●集合框架(去除 ArrayList 中重复字符串元素方式)(掌握)

A:案例演示

需求：ArrayList 去除集合中字符串的重复值(字符串的内容相同)

思路：创建新集合方式

```
public class Demo1_ArrayList {
    public static void main(String[] args) {
        ArrayList list=new ArrayList();
        list.add("a");
        list.add("a");
        list.add("b");
        list.add("b");
        list.add("c");
        list.add("c");

        ArrayList newList=getSingle(list);
        System.out.println(newList);

    }


    //案例 需求：ArrayList 去除集合中字符串的重复值(字符串的内容相同)
    //创建新集合将重复元素去掉
    //明确返回类型ArrayList
    //明确参数列表ArrayList
    //1、创建新集合
    //2、根据传入的集合(老集合)获取迭代器
    //3、遍历老集合
    //4、通过新集合判断是否包含老集合中的元素 不包含就添加
    public  static ArrayList getSingle(ArrayList list){
        ArrayList newlist=new ArrayList();
        //老集合获取迭代器
        Iterator it=list.iterator();
        //遍历老集合
        while (it.hasNext()){
            //老集合遍历出来了之后记录
            Object obj=it.next();//记录每个元素
            //如果新集合里没有老集合里元素
            if (!newlist.contains(obj)){
                //就添加
                newlist.add(obj);
            }
        }
```

```
        //返回新集合
        return  newlist;
    }
```

```
package com.sxt.fuxi;

import java.util.ArrayList;
import java.util.Iterator;

public class Demo9 {
    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(4);  //add 方法往集合中添加数据通通都会自动提升为Object 类型
       list.add(5);
        list.add(5);
        list.add(6);
        list.add(6);
        list.add(4);
        //取出集合中重复的数据
       ArrayList list2=  new ArrayList();
        Iterator it = list.iterator();
        while(it.hasNext()){
            Object obj = it.next();
            if(!list2.contains(obj)){  //判断新的集合中是否包含元素,不包含就 add 进去
             list2.add(obj);
            }
        }
        System.out.println(list2);

    }
}
```

●集合框架(去除 ArrayList 中重复自定义对象元素)(掌握)

A:案例演示

需求：ArrayList 去除集合中自定义对象元素的重复值(对象的成员变量值相同)

B:注意事项

remove()与 contains（）方法的源码底层是用 equals 来比较

重写 Student 里面的 equals()方法来解决问题

```
package com.sxt.fuxi;


import java.util.ArrayList;
```

```
import java.util.Iterator;

public class Demo9 {
    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(new Student("小明",19));  //add 方法往集合中添加数据通通都会自动提升为Object 类型
      list.add(new Student("小明",19));  //add 方法往集合中添加数据通通都会自动提升为Object 类型
      list.add(new Student("小红",20));  //add 方法往集合中添加数据通通都会自动提升为Object 类型
      list.add(new Student("小红",20));  //add 方法往集合中添加数据通通都会自动提升为Object 类型
      list.add(new Student("高晓松",30));  //add 方法往集合中添加数据通通都会自动提升为Object 类型
      list.add(new Student("高晓松",30));  //add 方法往集合中添加数据通通都会自动提升为Object 类型
      //取出集合中重复的数据
      ArrayList list2=  new ArrayList();
        Iterator it = list.iterator();
        while(it.hasNext()){
            Object obj = it.next();
            if(!list2.contains(obj)){  //判断新的集合中是否包含元素,不包含就 add 进去
             list2.add(obj);
            }
        }
        System.out.println(list2);

    }
}
```

```
package com.sxt.fuxi;
class Student{
    private String name;
    private int age;

    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    @Override
    public String toString() {
```

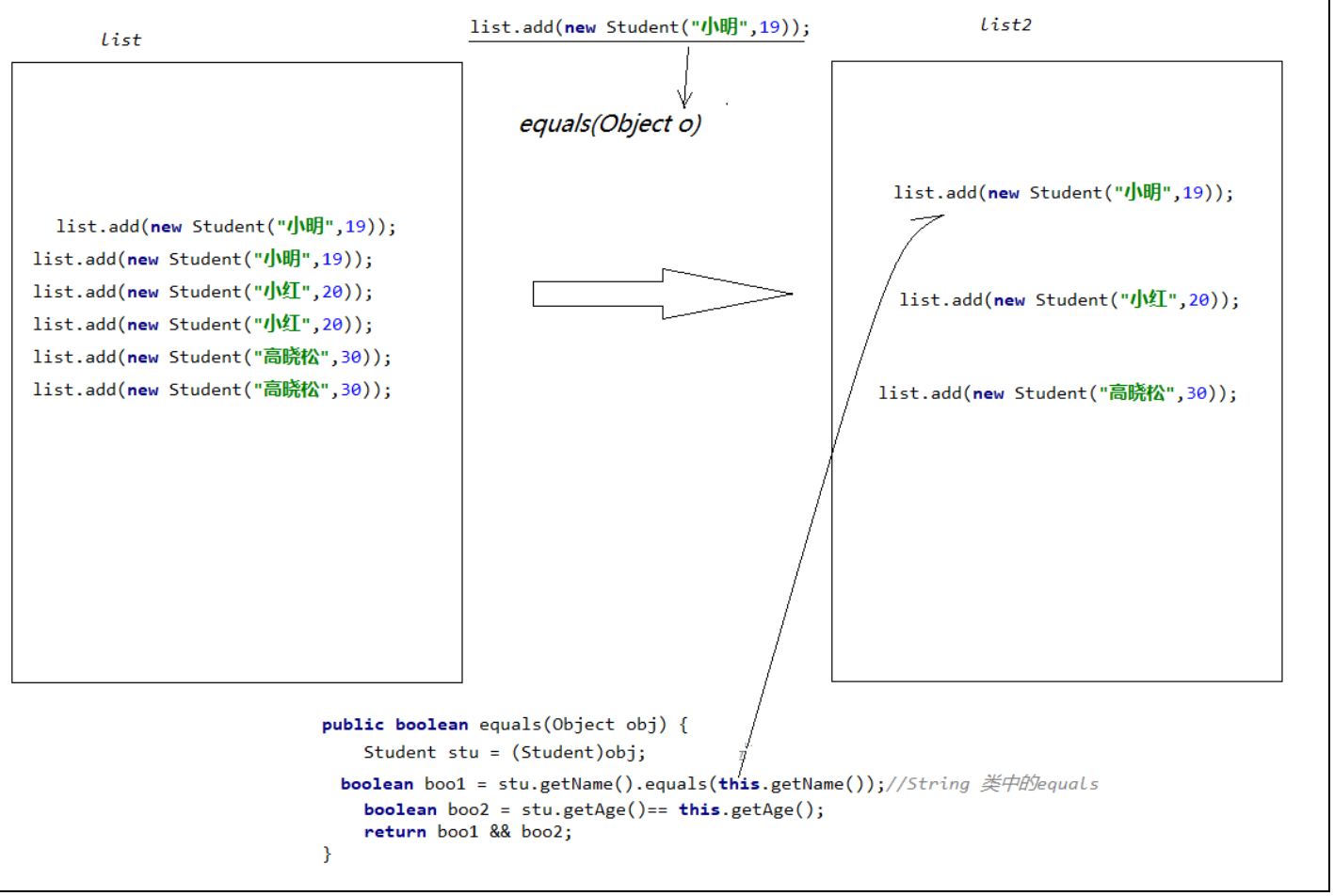
```
        return "Student{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }


}
```

以上代码发现并不能去重,查看 contains 的源码发现,底层用的 equals 对比对象是否重复

父类Object 默认是对比对象的内存地址 所以即使名字和年龄一样 地址不一样也会被集合添加进去

解决:重写 Student 类 equals 方法

```
package com.sxt.fuxi;
class Student{
    private String name;
    private int age;

    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    @Override
    public String toString() {
        return "Student{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }

    //父类Object 默认是对比对象的内存地址,重写父类的equals 方法根据我们的要求来对比对象中的属性作为衡量的重复要求
    public boolean equals(Object obj) {
        Student stu = (Student)obj;
        boolean boo1 = stu.getName().equals(this.getName());//String 类中的equals
        boolean boo2 = stu.getAge()== this.getAge();
        return boo1 && boo2;
    }
}
```



开发中推荐用 idea 自动生成 equals 方法,比我们自己写的要健壮一些

●集合框架(LinkedList 的特有功能)(掌握)

A:LinkedList 类概述

B:LinkedList 类特有功能

public void addFirst(E e)及 addLast(E e)

public E getFirst()及 getLast()

public E removeFirst()及 public E removeLast()

public E get(int index);

LinkedList 的数据结构是链子，所以 get(index i)方法要判断从前面还是后面寻找

了解:get(index i)方法要判断从前面还是后面寻找

```
Node<E> node(int index) {
    // assert isElementIndex(index);

    if (index < (size >> 1)) {
        Node<E> x = first;
        for (int i = 0; i < index; i++)
            x = x.next;
        return x;
```

```
    } else {
        Node<E> x = last;
        for (int i = size - 1; i > index; i--)
            x = x.prev;
        return x;
    }
}
```

get(index i)方法底层会判断从链表的左边还是右边查找
i< size/2 来决定

```
package com.sxt.list;

import java.util.Iterator;
import java.util.LinkedList;

public class Demo2 {
    public static void main(String[] args) {
        LinkedList list= new LinkedList();
        list.add("你好");
        list.add('a');
        list.addFirst(3);
        list.addLast(56.8);
        System.out.println(list);
        Object obj  =list.getFirst();  //获取第一个元素
      Object obj2 = list.getLast();  //获取第最后的元素
      System.out.println(obj);
        System.out.println(obj2);
        System.out.println("-------");

        //遍历
      Iterator it = list.iterator();
        while(it.hasNext()){
            Object ob = it.next();
            System.out.println(ob);
        }
        System.out.println("**********************");
        //遍历2
        for (int i = 0; i <list.size() ; i++) {
            Object ob = list.get(i);//
            System.out.println(ob);
        }
    }
}
```

用的频率不高

●集合框架(栈和队列数据结构)(掌握)

这些操作允许将链接列表用作堆栈(子弹夹)、队列(管道)

此类实现 Deque 接口，为 add、poll 提供先进先出队列操作

栈

先进后出

队列

先进先出

●集合框架(用 LinkedList 模拟栈数据结构的集合并测试)(掌握)

A:案例演示

需求：请用 LinkedList 模拟栈数据结构的集合，并测试
添加数据，先进的后出，后进的先出

题目要求：模拟栈数据结构的集合　应该创建栈对象 而不是 LinkedList 对象

●集合框架(泛型的由来)(了解)

A:案例演示

```
package com.sxt.list;

import java.util.ArrayList;
import java.util.Iterator;

public class Demo3 {
    public static void main(String[] args) {
        ArrayList list =new ArrayList();
        list.add("a");
        list.add("abc");
        list.add(34);
        list.add("bcd");
        Iterator it = list.iterator();
        while(it.hasNext()){
            Object obj = it.next();
```

```
            String s = (String)obj;
            System.out.println(s);
        }
    }
}
```