目录

[1. https://www.jianshu.com/p/c70989bd5f29?from=message&isappinstalled=0 1](#_Toc46784358)

[2. java中==和equals和hashCode的区别 1](#_Toc46784359)

[3. int、char、long各占多少字节数 2](#_Toc46784360)

[4. int与integer的区别 2](#_Toc46784361)

[5. 谈谈对java多态的理解 4](#_Toc46784362)

### <https://www.jianshu.com/p/c70989bd5f29?from=message&isappinstalled=0>

### java中==和equals和hashCode的区别

“==”：

　　==是运算符，用来比较两个值、两个对象的内存地址是否相等；

“equals()”：

　　equals是Object类的方法，默认情况下比较两个对象是否是同一个对象，内部实现是通过“==”来实现的。

如果想比较两个对象的其他内容，则可以通过重写equals方法，

例如：String类就重写了equals方法，改成了对象的内容是否相等。

具体实现可以参考：<https://www.cnblogs.com/hujinshui/p/10038044.html>

“hashCode()”：

　　hashCode也是Object类里面的方法，返回值是一个对象的哈希码，同一个对象哈希码一定相等，但不同对象哈希码也有可能相等。

如果两个对象通过equals方法比较相等，那么他的hashCode一定相等；

如果两个对象通过equals方法比较不相等，那么他的hashCode有可能相等；

例如以下程序：

class Untitled {

public static void main(String[] args) {

System.out.println("AaAa".hashCode());//2031744

System.out.println("BBBB".hashCode());//2031744

}

}

### int、char、long各占多少字节数

Java基本类型占用的字节数：  
1字节： byte , boolean  
2字节： short , char  
4字节： int , float  
8字节： long , double

编码与中文：  
Unicode/GBK： 中文2字节  
UTF-8： 中文通常3字节，在拓展B区之后的是4字节  
综上，中文字符在编码中占用的字节数一般是2-4个字节。

C语言中short通常是2字节，具体可能和平台编译器相关。

### int与integer的区别

一、区别：

1.Integer是int的包装类，int则是java的一种基本的数据类型；

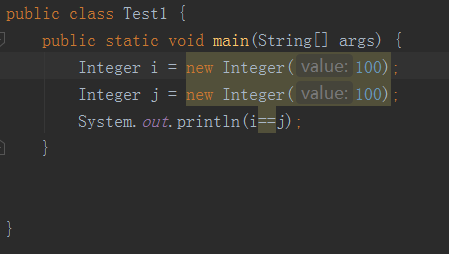
2.Integer变量必须实例化之后才能使用，而int变量不需要实例化；

3.Integer实际是对象的引用，当new一个Integer时，实际上生成一个指针指向对象，而int则直接存储数值

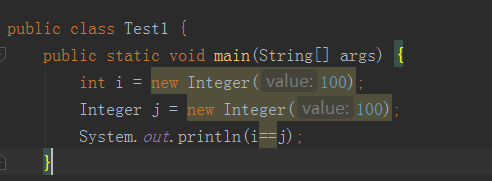
4.Integer的默认值是null，而int的默认值是0。

二、Integer和int的比较

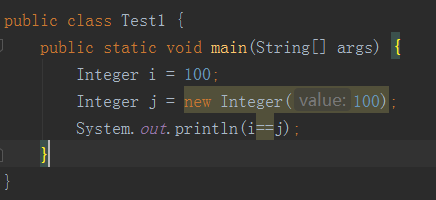
　　1.由于Integer实际是对一个Integer对象的引用，所以两个通过new生成的Integer变量永远是不相同的，因为New生成的是两个不同的对象，其内存地址不同。下面运行的结果为false



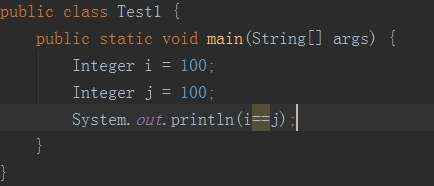
　　2.Integer变量和int变量进行比较时，只要两个变量的值相等，则结果就为True，(因为包装类Integer和基本数据类型比较的时候，java会自动拆箱为int，然后进行比较，实际上就是两个int变量进行比较)，下面运行的结果为true



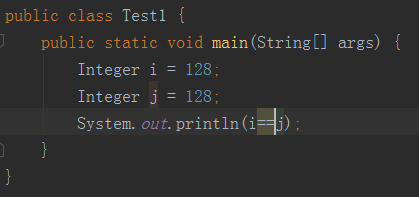
　　3.非new生成的Integer变量和new Integer生成的Integer变量比较的时候，结果为false(因为非new生成的Integer变量指向的是Java常量池中的对象，而new出来的对象指向的是堆中新建的对象，两者内存地址不同)，下面返回的是false



　　4.两个非new出来的Integer对象，进行比较的时候，如果两个变量的值区间在-127~128之间的时候，则返回的结果为true，如果两个变量的变量值不在这个区间，则比较的结果为false。下面返回的是true



　　下面返回的是false



三、java 基本类型与引用类型的区别：

　　1.基本数据类型保存原始值，引用数据类型保存的是引用值(引用值就是指在对象中所处的地理位置)

### 谈谈对java多态的理解

<https://www.cnblogs.com/eyesCentre/archive/2019/04/12/10697487.html>

1.什么是java的多态

浏览了别人博客中的一些介绍多态的文章，发现大家的描述有点不一样，主要区别在于是否把方法的重写算做多态。一种我比较认同的说法如下：

多态分为两种

a. 编译时多态：方法的重载；

b. **运行时多态**：JAVA运行时系统根据调用该方法的实例的类型来决定选择调用哪个方法则被称为运行时多态。（我们平时说得多的事运行时多态，所以多态主要也是指运行时多态）；

上述描述认为重载也是多态的一种表现，不过多态主要指运行时多态。

2.运行时多态

**a. 面向对象的三大特性**：封装、继承、多态。从一定角度来看，封装和继承几乎都是为多态而准备的。这是我们最后一个概念，也是最重要的知识点。

**b. 多态的定义**：指允许不同类的对象对同一消息做出响应。即同一消息可以根据发送对象的不同而采用多种不同的行为方式。（发送消息就是函数调用）

**c. 实现多态的技术称为**：动态绑定（dynamic binding），是指在**执行期间**判断所引用对象的实际类型，根据其实际的类型调用其相应的方法。

**d. 多态的作用**：消除类型之间的耦合关系。

**e. 现实中，关于多态的例子不胜枚举**。比方说按下 F1 键这个动作，如果当前在 Flash 界面下弹出的就是 AS 3 的帮助文档；如果当前在 Word 下弹出的就是 Word 帮助；在 Windows 下弹出的就是 Windows 帮助和支持。同一个事件发生在不同的对象上会产生不同的结果。

下面是多态存在的三个必要条件，要求大家做梦时都能背出来！

多态存在的三个必要条件  
一、要有继承；  
二、要有重写；  
三、父类引用指向子类对象。

**多态的好处**：

1.可替换性（substitutability）。多态对已存在代码具有可替换性。例如，多态对圆Circle类工作，对其他任何圆形几何体，如圆环，也同样工作。  
2.可扩充性（extensibility）。多态对代码具有可扩充性。增加新的子类不影响已存在类的多态性、继承性，以及其他特性的运行和操作。实际上新加子类更容易获得多态功能。例如，在实现了圆锥、半圆锥以及半球体的多态基础上，很容易增添球体类的多态性。  
3.接口性（interface-ability）。多态是超类通过方法签名，向子类提供了一个共同接口，由子类来完善或者覆盖它而实现的。如图8.3 所示。图中超类Shape规定了两个实现多态的接口方法，computeArea()以及computeVolume()。子类，如Circle和Sphere为了实现多态，完善或者覆盖这两个接口方法。  
4.灵活性（flexibility）。它在应用中体现了灵活多样的操作，提高了使用效率。  
5.简化性（simplicity）。多态简化对应用软件的代码编写和修改过程，尤其在处理大量对象的运算和操作时，这个特点尤为突出和重要。

### String、StringBuffer、StringBuilder区别

**String** 类不可变，内部维护的char[] 数组长度不可变，为final修饰，String类也是final修饰，不存在扩容。字符串拼接，截取，都会生成一个新的对象。频繁操作字符串效率低下，因为每次都会生成新的对象。

**StringBuilder** 类内部维护可变长度char[] ， 初始化数组容量为16，存在扩容， 其append拼接字符串方法内部调用System的native方法，进行数组的拷贝，不会重新生成新的StringBuilder对象。

它是非线程安全的字符串操作类， 其每次调用 toString方法而重新生成的String对象，不会共享StringBuilder对象内部的char[]，会进行一次char[]的copy操作。

**StringBuffer** 类内部维护可变长度char[]， 基本上与StringBuilder一致，但其为线程安全的字符串操作类，大部分方法都采用了Synchronized关键字修改，以此来实现在多线程下的操作字符串的安全性。

其toString方法而重新生成的String对象，会共享StringBuffer对象中的toStringCache属性（char[]），但是每次的StringBuffer对象修改，都会置null该属性值。

### 什么是内部类？内部类的作用

**什么是内部类**：  
将一个类定义在另一个类里面或者一个方法里面，这样的类称为内部类。

**内部类的作用：**

1.成员内部类  
成员内部类可以无条件访问外部类的所有成员属性和成员方法（包括private成员和静态成员）。  
当成员内部类拥有和外部类同名的成员变量或者方法时，会发生隐藏现象，即默认情况下访问的是成员内部类的成员。

2.局部内部类  
局部内部类是定义在一个方法或者一个作用域里面的类，它和成员内部类的区别在于局部内部类的访问仅限于方法内或者该作用域内。

3.匿名内部类  
匿名内部类就是没有名字的内部类

4.静态内部类  
指被声明为static的内部类，他可以不依赖内部类而实例，而通常的内部类需要实例化外部类，从而实例化。静态内部类不可以有与外部类有相同的类名。不能访问外部类的普通成员变量，但是可以访问静态成员变量和静态方法（包括私有类型）  
一个 静态内部类去掉static 就是成员内部类，他可以自由的引用外部类的属性和方法，无论是静态还是非静态。但是不可以有静态属性和方法、

**作用**  
1.每个内部类都能独立的继承一个接口的实现，所以无论外部类是否已经继承了某个(接口的)实现，对于内部类都没有影响。内部类使得多继承的解决方案变得完整， 　　  
2.方便将存在一定逻辑关系的类组织在一起，又可以对外界隐藏。 　　  
3.方便编写事件驱动程序 　　  
4.方便编写线程代码