多线程与socket ，可见性，this，变量的详解，值传递与引用 的一部分记载

线程常用方法

1.获取线程名称：getName();

2.取得当前线程对象：currentThread();

3.判断是否启动：isAlive();

4.强行运行：join();

5.线程休眠：sleep();

6.线程礼让：yield();

synchronized 线程的互斥
wait/notifyAll
volatile &fianl

synchronized是Java中的关键字，是一种同步锁。它修饰的对象有以下几种：
1. 修饰一个代码块，被修饰的代码块称为同步语句块，其作用的范围是大括号{}括起来的代码，作用的对象是调用这个代码块的对象；
2. 修饰一个方法，被修饰的方法称为同步方法，其作用的范围是整个方法，作用的对象是调用这个方法的对象；
3. 修改一个静态的方法，其作用的范围是整个静态方法，作用的对象是这个类的所有对象；
4. 修改一个类，其作用的范围是synchronized后面括号括起来的部分，作用主的对象是这个类的所有对象。

wait方法，是等待，放入setwait中

Interrupt方法 不是用来停止线程的
，Thread.interrupt()方法不会中断一个正在运行的线程。这一方法实际上完成的是，在线程受到阻塞时抛出一个中断信号，这样线程就得以退出阻塞的状态。更确切的说，如果线程被Object.wait, Thread.join和Thread.sleep三种方法之一阻塞，那么，它将接收到一个中断异常（InterruptedException），从而提早地终结被阻塞状态。
正确的停止线程方式是设置共享变量，并调用interrupt()（注意变量应该先设置）。
如果线程没有被阻塞，这时调用interrupt()将不起作用；
否则，线程就将得到异常（该线程必须事先预备好处理此状况），接着逃离阻塞状态。

还有一个错误的方法 stop方法，是以前错误的遗留，不再提及

以下是socket

http:80 ftp:21 telent:23 这是端口
java提供的网络功能有四大类
1 InetAddress ：用于标识网络的硬件资源
2 URL :统一资源定位符 通过url可以直接读取活写入网络上的数据
3 Sockets :使用Tcp协议实现网络通信的Socket相关的类
4 Datagam: 使用UDP协议，将数据保存在数据报中，通过网络进行通信
TCP编程
为SeverThread
Socket
SocketSevice
UDP编程
URL统一资源定位符
协议名称和资源名称，中间由冒号隔开
?传参数#锚点
各种get
openStream()方法
"utf-8"改变编码是很容易的事情
如果我们没有指定端口号，则使用默认的端口号，返回给我们的就是 -1 这 个是URL的getPort方法（ ）的特性

通过DatagramSocket
DatagramPacket 这俩个类来进行传输，DatagramPacket里放的是数据
DatagramSocket这是进行通讯的地方
在相应的API文档中有说明

TCP（Transmission Control Protocol，传输控制协议）是基于连接的协议，也就是说
，在正式收发数据前，必须和对方建立可靠的连接。一个TCP连接必须要经过三次“对话”才能建立起来，
其中的过程非常复杂，我们这里只做简单、形象的介绍，你只要做到能够理解这个过程即可。
我们来看看这三次对话的简单过程：主机A向主机B发出连接请求数据包：“我想给你发数据，可以吗？”，
这是第一次对话；主机B向主机A发送同意连接和要求同步（同步就是两台主机一个在发送，一个在接收，协调工作）的数据包：
“可以，你什么时候发？”，这是第二次对话；主机A再发出一个数据包确认主机B的要求同步：“我现在就发，你接着吧！”，
这是第三次对话。三次“对话”的目的是使数据包的发送和接收同步，经过三次“对话”之后，主机A才向主机B正式发送数据。
这是一个全双工的、面向连接的、可靠的并且是精确控制的协议。

UDP（User Data Protocol，用户数据报协议）是与TCP相对应的协议。它是面向非连接的协议，它不与对方建立连接，而是直接就把数据包发送过去！
UDP适用于一次只传送少量数据、对可靠性要求不高的应用环境。
比如，我们经常使用“ping”命令来测试两台主机之间TCP/IP通信是否正常，
其实“ping”命令的原理就是向对方主机发送UDP数据包，然后对方主机确认收到数据包，
如果数据包是否到达的消息及时反馈回来，那么网络就是通的。例如，在默认状态下，一次“ping”操作发送4个数据包（如图2所示）。
大家可以看到，发送的数据包数量是4包，收到的也是4包（因为对方主机收到后会发回一个确认收到的数据包）。这充分说明了UDP协议是面向非连接的协议，
没有建立连接的过程。正因为UDP协议没有连接的过程，所以它的通信效果高；但也正因为如此，它的可靠性不如TCP协议高。QQ就使用UDP发消息，
因此有时会出现收不到消息的情况。
因此UDP适合用在那些实时性强、允许出错的场合。

TCP协议和UDP协议的差别
TCP UDP
是否连接 面向连接 面向非连接
传输可靠性 可靠 不可靠
应用场合 传输大量数据 少量数据
速度 慢 快

以下是可见性
可见性是什么？

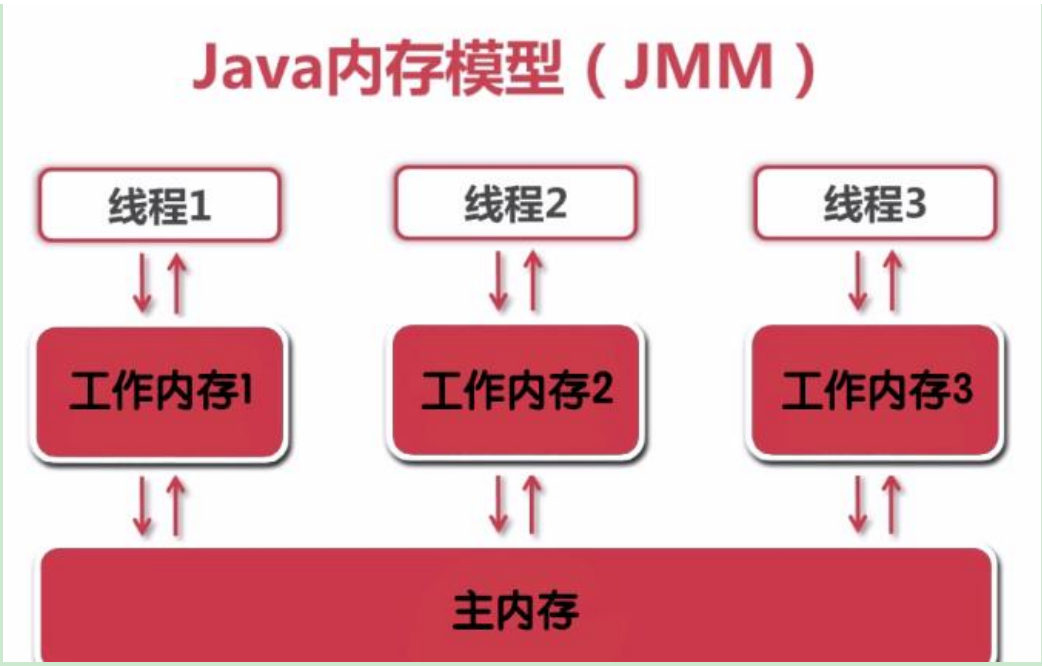
一个线程对共享变量值的修改，能够及时地被其他线程看到

共享变量，如果一个变量在多个线程的工作内存中都存在副本，那么这个变量就是这几个线程的共享变量

JMM

Java内存模型（Java Memory Model）描述了Java程序中各种变量（线程共享变量）的访问规则，以及在JVM中将变量存储到内存和从内存中读取出变量这样的底层细节。

所有的变量都存储在主内存中
每个线程都有自己独立的工作内存，里面保存该线程使用到的变量的副本（主内存中该变量的一份拷贝）
线程对共享变量的操作必须在自己的工作内存中进行，
不能直接从主内存中读写
这边还是粘贴一下关于JMM的图吧



：
线程对共享变量的所有操作都必须在自己的工作内存中进行，不能直接从主内存中读写
不同线程直接无法直接访问其他线程工作内存中的变量，线程间变量值的传递需要从主内存来完成

java语言层面支持的可见性实现方式
synchronized
http://www.cnblogs.com/GnagWang/archive/2011/02/27/1966606.html
关于synchrized的具体讲解
特性：可见性，原子性
线程解锁前，必须把共享变量的最新值刷新到主内存中
线程加锁时，将清空工作内存中共享变量的值，从而使用共享变量时
需要从主内存重新读取最新的值（注意：加锁与解锁是同一把锁）
可以得到
线程解锁前对共享变量的修改在下次加锁时对其他线程可见
1 获得互斥锁
2 清空工作内存
3 从主内存拷贝变量最新的副本到工作内存
4 执行代码
5 将更改后的共享变量的值刷新到主内存
6 释放互斥锁

重排序
1 编译器进行的优化(编译器优化)
2 指令级进行的优化（处理器优化）
3 内存系统的优化(处理器优化)

as-if-serial
无论如何重排序，程序执行的结果应该与代码顺序执行的结果一致（java编译器。运行时和处理器都会保证java在单线程下遵循as-if-serial语义）
导致**共享变量在线程间不可见**的原因：
1，线程的交叉执行
2 重排序结合线程交叉执行
3 共享变量更新后的值没有在工作内存与主内存间及时更新

只有数据依赖关系才能禁止重排序
synchroized对这个的解决方法
1 原子性
2 可见性

1. synchronized包裹代码块：
   I . synchronized(对象){}
   II . synchronized(类名.class){}
   III. synchronized(this){}
2. synchronized修饰方法：
   I .public synchronized void memberMethod(){};
   II.public static synchronized void staticMethod(){};

volaile可见性

volaile如何实现内存可见性
深入来说：通过**加入内存屏障和禁止重排序优化来实现**的
根上还是通过保证多线程中的共享变量在其工作内存空间的值是一致的。
线程写volaile变量的过程
（1）改变变成工作内存中volatile变量副本的值
（2）将改变后的副本的值从工作内存刷新到主内存
线程读volatile变量的过程：
（1）从主内存中读取volatile变量的最新值到线程的工作内存中
（2）从工作内存中读取volatile变量的副本

不能保证volatile变量复合操作的原子性
原子性：最小的操作单元，只能一个线程来完成此最小单元中的所有操作后其他的线程才能操作。

volatile适合场所
要在多线程中安全的使用volatile变量，必须满足
1 对变量的写入操作不依赖当前值
不满足变量：number++。count=count*5
满足：boolean变量。记录温度变化的变量等
2 该变量没有包含在具有其他变量的不变式中

synchronized和volatile这两种保证多线程中共享变量可见的方式的比较如下，结论是：
1：因为volatile消耗更小，所以，能满足时推荐使用
2：当volatile处理起来比较麻烦，只能牺牲点效率了——使用synchronized

对64位（long,double）变量的读写可能不是原子操作

以下是final的用法与详解

final的作用随着所修饰的类型而不同
根据程序上下文环境，Java关键字final有“这是无法改变的”或者“终态的”含义

1、final修饰类中的属性或者变量

无论属性是基本类型还是引用类型，final所起的作用都是变量里面存放的“值”不能变。

这个值，对于基本类型来说，变量里面放的就是实实在在的值，如1，“abc”等。

而引用类型变量里面放的是个地址，所以用final修饰引用类型变量指的是它里面的地址不能变，并不是说这个地址所指向的对象或数组的内容不可以变，这个一定要注意。

例如：类中有一个属性是final Person p=new Person("name")；那么你不能对p进行重新赋值，但是可以改变p里面属性的值，p.setName('newName');

final修饰属性，声明变量时可以不赋值，而且一旦赋值就不能被修改了。对final属性可以在三个地方赋值：声明时、初始化块中、构造方法中。总之一定要赋值。

2、final修饰类中的方法

作用：可以被继承，但继承后不能被重写。

3、final修饰类

作用：类不可以被继承。

思考一个有趣的现象：

```
byte b1=1;

byte b2=3;

byte b3=b1+b2;//当程序执行到这一行的时候会出错，因为b1、b2可以自动转换成int类型的变量，运算时java虚拟机对它进行了转换，结果导致把一个int赋值给byte-----出错

如果对b1 b2加上final就不会出错

final byte b1=1;
```

final byte b2=3;

byte b3=b1+b2;//不会出错，相信你看了上面的解释就知道原因了。

使用final方法的原因有二：

第一、把方法锁定，防止任何继承类修改它的意义和实现。

第二、高效。编译器在遇到调用final方法时候会转入内嵌机制，大大提高执行效率

用final修饰的成员变量表示常量，值一旦给定就无法改变！

final修饰的变量有三种：静态变量、实例变量和局部变量，分别表示三种类型的常量。

从下面的例子中可以看出，一旦给final变量初值后，值就不能再改变了。

当函数参数为final类型时，你可以读取使用该参数，但是无法改变该参数的值

关于final的重要知识点

1. final关键字可以用于成员变量、本地变量、方法以及类。
2. final成员变量必须在声明的时候初始化或者在构造器中初始化，否则就会报编译错误。
3. 你不能够对final变量再次赋值。
4. 本地变量必须在声明时赋值。
5. 在匿名类中所有变量都必须是final变量。
6. final方法不能被重写。
7. final类不能被继承。
8. final关键字不同于finally关键字，后者用于异常处理。
9. final关键字容易与finalize()方法搞混，后者是在Object类中定义的方法，是在垃圾回收之前被JVM调用的方法。
10. 接口中声明的所有变量本身是final的。
11. final和abstract这两个关键字是反相关的，final类就不可能是abstract的。
12. final方法在编译阶段绑定，称为静态绑定(static binding)。
13. 没有在声明时初始化final变量的称为空白final变量(blank final variable)，它们必须在构造器中初始化，或者调用this()初始化。不这么做的话，编译器会报错“final变量(变量名)需要进行初始化”。
14. 将类、方法、变量声明为final能够提高性能，这样JVM就有机会进行估计，然后优化。
15. 按照Java代码惯例，final变量就是常量，而且通常常量名要大写：

```
1  privatefinalintCOUNT = 10;
```

1. 对于集合对象声明为final指的是引用不能被更改，但是你可以向其中增加，删除或者改变内容。譬如：

```
1  privatefinalList Loans = newArrayList();
2  list.add(“home loan”);  //valid
3  list.add("personal loan"); //valid
4  loans = newVector();  //not valid
```

我们已经知道final变量、final方法以及final类是什么了。必要的时候使用final，能写出更快、更好的代码的。

关于变量的说明

成员变量（实例变量）
在类中定义 ，用来描述对象将有什么
局部变量
在类的方法中对应，在方法中临时保存数据
局部变量的作用域仅限于定义它的方法
成员变量的作用域在整个类内部都是可见的
java会给成员变量一个初值
java不会给局部变量赋予初始值
在同一个方法中，不允许有同名局部变量
在不同方法中，可以有同名局部变量
优先调用局部变量
构造方法也是方法，只不过是特殊的

关于实例变量 类变量 局部变量的区分
http://damoqingquan.iteye.com/blog/234737

this的详解

this对象就表示当前变量的属性
什么是对象 万物皆对象，客观存在的事物皆为对象
类就是对象的类型
具有相同属性和方法的一组对象的集合
类是模子，确定对象将会拥有的特征（ ）和行为（方法）
属性--对象具有的各种特征
每个对象的每个属性都拥有特定值

类是抽象的概率，仅仅是模板，比如说：“手机”对象是一个你能够看到的，摸到的具体实体
类的重要性： 所有java属性都以类class为组织单元
什么是类？
public class 类名{
//定义属性部分（成员变量
）属性1的类型 属性1
属性2的类型 属性2
。
。
。
属性n的类型 属性n
定义方法部分
方法一；
方法二
。
。
。
方法n
创建对象：
类名 对象=new 类名（）；
Telephone phone =new Telphone（）；
使用对象
引用对象的属性：对象名：属性
phone.screen=5;//给screen属性赋值5
引用对象的方法：对象名.方法名()
phone.sendMessage();//调用sendMessage()方法

java中的值传递与引用传递
https://www.zhihu.com/question/31203609
http://blog.csdn.net/ithomer/article/details/6906797