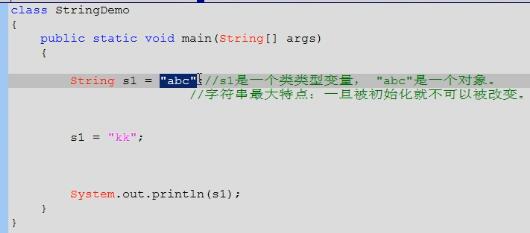
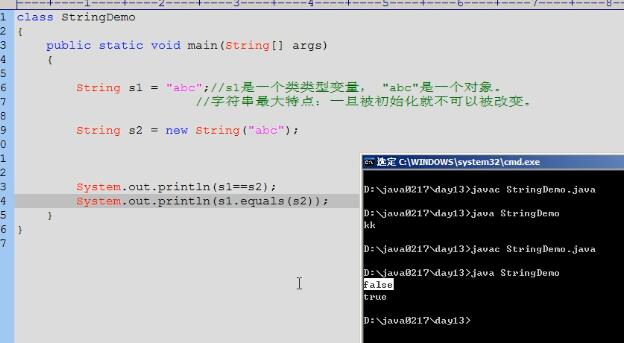
**P155 String（概述）**



虽然最后输出的结果是”kk”,但是“abc”没有改变过，变的是s1指向的对象。这里相当于new了一个对象“kk”并让s1指向它。



-s1==s2；比较的是两个地址值

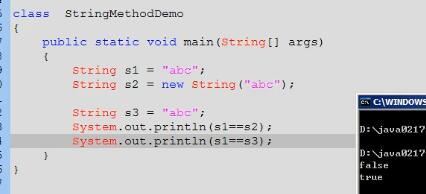
-equals在Object类中讲道理比较的也是地址值，但是在String中对equals方法进行了复写，比较的是两者所指向的对象。

问：s1和s2有什么区别？

-s1有一个对象，s2有两个对象：new了一个，然后String()中又有一个对象。

了解即可：字符串在内存当中其实有一个常量池，存放的是”a”、”b”这些字符，池子里面是个数组，对字符进行组装。

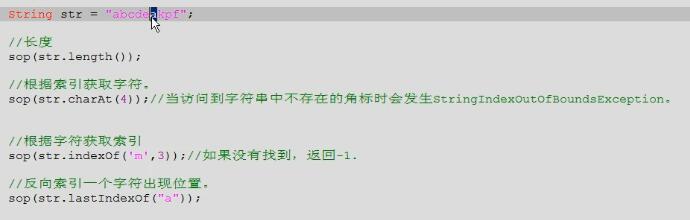
**P156 String（常见功能-获取和判断）**



为什么第二个是true？

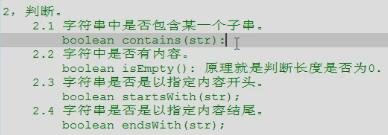
-s1在内存中创建了字符串对象”abc”，字符串是一种特殊的对象，存在于常量池中（常量池也在方法区中，除此之外还有方法代码，静态区）。所以s3在进行初始化时发现“abc“已经存在，就不会独立开辟空间了。

常见的字符串获取操作



indexOf()可以传入字符、字符串、ascii码

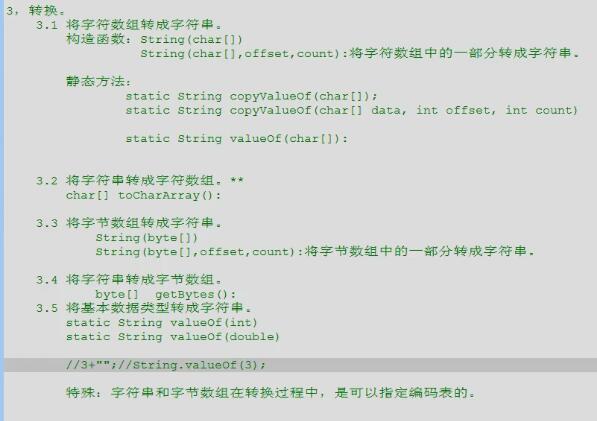
常见的字符串判断操作



以2.1为例

首先字符串是一个对象，有contains方法，所以直接 对象.contains()进行调用，向方法中传入一个String类型的子串，该方法就会返回一个boolean类型的值。

**P157 String（常见功能-转换）**



**P158 String（常见功能-切割和替换）**



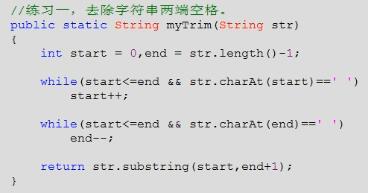
即s == s1

**P159 String（常见功能-比较和去除空格）**

//

**P160 String（字符串练习1）**

不借助trim方法（跟python很像，挺简单的）



**P161 String（字符串练习2）**

//

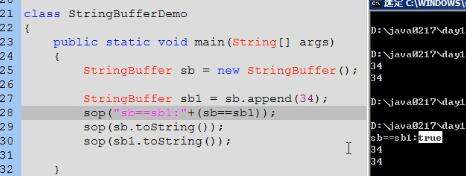
**P162 String（字符串练习3）**

//

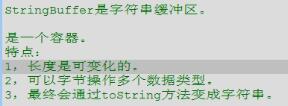
**P163 String（字符串练习4）**

//

**P164 StringBuffer（常见功能-添加）**



说明sb和sb1指向了同一个对象



**P165 StringBuffer（常见功能-删除和修改）**

//

**P166 StringBuilder**

单线程建议使用StringBuilder（线程不同步）

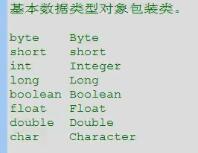
多线程建议使用StringBuffer（线程同步）

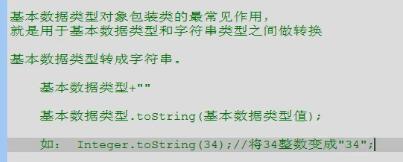
Java升级不外乎3个因素：

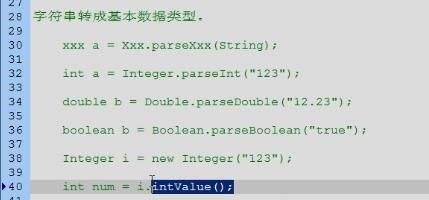
1. 提高效率
2. 简化书写
3. 提高安全性

StringBuilder的出现是原因1.

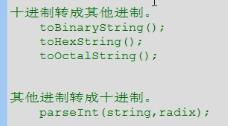
**P167 基本数据类型对象包装类**



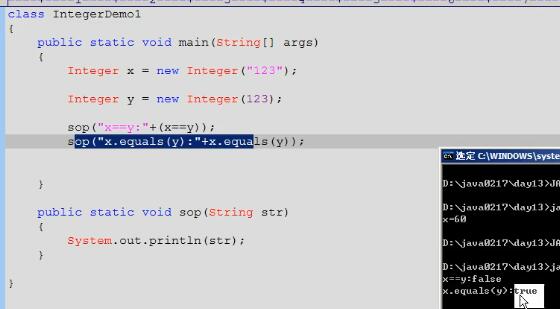




38和40行的效果等同于32行



**P168 基本数据类型对象包装类新特性**

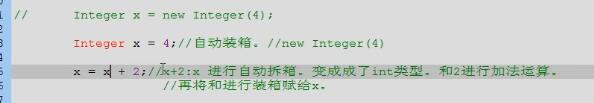


不要被什么int，byte之类的迷惑，把它当成普通的对象处理

“x==y”：xnew了一个对象，y又new了一个对象，两个对象的地址值当然不同。

“x.equals(y)”：Integer类重写了equals方法，比较的是对象的值，相等。

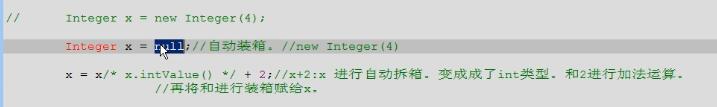
JDK1.5版本以后出现的第一个新特性



下面这行看上去跟 int x =4； 很相似，其实是等价于上面这行：new了一个Integer类型的变量x，又new了一个Integer对象并传入参数4，并将地址赋给x。是为了简化书写的优化

类类型变量必然指向对象

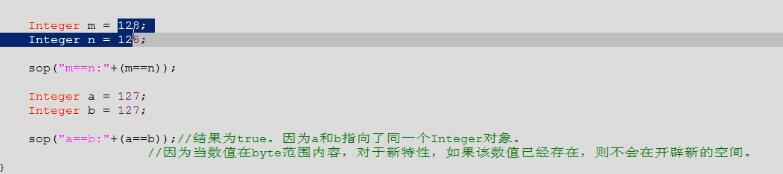
但是在使用自动拆箱和装箱时有个问题



-x毕竟是个类类型变量，可以为空。此时就会产生空指针异常。

所以在使用该方法直接运算时，注意要判断是否为空

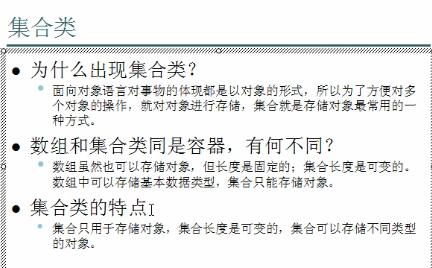
JDK1.5版本以后出现的第二个新特性

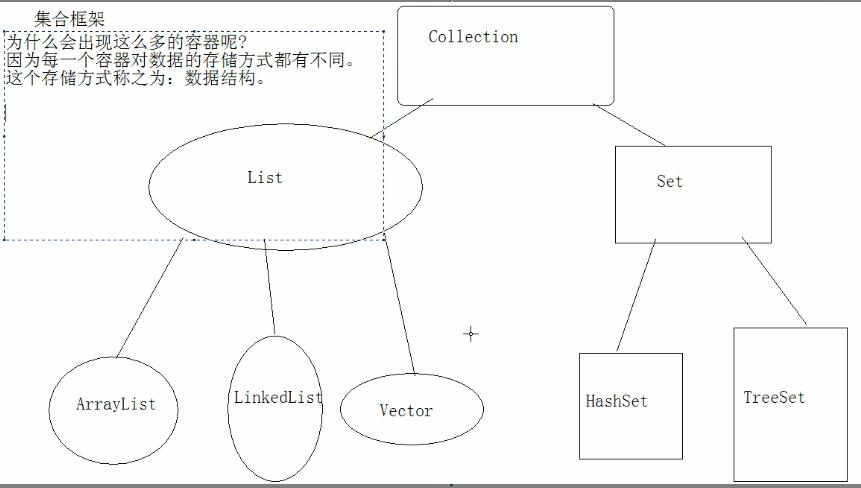


第一个输出false，很好理解，因为m和n都是类类型变量，指向了不同的对象。

第二个输出true。这是新特性。

**P169 集合框架（体系概述）**





**P170 集合框架（共性方法）**

为什么ArrayList中的add(Object obj)方法接收的是Object类型的？

因为集合容器接受的是类类型变量，多态的体现。

集合创建之后是在堆内存中，里面存放的是对象的地址而不是对象实体

**P171 集合框架（迭代器）**

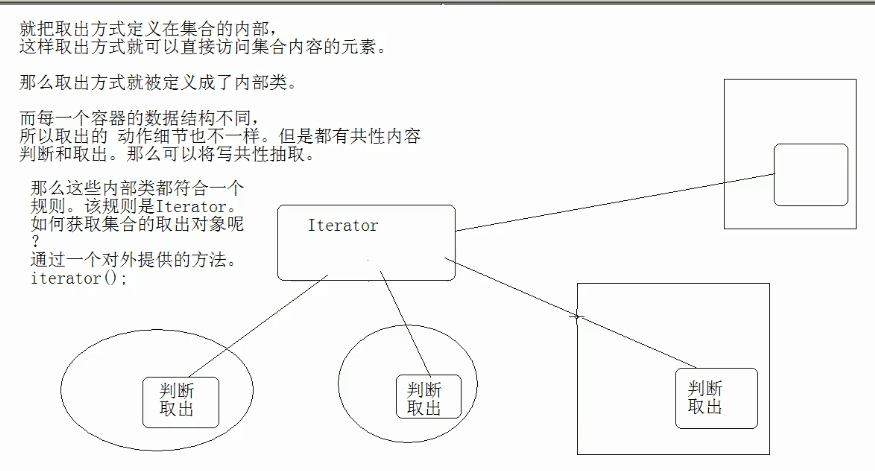


-al.iterator()返回了Iterator接口的子类对象，用Iterator类型的引用型变量去指向Iterator接口的子类对象，多态。

迭代器的由来（这也是一个对象在调用方法后返回其他类型对象的思想）

每一个容器都有存和取的方式，而且因为每个容器的数据结构不一样，所以存和取具体实现的动作细节也不一样。取不像存那么简单，用一个方法不足以来描述（就是取动作在实现前要先判断容器中是否还有未取出的），这就需要两个方法来实现：判断和取出。大于一个方法时，最好把这些方法封装成一个类，通过类来调用（我们看到Iterator对象，就具备hasNext和next方法。这两个方法在实际使用中也是绑定的）。同时为了便于直接操作容器内的元素，就将其定义为内部类。不同的数据类型虽然操作的实现方式不同，但是都需要这些操作，就抽取成了Iterator接口。

也可以这么想：判断和取出是拓展功能，并不是每个数据类型都要有这两个功能，所以作为接口最为合适。于是就变成了接口1-判断，接口2-抽取。但是这两个接口又往往是绑定使用的：即要么两个接口都不用，要么两个接口都使用（在用的时候当然可以只用一种方法，但缺少另一种方法就不完整。就像加油站可以加93号，也可以加97号，但是只提供一种型号的汽油就显得不完整），所以就把这两个接口给进一步封装成Iterator接口。

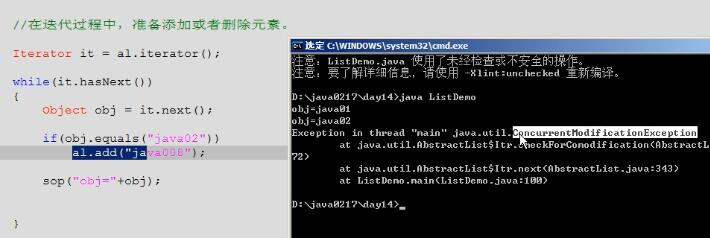


形象的例子：容器相当于娃娃机，Iterator相当于娃娃机里面的夹子。夹子是个复杂事物，方法很多：移动、夹取。

**P172 集合框架（List集合共性方法）**

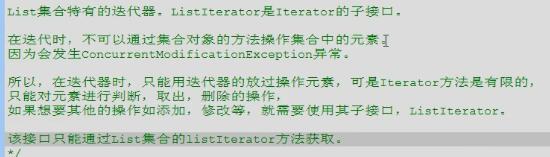
//

**P173 集合框架（ListIterator）**

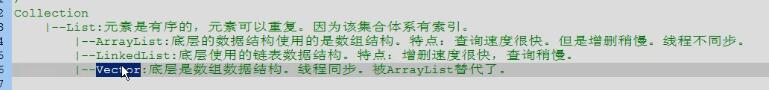


问题产生的原因：你既用迭代器对列表进行操作，又用列表自身的方法对列表进行操作。比如你生成迭代器的时候列表有3个元素，然后你有用列表自身的方法删除了一个，那么删掉的这个迭代器怎么处理呢？

解决方法：



**P174 集合框架（List集合具体对象的特点）**

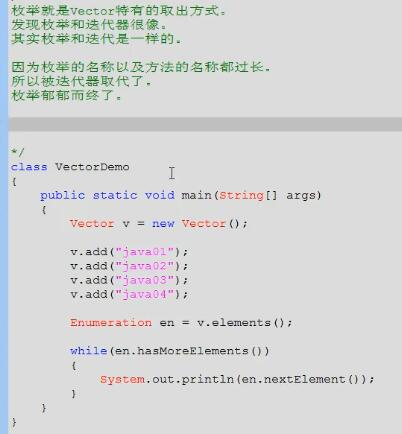


ArrayList，LinkedList就是python里面的列表和链表。

ArrayList长度可变，怎么变的了解一下：默认new的是长度为10，当不够时，新建一个延长50%的ArrayList，把原ArrayList的10个复制过来，然后在后面接着添加。

**P175 集合框架（Vector中的枚举）**

随便看看



**P176 集合框架（LinkedList）**

//

**P177 集合框架（LinkedList练习）**

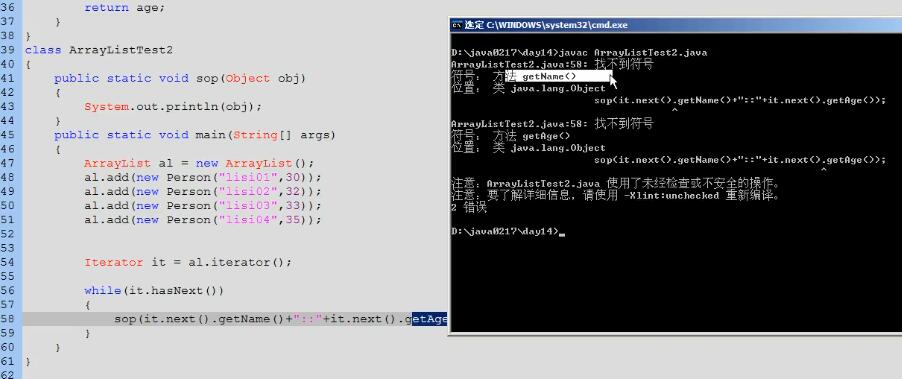
//

**P178 集合框架（ArrayList练习）**

//

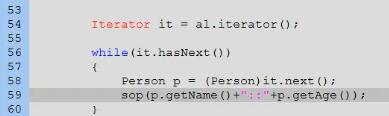
**P179 集合框架（ArrayList练习2）**

练习1：获取其他类型对象的一些信息

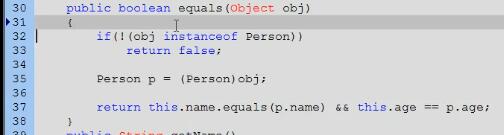


明明Person类定义了getName和getAge方法，为什么说找不到呢？

这是一个老生常谈的问题了。因为it.next()返回的是Object类，该类并没有这些方法，所以在调用这些子类的特有方法之前，要先向下转型。



练习2：去除其他类型对象中的重复对象



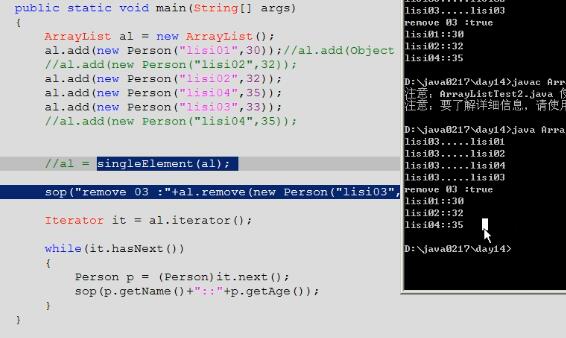
默认比较的是地址值，这一般没啥意义，所以要自己重新定义一下。

注意35行的强制转换。

问：30行和37行的equals方法是否相同？

不是。首先看对象。30行的对象是Person类型的引用型变量，37行的对象是this.name，是String类型

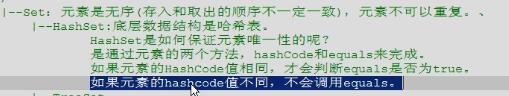
其实很多列表中的方法，调用的就是列表中元素的equals方法。默认的比较是地址值，没啥意义，当我们重写了之后发现（以remove为例）remove方法的底层实现：我们想要删除列表中的lisi03，它就拿lisi03和列表中的每个元素相比，如果都不相同返回false，有相同的删掉相同，同时返回true

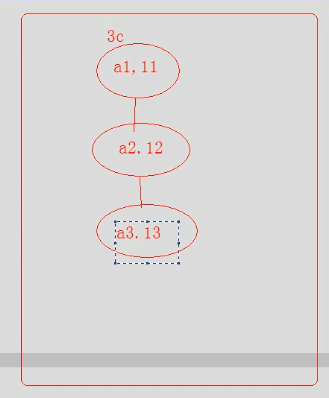


**P180 集合框架（HashSet）**

//

**P181 集合框架（HashSet存储自定义对象）**





先判断hashCode是否一样，一样的话判断equals返回的是true还是false，false的话串在后面

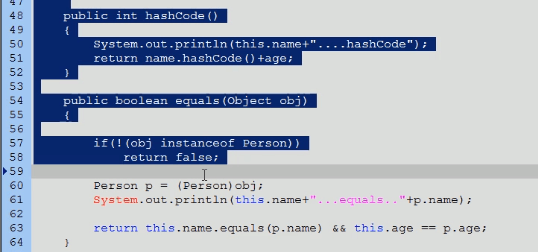
//需求：Person对象，传入的姓名年龄相同，认为是相同

像在P179中，对ArrayList的相关操作的底层实现是依赖于equals方法的，所以我们只要重写equals方法就能实现我们想要实现的功能，

但是HashSet不同，它的储存方法的底层实现是先判断hash值是否相同（即先调用hashCode方法看是否相同，默认的hashCode方法返回值是根据内存地址值，用hash算法算出来的，所以内存地址值不同，hashCode返回值必不同），不相同直接存储，相同的话比较对象是否相同（即再调用equals方法，也就是==看是否相同，即类类型和内存地址值是否相同。注意自定义类型的equals继承自Object，而String类对equals已经进行过复写，所以两者差别很大），相同的话不存入，不同的话串在后面。

所以要重写hashCode和equals方法：

首先，传入的姓名年龄相同的两个new出来的对象，它们调用hashCode应该返回相同的值。在此基础上，调用equals方法的判断结果也应该基于name和age。



当然，50和61行没意义，可以不写

总结：往集合里面存东西，一般要复写hashCode和equals

**P182 集合框架（HashSet判断和删除的依据）**

-ArrayList判断和删除元素只依赖equals

-hashSet判断和删除元素依赖于hashCode和equals。先比较hashCode，不同的话不执行操作。相同的话再比较equals，不同的话不执行操作，相同的话执行操作。