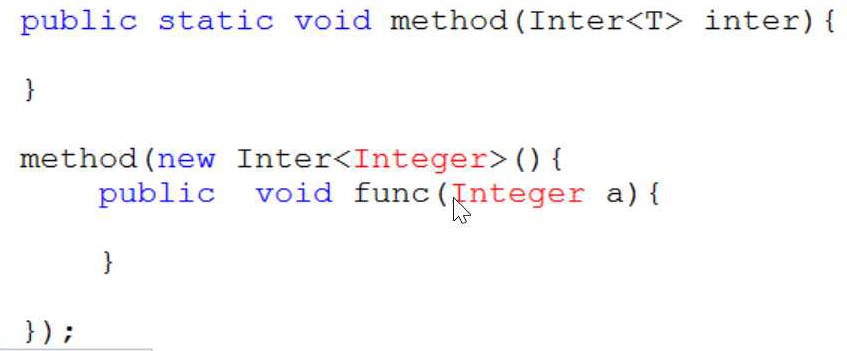
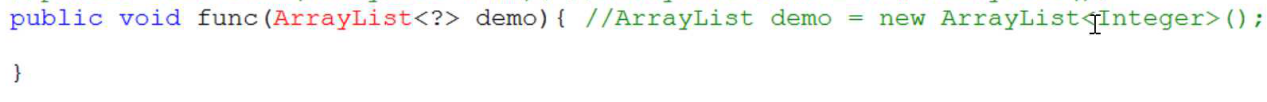
1 不能改原来的类代码，因为会影响其他的引用该类的代码。

2 泛型接口作为形参要重写方法里面的泛型，变为具体的类型：

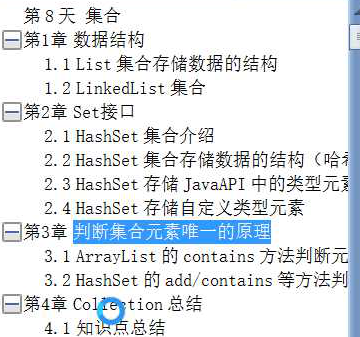


3 问号的作用是允许对象使用泛型（**若没有问号的话右边的对象不能使用泛型**）：



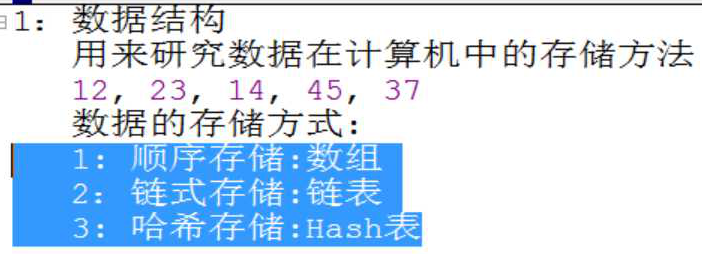
4 问号的缺点是只能获取数据不能添加数据。

5 今天内容：

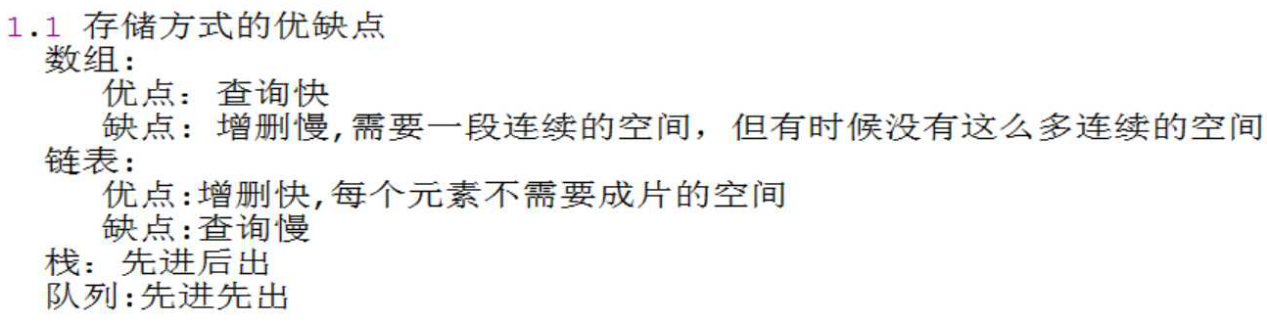


6 今天的重点是数据结构和HashSet的实现原理。

7

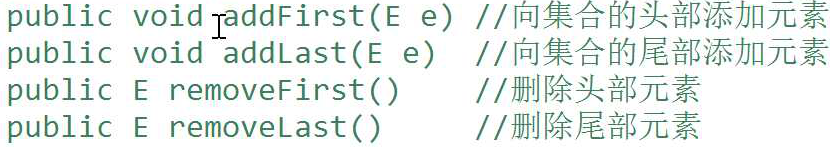


8 各种存储方式的优缺点：



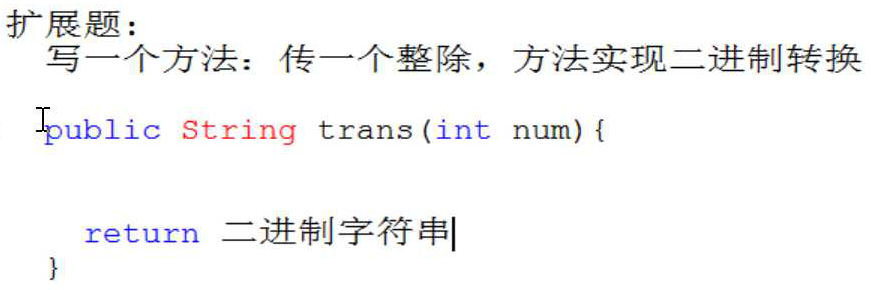
9 方法的调用的数据结构是栈，记录方法的返回点。十进制转换成二进制也是用栈。所有取出和输入的顺序相反的都用栈。

10 LinkedList的特有方法，自己看。例如：（**注意remove方法是有返回值的。**）



11 **链表LinkedList会有栈的pop和push方法。**

12 作业题：

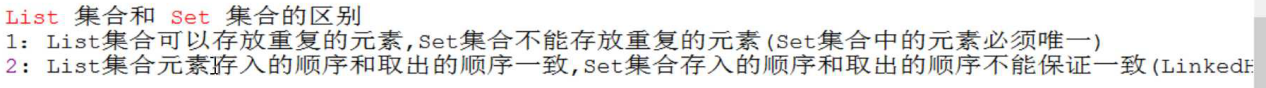


13 ArrayList和LinkedList的区别：



14 快速排序也是用的栈。

15 List集合和Set集合的区别：

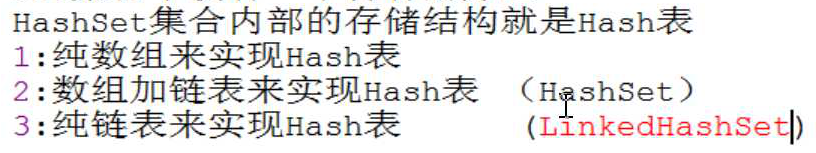


16 **Set集合的方法全都是collection的方法，没有一个特有的方法。**

17 set集合无法保证自定义集合的唯一性，因为没有告诉集合比较的规则，**解决方法不仅是重写equals()（alt shift+s），还要重写hashCode()方法。(add和contain方法里面先调用hashcode再调用equals)**

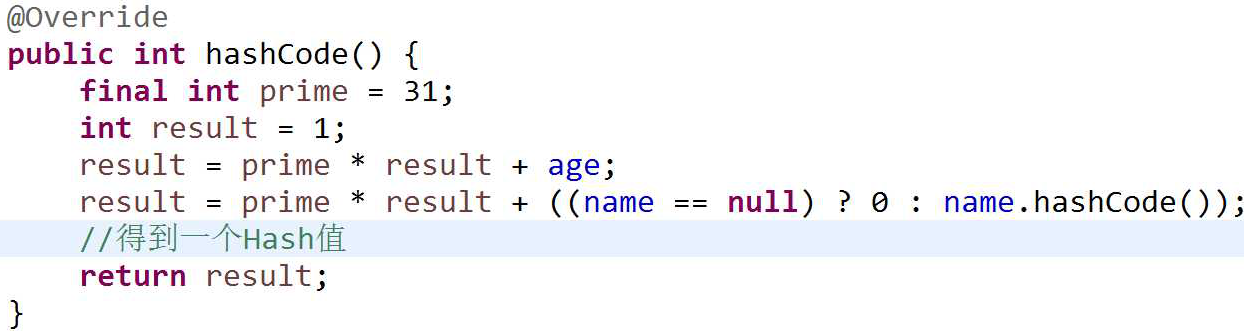
**18 从方法中回去用alt+左方向键。**

**19 哈希表的分类：**

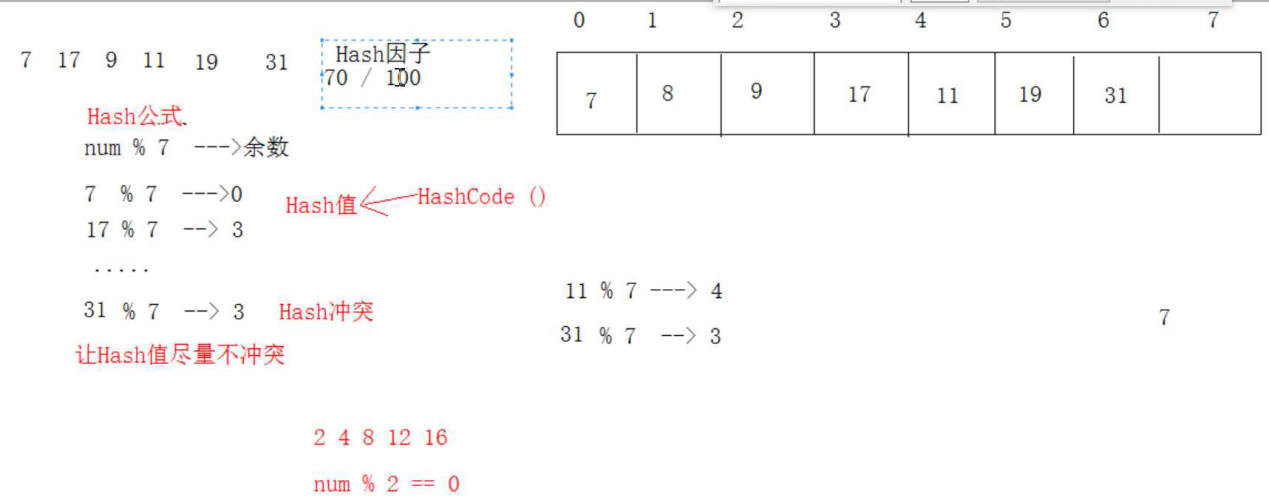


20 哈希表是查询速度最快的存储方式，比数组的速度快几个数量级。

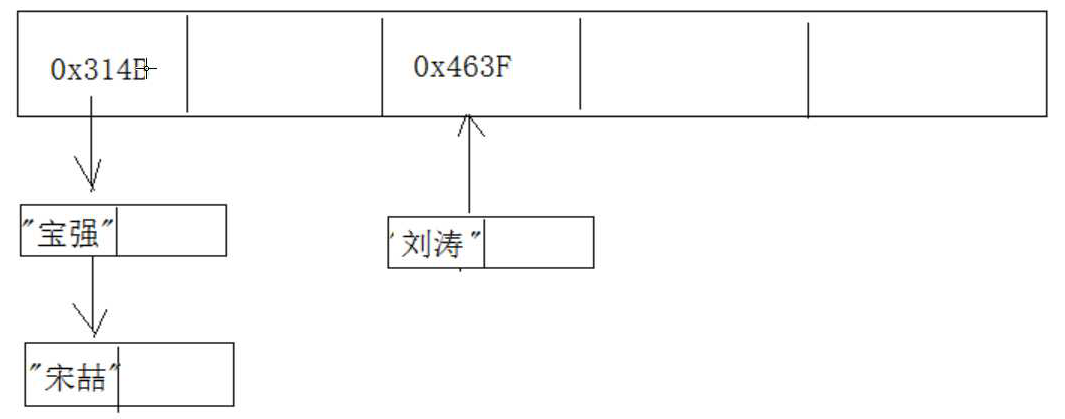
21 hashCode()方法实现：



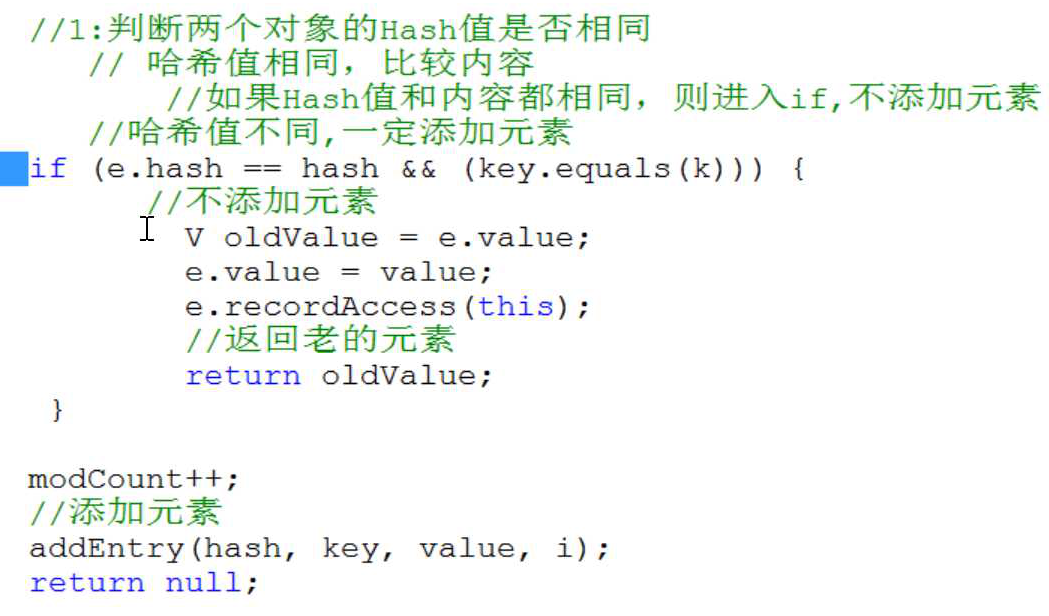
22 纯数组哈希表：（**留出空间的原因是当哈希值冲突的时候存放冲突的元素，对应的概念是哈希因子。**）



23 数组加链表实现哈希表：是将哈希冲突的数据放入链表中：



24 哈希表的add()方法原理（不能添加相同元素的原因）：（contains也一样）



可以看出只有哈希值相同才会走equals()方法。

25 **上帝的hashCode()走的是对象的地址值**，所以自定义对象要重写hashCode()方法，因为地址值必然不相同。

26 Set集合输出顺序和添加顺序不同的原因是哈希值的算法。

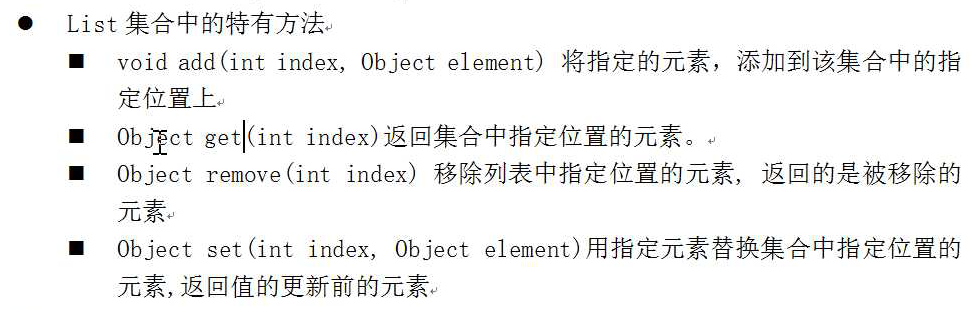
27 LinkedHashSet:（链表的相互嵌套）



28 LinkedHashSet不是排序，而是存取有序。

29 set集合要同时重写equals()和hashCode()方法。

30 区分**add()和set()**，前者是将原来位置上的后移一位，后者是直接替代：

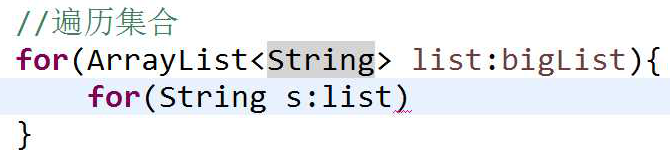


31 **只有List集合才能操作下标。**（**注意LinkedList链表也可以有下标？？？？？？**）

32 集合的嵌套在实际开发中常用。



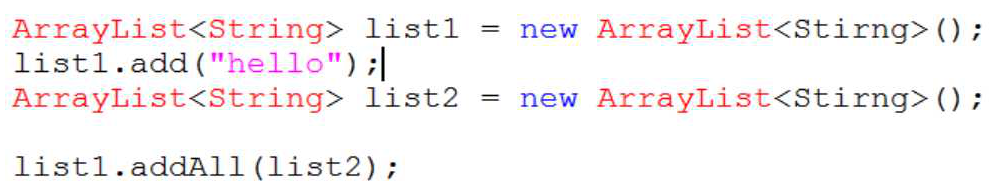
嵌套遍历嵌套集合：



33 点中一个方法名如何全选？答：双击。

34 遍历集合如何判断当前元素的索引？？？？？？？？？？？？？？？？有几种方法？

35 addAll()用于添加整个集合：



36 **集合做成员的时候必须初始化：（否则也要在添加集合元素之前初始化）**

**private** ArrayList<Student> arr=**new** ArrayList<>();//注意集合必须new!!!!!必须出初始化

37 并发修改异常是运行时异常，编译时不报错。

38 集合的addAll()必须传入集合，不能传入数组等。

39 **pop（）会返回（弹出）第一个add（）进去的元素。**

40 重载和重写：前者参数不同，后者参数相同。比如equals()方法，就是重写。参数相同的话即使函数体相同也不叫重写。

41 数组和链表是存储方式，栈和队列是逻辑关系，而不是存储方式。数组和链表都可以实现各种存储方式。