prototype模式【拷贝逻辑放在统一的地方】

场景，如果一个对象，我们希望拷贝一个这个对象，就是弄出来这个类的另外一个实例，实例的属性跟之前的实例是一模一样的

我们要用原型模式，以某一个对象为原型，然后对这个对象进行拷贝，得到拷贝后的另外一个对象

public interface **Prototype** {

   Prototype clone();

}

public class **ConcretePrototype implements Prototype** {

   private String field1;

   private String field2;

   public Prototype clone() {

       Prototype prototype = new ConcretePrototype();

       prototype.setField1(field1);

       prototype.setField2(field2);

       return prototype;

   }

}

原型模式，顾名思义，其实说白了，就是让一个对象可以自己拷贝自己，对象把自己当成一个原型，然后提供一个方法出去，外部要一个对象的克隆和拷贝，直接就copy一份就可以了，但是这里要记住深拷贝和浅拷贝的区别

因为一个对象可能还持有别的对象的引用，浅拷贝就是不管引用的其他对象了；深拷贝就是将引用的对象也一起拷贝一份；**一般原型模式都是要支持深拷贝的**

而且其实一般实现原型模式的时候，**直接是通过覆盖Object类的clone()方法即可**（不用单独写一个 **Prototype** 接口），在里面实现自己的拷贝逻辑就可以了

使用原型模式比较好的一点，就是如果别人要拷贝你的对象，不需要调用方自己实现拷贝逻辑，将拷贝逻辑放在对象自己身体里就可以了，对外面调用都是透明的

这个模式其实在我们的电商系统中，完全有场景可以使用，大家想象一下，比如我们的这个订单、发货单、销售出库单、财务结算单，这些单子之间，可能就是有大量的属性是要互相拷贝的。我们手上有一个订单之后，想将订单中的数据拷贝到一个发货单中，将发货单中的属性拷贝到一个销售出库单中，再将销售出库单中的属性拷贝到财务结算单中。

这种大量的属性拷贝的业务场景，我们就可以对原型模式进行改造，我们可以实现一个适合自己系统的通用的对象属性拷贝工具类，然后对那些需要互相之间支持属性拷贝的类，我们可以重写其clone()方法，这个我们后面再讲解，发货单的clone()方法，调用之后，克隆出来的是一个销售出库单

853769620