集合框架练习

一、

```
import java.util.*;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.TreeMap;


public class w {

    public static void main(String[] args) {
    Collection<String> l=new ArrayList<String>();
    l.add("Java1");
    l.add("Java2");
    l.add("Java3");

    for(String s:l){
System.out.println(s);
    }
}

}
```

二、

```
class student{
    private String name;
    private int age;

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

public String toString(){
    return name+" "+age;
}

public int hashCode(){
    return name.hashCode()+age;
}
```

```
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    student stu = (student)o;
    return age == stu.age &&
            Objects.equals(name, stu.name);
}

}



public class w {

    public static void main(String[] args) {
    Set<student> S =new HashSet<>();

S.add(new student(1, "Java1"));
S.add(new student(2, "Java2"));
S.add(new student(1, "Java1"));

    System.out.println(S);

}

}
```

q1:为啥不能

```
public boolean equals(Object i){
if(i==this){
    return true;
}
else{
    return false;
}
}
```

这是不充分的。这种实现只能判断两个引用是否指向同一个对象，而不能判断两个对象的属性值是否相等。为了更好地理解，让我们来看看具体的原因和正确的实现方法。

1. **引用相等性**:

   ```
   javaCopy codeif (i == this) {
       return true;
   }
   ```

   这一行代码只是在检查 `i` 和 `this` 是否是同一个引用。这在某些情况下是有用的，例如当我们希望快速判断两个变量是否引用同一个对象时。但是，这忽略了更重要的对象内容相等性。

2. **属性相等性**： 比较两个对象是否相等通常是比较它们的属性是否相等。例如，两个 `Student` 对象应该被认为是相等的，如果它们的 `name` 和 `age` 属性都相等。

q2:为啥不能

```
public boolean equals(student i)
```

在 `equals` 方法中，参数类型应该是 `Object` 而不是具体的 `student` 类型。这是因为 `equals` 方法是从 `Object` 类继承的，覆盖它时必须使用相同的签名。如果参数类型是 `student`，它不会覆盖 `Object` 的 `equals` 方法，而是定义了一个新的方法。这会导致集合类（例如 `HashSet`）在比较对象时使用默认的 `Object.equals` 方法，从而无法正确比较 `student` 对象。