输出由参数len指定数目的字节；

IP地址：  
1）IP网络中每台主机都必须有一个惟一的IP地址；本机ip  127.0.0.1(本机地址)  
2）IP地址是一个逻辑地址；动态改变    电脑的唯一标示：物理地址  
3）因特网上的IP地址具有全球唯一性；  
4）32位，4个字节，常用点分十进制的格式表示，例如：192.168.0.16。

端口号：  
端口使用一个16位的数字来表示，它的范围是0--65535，1024以下的端口号保留给预定义的服务。例如：http使用80端口。  
ip:查找主机  
port:查找主机上的对应服务的端口号

基于TCP的Socket编程步骤：  
  
1）服务器程序编写：  
        ①调用ServerSocket(int  port)创建一个服务器端套接字，并绑定到指定端口上；  
        ②调用accept()，监听连接请求，如果客户端请求连接，则接受连接，返回通信套接字；  
        ③调用Socket类的getOutputStream()和getInputStream获取输出流和输入流，开始网络数据的发送和接收；  
        ④最后关闭通信套接字。  
2）客户端程序编写：  
        ①调用Socket()创建一个流套接字，并连接到服务器端；  Socket(ip,port)  
        ②调用Socket类的getOutputStream()和getInputStream获取输出流和输入流，开始网络数据的发送和接收；    
        ③最后关闭通信套接字。