设计模式测试

表格

编号	设计模式	Class/Interface API	framework完成分数	Sample program完成度	备注说明
1	Abstract Factory	WemStore::newDrink() WemStore::newSnacks() WsStore::newDrink() WsStore::newSnacks() Drink::sell() Drink::Import() Snack::sell() Snack::Import()	70	30	
2	Bridge	PurchaseByWechat::Payment() PurchaseByAlipay::Payment() Payment::Purchase()	70	30	
3	Proxy	OnlineProxy::BuyAndPay()	70	30	
4	Template	Facility::FacilityRunTemplate()	70	30	
5	Prototype	Dorm::clone() Spot()::clone()	70	30	
6	State	AgeContext::input() AgeContext::setState()	70	30	

评价

在使用Abstract Factory设计模式时，该组将两个不同超市作为工厂定义工厂基类，每一个工厂只生成固定的一类产品，与该设计模式思想较为符合，正确的使用了相关定义。

在使用Bridge设计模式时，该组将pay行为定义为payment行为的属性，使得两者实现解耦，可以各自处理自己的继承关系，较好的体现了桥接的思想。美中不足的是在pay购买类中存在可以简化的定义方式。

在使用Proxy设计模式时，该组将购买行为作为代理购买行为的属性，实现了网上代理买票的功能场景。在测试中，需将实例化的真实购买场景作为参数对在线购买进行初始化。

在使用Template测试模式时，该组将设施的使用流程细分为五个步骤，将其中几个待定的步骤交给继承的子类完成，非常好的体现了模版的思想，代码清晰易懂。

在使用prototype设计模式时，该组通过使dorm和spot继承java默认的Clonable类，非常简洁的实现了该设计模式的思想。

在使用State设计模式时，该组将不同票价作为不同状态，通过改变年龄同步修改状态，实现了票价状态随年龄变化的功能

2019/11/8