**常用数据结构：栈、队列、数组、链表和红黑树**

栈：先进后出

栈顶

栈底

入口出口同一侧

存储元素到集合：入栈/压栈

取出集合的元素：出栈/弹栈

队列：先进先出

队头 队尾

入口和出口在集合的两侧

数组：

查询快：数组的地址是连续的，通过索引可快速查找某一个元素

增删慢：数组的长度固定，新增删除一个元素，必须创建新的数字，将源数组的数据复制过来

新数组长度为源数组长度减一，将新数组的地址赋给变量arr

源数组会在内存中被销毁

链表：

查询慢：链表中的地址不是连续的，每次查询必须从头开始查询

增删快：增删元素对链表的整体结构没有影响

链表中每个元素称之为一个节点

一个节点包括了一个数据源，2个指针域（存储地址）

自己的地址 数据 下一个节点的地址

单向链表：链表中只有一条链子，不能保证元素的顺序，无序

双向链表：链表有两条链子，一条专门记录元素的顺序，是有序集合

红黑树

二叉树：分支不能超过两个

java.util.list接口extends Collection接口

有序集合，存储元素和取出元素的顺序是一致的

有索引，包含一些带索引方法

允许存储重复的元素

add index elment添加

set index element替换

get index

remove index

使用迭代器

Iterator<String> it=list.iterator();

while(it.hasNext()){

String s=it.next();

System.out.println(s);

}

增强for

for(String s:list){

System.out.println(s);

}

List的实现类：

1、ArrayList

不是同步，查询快，增删慢，底层是个数组结构

2、LinkdList

底层是链表结构，查询慢，增删快

addFirst

addLast

push=addFirst

getFirst

getLast

removeFirst

removeLast

pop=removeFirst

clear清空集合

了解：Vector

可实现可增长的对象数组，底层同ArrayList为数组，同步

Set接口同List接口同样继承collection接口

不包含重复元素的collection，因此没有带索引的方法，也不能使用普通的for循环遍历

HashSet实现Set接口

HashSet：

不允许重复元素存储

不同步，是无序集合，存储元素和取出元素的顺序有可能不一致

底层是hash表结构（查询快）

使用迭代器遍历set或者增强for遍历

Iterator<Integer> it=set.iterator();

while(it.hasNext()){

Integer n=it.next();

System.out.println(n);

}

哈希值：是一个十进制整数，由系统随机给出，即对象的地址值，是个逻辑地址，是模拟得到不是数据实际存储的物理地址

重写hashcode

由于Object的hashCode返回的是对象的hash值，所以即使equals返回TRUE，集合也可能判定两个对象不等，所以必须重写hashCode方法，以保证当equals返回TRUE时，hashCode也返回Ture，这样才能使得集合中存放的对象唯一。stu1和stu2这两个对象的属性值是完全一样的，那么从业务角度来说，这两个对象应该就是重复的，那么他们生成的哈希值也应该是一致的，而现在显然并不一致，因此我们需要为这个Student类重写hashCode()方法。

哈希表=数组+链表

哈希表=数组+红黑树（提高查询的速度）

如果链表长度超过8位，将会把链表转换为红黑树