设计模式 - 原型模式（下）

寂然

前情提要

案例演示

浅拷贝

> 对于数据类型是基本数据类型的成员变量，浅拷贝会直接进行值传递
>
> 对于数据类型是引用数据类型的成员变量，那么浅拷贝会进行引用传递

深拷贝

> 复制对象的所有基本数据类型的成员变量值
>
> 为所有引用数据类型的成员变量申请存储空间，并复制每个引用数据类型成员变量所引用的对象，直到该对
>
> 象可达的所有对象

进行深拷贝会复制整个对象，包含对象的引用类型

实现方式一：重写 clone() 方法

分布进行

处理基本数据类型和String类型

处理引用类型 ---》完成深拷贝的

非常繁琐的，单独处理， 一劳永逸的方法？？？

级联处理演示

如果引用类型是个class，类型，里面仍然有引用类型的成员属性同样，核心思想是分布单独处理，假设类

DeepProtoType 里有成员属性 deepCloneableTarget ，而DeepCloneableTarget 类中有成员属性 witness 同样

使用方式一，重写 clone() 方法完成深拷贝，示例代码如下图所示

实现方式二：通过对象序列化

？？？ 一步到位 （推荐）不管类的结构如何复杂，整体处理

注意事项

特点

- 简化对象的创建的流程，同时也可以提高效率
- 不用重新初始化对象
- 如果原始对象发生变化，其克隆对象也会发生相应的变化，无需修改代码（手动测试）

劣势

每个类都需要配备一个克隆方法 ocp原则

小李求职 深拷贝和浅拷贝

下节预告

建造者模式 创建型模式的最后一种