

# day13\_object\_王世杰

## 1.以下简单题:

1. equals方法和hashCode方法的重写需要注意什么事项?

equals方法:

满足自反性、排他性、对称性、传递性、一致性。

hashCode方法:

1. 当在一个应用程序执行过程中,如果在equals方法比较中没有修改任何信息,在一个对象上重复调用hashCode方法时,它必须始终返回相同的值。从一个应用程序到另一个应用程序的每一次执行返回的值可以是不一致的。

2. 如果两个对象根据equals (Object) 方法比较是相等的,那么在两个对象上调用hashCode就必须产生的结果是相同的整数。

3. 如果两个对象根据equals (Object) 方法比较并不相等,则不要求在每个对象上调用hashCode都必须产生不同的结果。

2. 给一个类的对象做克隆操作的步骤是什么?

浅克隆: 依赖于Object类当中的clone默认实现。

深克隆:

1. 深度克隆是在浅克隆的基础上完成的,所以深度克隆仍然要依赖于Object类当中的clone默认实现

2. 将对象中引用所指向的对象,克隆一份

3. 将克隆后的对象的拷贝引用指向拷贝的对象

3. 什么是深度克隆? 怎么做深度克隆?

深度克隆指的是: 对引用数据类型的成员变量引用所指向的对象,再克隆一次,然后让拷贝成员变量引用指向拷贝对象。这样,深度克隆得到的对象,就会和原先的对象完全独立,是一个全新的对象。

深度克隆做法:

1. 深度克隆是在浅克隆的基础上完成的,所以深度克隆仍然要依赖于Object类当中的clone默认实现。

2. 将对象中引用所指向的对象,克隆一份

3. 将克隆后的对象的拷贝引用指向拷贝的对象

4. 包装类型的自动拆装箱如何实现?

jvm自动实现

5. 包装类型比较取值为什么不能用"=="呢?

因为是引用数据类型。要用equals()方法。

## 2.深度克隆练习

注:String虽然也是引用数据类型,但无需考虑它的深度克隆。

分别定义以下类:

教师类Teacher

属性: int age;String name;Student stu

学生类Student

属性: int age; String name; Star s

明星类Star

属性: int age, String name

尝试完成Teacher对象的深度克隆,并写代码进行测试

```
package com.cskaoyan.homework.day13;

/*
    分别定义以下类:
        教师类 Teacher
        属性: int age;String name;Student stu
        学生类 Student
        属性: int age; String name; Star s
        明星类 Star
        属性: int age, String name

        尝试完成Teacher对象的深度克隆,并写代码进行测试
*/

public class DeepCloneTest {

    public static void main(String[] args) throws
CloneNotSupportedException {
        Star s1 = new Star(21, "迪丽热巴");
        Student stu1 = new Student(18, "王世杰", s1);
        Teacher t1 = new Teacher(41, "长风", stu1);

        Teacher cloneT1 = t1.clone();
        System.out.println(t1.toString());
        System.out.println(cloneT1.toString());
        System.out.println("t1 == cloneT1 : "+(t1 == cloneT1));

    }
}
```

```
class Teacher implements Cloneable {
    int age;
    String name;
    Student stu;

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

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

    @Override
    protected Teacher clone() throws CloneNotSupportedException {
        Teacher cloneTeacher = (Teacher) super.clone();
        Student cloneStu = stu.clone();
        cloneTeacher.stu = cloneStu;
        return cloneTeacher;
    }
}

class Student implements Cloneable {
    int age;
    String name;
    Star s;

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

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

```

@Override
protected Student clone() throws CloneNotSupportedException {
    Student cloneStu = (Student) super.clone();
    Star cloneStar = s.clone();
    cloneStu.s = cloneStar;
    return cloneStu;
}

}

class Star implements Cloneable {
    int age;
    String name;

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








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

    @Override
    protected Star clone() throws CloneNotSupportedException {
        return ((Star) super.clone());
    }

}
}

```

Run: DeepCloneTest x

```

D:\eve\windows\jdk-8u311\bin\java.exe ...
Teacher{age=41, name='长风', stu=Student{age=18, name='王世杰', s=Star{age=21, name='迪丽热巴'}}}
Teacher{age=41, name='长风', stu=Student{age=18, name='王世杰', s=Star{age=21, name='迪丽热巴'}}}
t1 == cloneT1 : false

Process finished with exit code 0
|

```