————————————————语言基础———————————————————

**九种基本数据类型的大小，以及他们的封装类，占用大小。**

Int Integer 4byte 0

char Character 2bye [] 空字符

long Long 8byte 0

double Double 8byte 0.0

float Float 4byte 0.0

bool Boolean (jvm虽然定义了boolean这种数据类型，但是只对它提供了非常有限的支持。在Java虚拟机中没有任何供boolean值专用的字节码指令 4byte或1byte) false

byte Byte 1byte 0

short Short 2byte 0

void Void

类的成员是基本类型（封装类是类不行）会有一个初始值

局部变量使用前必须初始化

**Switch能否用string做参数？**

在Java5以前，switch(expr)中，exper只能是byte，short，char，int类型。从Java5开始，java中引入了枚举类型，即enum类型。从Java7开始，exper还可以是String类型。

**equals与==的区别**

==比的是引用

Equals调用的是equals方法，如果equals没被声明则会调用父类的equals方法，object的equals比较的是引用

**Object有哪些公用方法？**

protected Object clone() 克隆一个对象

boolean equals(Object obj) 判等

protected void finalize() 垃圾回收器回收对象空间时调用

Class<?> getClass() 得到类的引用

int hashCode() 类的hash值

void notify()

void notifyAll()

String toString()

void wait()

void wait(long timeout)

void wait(long timeout, int nanos)

**Java的四种引用，强弱软虚，用到的场景。**

强引用 以前我们使用的大部分引用实际上都是强引用，这是使用最普遍的引用。

Counter counter = new Counter(); // strong reference

WeakReference<Counter> weakCounter = new **WeakReference**<Counter>(counter); //weak reference//weakCounter弱引用

SoftReference soft= new SoftReference(counter) ; //soft reference

ReferenceQueue refQueue = new ReferenceQueue(); //引用会在队列中等待被清楚

PhantomReference<Counter> phantom = new PhantomReference<Counter>( counter, refQueue);//虚引用

强引用是使用最普遍的引用。如果一个对象具有强引用，那垃圾回收器绝不会回收它。

如果一个对象只具有软引用，则内存空间足够，垃圾回收器就不会回收它；

在垃圾回收器线程扫描它所管辖的内存区域的过程中，一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存。

如果一个对象仅持有虚引用，那么它就和没有任何引用一样，在任何时候都可能被垃圾回收器回收。虚引用主要用来跟踪对象被垃圾回收器回收的活动。

**Hashcode的作用。**

hashCode是用于查找使用的（根据hashcode得到存储位置），而equals是用于比较两个对象的是否相等的

1、hashCode的存在主要是用于查找的快捷性，如Hashtable，HashMap等，hashCode是用来在散列存储结构中确定对象的存储地址的；

2、如果两个对象相同，就是适用于equals([Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank).lang.Object) 方法，那么这两个对象的hashCode一定要相同；

3、如果对象的equals方法被重写，那么对象的hashCode也尽量重写，并且产生hashCode使用的对象，一定要和equals方法中使用的一致，否则就会违反上面提到的第2点；

4、两个对象的hashCode相同，并不一定表示两个对象就相同，也就是不一定适用于equals(java.lang.Object) 方法。

**参数签名**

为什么使用内部类

——————————————————Volatile——————————————————