5.什么是不可变对象（immutable object）？Java 中怎么创建一个不可变对象？

  String类被声明成final，所以任何类都不能继承，重载它的方法。

总结：不可变类的特点：

    这个类被声明成final，不可以被继承，也不可以重载它的方法

    这个类的成员变量都是final并且是私有的

    如果成员变量是一个引用类型，是可变的，那么在他的getter()方法中，返回的是对该对象的拷贝

如果一个类被设计成不可变的类 ，那么这个类的实例化对象也是不可变的。

不可变类：当你获得这个类的一个实例引用时，你不可以改变这个实例的内容。

不可变对象指对象一旦被创建，状态就不能再改变。任何修改都会创建一个新的对象，如 String、Integer及其它包装类。可以调用访问器方法（getter），复制对象，或者传递对象，但是不允许任何方法改变这个对象的状态。如果一定要改变这个对象的内容，那就创建一个新的不可变对象内容做相应的修改，返回修改后对象的引用

**不可变对象不是真的"完全不可改变"**

通过反射还是可以修改的。

public static void stringReflection() throws Exception {

  String s = "Hello World";

  System.out.println("s = " + s); //Hello World

  //获取String类中的value字段

  Field valueField = String.class.getDeclaredField("value");

  //改变value属性的访问权限

  valueField.setAccessible(true);

  char[] value = (char[]) valueField.get(s);

  //改变value所引用的数组中的第5个字符

  value[5] = '\_';

  System.out.println("s = " + s); //Hello\_World

6.我们能创建一个包含可变对象的不可变对象吗？

是的，我们是可以创建一个包含可变对象的不可变对象的，你只需要谨慎一点，不要共享可变对象的引用就可以了，如果需要变化时，就返回原对象的一个拷贝。最常见的例子就是对象中包含一个日期对象的引用。

**三.如何创建不可变对象**

　　通常来说，创建不可变类原则有以下几条：

　　**1）所有成员变量必须是private**

**2）最好同时用final修饰(非必须)**

　　3**）不提供能够修改原有对象状态的方法**

最常见的方式是不提供setter方法

如果提供修改方法，需要新创建一个对象，并在新创建的对象上进行修改

　　4**）通过构造器初始化所有成员变量，引用类型的成员变量必须进行深拷贝(deep copy)**

　　5**）getter方法不能对外泄露this引用以及成员变量的引用**

　　**6）最好不允许类被继承(非必须)**

===如何创建一个不可变类？

对于基本类型的成员变量，用final修饰，一旦它被初始化，就不能改变了。而对于引用类型的成员变量，不能改变它的引用。

简述：成员变量是private final类型，初始化用构造函数克隆的方式初始化，没有set方法，get方法使用克隆的方式进行访问。

---类定义：final修饰类，不能够继承

---成员变量：1.成员变量设为私有private final,通过构造器初始化；2.不为成员变量设置set方法;3.如果有可变对象那么初始化的时候必须使用clone的方式进行初始化，新建一个对象初始化，否则会影响到原有的对象。

---方法：1.对于不可变对象中的可变对象的访问方式：比如说有个map,那么如何获取呢？get方法不能够直接返回对象本身，而是通过克隆来返回对象，这样修改原始数据不会影响新建的这个数据。2.同时构造器进行初始化对象的时候必须创建一个新的hashmap来复制构造函数的入参map，就是克隆

目的：比如我有三个属性int,string,hashmap用来初始化了一个不可变对象，那么我获取到对象中的hashmap的引用，获取引用使用hasmap.clone()方法，那么这个引用指向新的数组地址，但是如果hashmap内部有对象引用的话，是不会进行复制的，还是指向了原有的对象。

**14.a = a + b 与 a += b 的区别（**=、\*=、/=，%=**）**

主要的区别是在运算时精度的问题 +=”是java中的一个运算符，而不是两个，所以在运算时 会进行自动类型转换。所以在编译时没有报错。

**short** a=4;

**short** b=5;

a=a+b;//报错 error : cannot convert from int to byte 提示你进行强转为（a= (**short**) (a+b);）

当你换成

a+=b;//会自动向高精度进行数据类型转换。所以上面的程序在执行时编译器没有报错。

二者报错信息一样　所以即使当两个操作数同类型时也可能报错　是因为编译器在进行运算时默认int处理了

（译者注：这个地方应该表述的有误，其实无论 a+b 的值为多少，编译器都会报错，因为 a+b 操作会将 a、b 提升为 int 类型，所以将 int 类型赋值给 byte 就会编译出错）

**.17.int 和 Integer 哪个会占用更多的内存？**

Integer 对象会占用更多的内存。Integer 是一个对象，需要存储对象的元数据。但是 int 是一个原始类型的数据，所以占用的空间更少。

**int和Integer的更多区别：**

除了之前写的通用的[基本数据类型与包装类的区别](https://link.jianshu.com/?t=https%3A%2F%2Famyyanjie.github.io%2F2017%2F10%2F08%2FBrief-Analysis-on-the-Differences-between-Java-Primitives-and-Wrappers%2F)之外,还有以下方面的比较：

(注：对于引用类型变量，==操作符比较的是两个引用是否指向同一个对象；对于基本类型变量，==操作符比较的是两个变量的值是否相等。)

**1.两个通过new出来的Integer变量比较，结果为false。**

/\*\*
\* 比较两个new出来的Integer
\*/
public class Test {
public static void main(String[] args) {
Integer i = new Integer(200);
Integer j = new Integer(200);
System.out.print(i == j);
//输出：false
}
}

Integer变量实际上是对一个Integer对象的引用。当new一个Integer时，实际上是生成一个指针指向此对象，两次new Integer生成的是两个对象，其内存地址不同，所以两个new出来的Integer变量不等。

**2.非new生成的Integer变量与new Integer()生成的变量比较，结果为false。**

/\*\*
\* 比较非new生成的Integer变量与new生成的Integer变量
\*/
public class Test {
public static void main(String[] args) {
Integer i= new Integer(200);
Integer j = 200;
System.out.print(i == j);
//输出：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 Integer()生成的变量指向堆中新建的对象，两者在内存中的地址不同。所以 输出为false。

**3.两个非new生成的Integer对象进行比较，如果两个变量的值在区间[-128,127]之间，比较结果为true；否则，结果为false。**

/\*\*
\* 比较两个非new生成的Integer变量
\*/
public class Test {
public static void main(String[] args) {
Integer i1 = 127;
Integer ji = 127;
System.out.println(i1 == ji);//输出：true
Integer i2 = 128;
Integer j2 = 128;
System.out.println(i2 == j2);//输出：false
}
}

java在编译Integer i1 = 127时，会翻译成Integer i1 = Integer.valueOf(127)

**4. Integer变量(无论是否是new生成的)与int变量比较，只要两个变量的值是相等的，结果都为true。**

/\*\*
\* 比较Integer变量与int变量
\*/
public class Test {
public static void main(String[] args) {
Integer i1 = 200;
Integer i2 = new Integer(200);
int j = 200;
System.out.println(i1 == j);//输出：true
System.out.println(i2 == j);//输出：true
}
}

包装类Integer变量在与基本数据类型int变量比较时，Integer会自动拆包装为int，然后进行比较，实际上就是两个int变量进行比较，值相等，所以为true。

作者：AmorFatiYJ

链接：https://www.jianshu.com/p/08010144d40f

来源：简书

简书著作权归作者所有，任何形式的转载都请联系作者获得授权并注明出处。

浅析Java基础类型与封装类型的区别

 基础类型(Primitives)与封装类型(Wrappers)的区别在哪里

**一、传递方式不同**

封装类是引用类型。

**基本类型（原始数据类型）在传递参数时都是按值传递，而封装类型是按引用传递的**(其实“引用也是按值传递的”，传递的是对象的地址)。由于包装类型都是不可变量，因此没有提供改变它值的方法，增加了对“按引用传递”的理解难度。

int是基本类型，直接存放数值；Integer是类，产生对象时用一个引用指向这个对象。

**二、封装类可以有方法和属性**

封装类可以有方法和属性，利用这些方法和属性来处理数据，如Integer.parseInt(Strings)。**基本数据类型都是final修饰的**，不能继承扩展新的类、新的方法。

**三、默认值不同**

**基本类型跟封装类型的默认值是不一样的**。如int i,i的预设为0；Integer j，j的预设为null,因为封装类产生的是对象，对象默认值为null。

**四、存储位置**

**基本类型在内存中是存储在栈中，引用类型的引用（值的地址）存储在栈中，而实际的对象（值）是存在堆中**。

虽然基本类型在栈上分配内存效率高，但是在堆栈上分配内存可能有内存泄漏的问题。

**基本数据类型的好处就是速度快（不涉及到对象的构造和回收），封装类的目的主要是更好的处理数据之间的转换。**

JDK5.0开始可以自动封包了，**基本数据类型可以自动封装成封装类**。

比如集合List，往里添加对象Object，需要将数字封装成封装类型对象，再存到List中。

|  |  |
| --- | --- |
| 1  2 | List list=new ArreyList();  list.add(new Integer(1)); |

在JDK5.0 以后可以自动封包，简写成

|  |  |
| --- | --- |
|  | List list=new ArrayList();  list.add(1); |

**20.Java 中的构造器链是什么？**

当你从一个构造器中调用另一个构造器，就是Java 中的构造器链。这种情况只在重载了类的构造器的时候才会出现。

链式构造器意味着我们以一种方式编写构造函数，构造函数调用其他构造函数，最终到达统一的构造函数。 这个统一的构造函数称为主构造函数，所有其他构造函数称为辅助构造函数。

主构造函数将是我们创建和初始化对象的单一点，因此我们可以将所有常见行为放入其中。 为了保持一致性，我们只有一个主要的构造函数; 理想情况下，此构造函数将设置所有类字段。 此外，重要的是，如果我们不想对外公开主构造函数，那么主构造函数可以设置成私有的。

使用链式构造器，你不必担心代码重复。 因此，它将使你的代码更简单，更易于维护。 所有构造函数最终都调用主构造函数，这是一个可以共享公共代码的单点创建。

|  |
| --- |
| // Java program to illustrate Constructor Chaining  // within same class Using this() keyword  // and changing order of constructors  **class** Temp  {      // default constructor 1      Temp()      {          System.out.println("default");      }        // parameterized constructor 2      Temp(**int** x)      {          // invokes default constructor  **this**();          System.out.println(x);      }        // parameterized constructor 3      Temp(**int** x, **int** y)      {          // invokes parameterized constructor 2  **this**(5);          System.out.println(x \* y);      }    **public** **static** **void** main(String args[])      {          // invokes parameterized constructor 3  **new** Temp(8, 10);      }  } |

输出：

default
5
80

注意：在示例1中，最后调用默认构造函数，但在示例2中默认构造函数首先被调用。因此，构造函数链中的顺序并不重要。

**29.JRE、JDK、JVM 及 JIT 之间有什么不同？**

JRE 代表 Java 运行时（Java run-time），是运行 Java 引用所必须的。JDK 代表 Java 开发工具（Java development kit），是 Java 程序的开发工具，如 Java 编译器，它也包含 JRE。JVM 代表 Java 虚拟机（Java virtual machine），它的责任是运行 Java 应用。JIT 代表即时编译（Just In Time compilation），当代码执行的次数超过一定的阈值时，会将 Java 字节码转换为本地代码，如，主要的热点代码会被准换为本地代码，这样有利大幅度提高 Java 应用的性能。

**34. “a==b”和”a.equals(b)”有什么区别？**

如果 a 和 b 都是对象，则 a==b 是比较两个对象的引用，只有当 a 和 b 指向的是堆中的同一个对象才会返回 true，而 a.equals(b) 是进行逻辑比较，所以通常需要重写该方法来提供逻辑一致性的比较。例如，String 类重写 equals() 方法，所以可以用于两个不同对象，但是包含的字母相同的比较。