Java 208 道面试题：第五模块答案

zy java经验总结 3月19日

对象拷贝

61. 为什么要使用克隆?

想对一个对象进行处理，又想保留原有的数据进行接下来的操作，就需要克隆了，Java语言中克隆针对的是类的实例。

62. 如何实现对象克隆?

有两种方式：

1). 实现Cloneable接口并重写Object类中的clone()方法；

2). 实现Serializable接口，通过对象的序列化和反序列化实现克隆，可以实现真正的深度克隆，代码如下：

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

public class MyUtil {

    private MyUtil() {
        throw new AssertionError();
    }

    @SuppressWarnings("unchecked")
    public static <T extends Serializable> T clone(T obj)
                                    throws Exception {
        ByteArrayOutputStream bout = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(bout);
```

```
        oos.writeObject(obj);

        ByteArrayInputStream bin =
                    new ByteArrayInputStream(bout.toByteArray());
        ObjectInputStream ois = new ObjectInputStream(bin);
        return (T) ois.readObject();

        // 说明：调用ByteArrayInputStream
        //或ByteArrayOutputStream对象的close方法没有任何意义
        // 这两个基于内存的流只要垃圾回收器清理对象就能够释放资源，
        //这一点不同于对外部资源（如文件流）的释放
    }
}
```

下面是测试代码：

```
import java.io.Serializable;

/**
 * 人类
 * @author nnngu
 *
 */
class Person implements Serializable {
    private static final long serialVersionUID
                                = -9102017020286042305L;

    private String name;    // 姓名
    private int age;        // 年龄
    private Car car;        // 座驾

    public Person(String name, int age, Car car) {
        this.name = name;
        this.age = age;
        this.car = car;
    }
```

```
21
22      public String getName() {
23          return name;
24      }
25
26      public void setName(String name) {
27          this.name = name;
28      }
29
30      public int getAge() {
31          return age;
32      }
33
34      public void setAge(int age) {
35          this.age = age;
36      }
37
38      public Car getCar() {
39          return car;
40      }
41
42      public void setCar(Car car) {
43          this.car = car;
44      }
45
46      @Override
47      public String toString() {
48          return "Person [name=" + name + ",
49                          age=" + age + ", car=" + car + "]";
50      }
51
52  }
```

```
1  /**
2   * 小汽车类
3   * @author nnngu
```

```
 *
 */
class Car implements Serializable {
    private static final long serialVersionUID
                                = -5713945027627603702L;

    private String brand;       // 品牌
    private int maxSpeed;       // 最高时速

    public Car(String brand, int maxSpeed) {
        this.brand = brand;
        this.maxSpeed = maxSpeed;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public int getMaxSpeed() {
        return maxSpeed;
    }

    public void setMaxSpeed(int maxSpeed) {
        this.maxSpeed = maxSpeed;
    }

    @Override
    public String toString() {
        return "Car [brand=" + brand + ",
                        maxSpeed=" + maxSpeed + "]";
    }

}
```

```
class CloneTest {

    public static void main(String[] args) {
        try {
            Person p1 = new Person("郭靖", 33,
                                   new Car("Benz", 300));
            Person p2 = MyUtil.clone(p1);   // 深度克隆
            p2.getCar().setBrand("BYD");
            // 修改克隆的Person对象p2关联的汽车对象的品牌属性
            // 原来的Person对象p1关联的汽车不会受到任何影响
            // 因为在克隆Person对象时其关联的汽车对象也被克隆了
            System.out.println(p1);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

注意：基于序列化和反序列化实现的克隆不仅仅是深度克隆，更重要的是通过泛型限定，可以检查出要克隆的对象是否支持序列化，这项检查是编译器完成的，不是在运行时抛出异常，这种是方案明显优于使用Object类的clone方法克隆对象。让问题在编译的时候暴露出来总是好过把问题留到运行时。

63. 深拷贝和浅拷贝区别是什么?

- 浅拷贝只是复制了对象的引用地址，两个对象指向同一个内存地址，所以修改其中任意的值，另一个值都会随之变化，这就是浅拷贝（例：assign()）
- 深拷贝是将对象及值复制过来，两个对象修改其中任意的值另一个值不会改变，这就是深拷贝（例：JSON.parse()和JSON.stringify()，但是此方法无法复制函数类型）

（完）