一．测试清单:

编号	设计模式	Class:Interface API	framework 完成分数	sample program 完成分数
1	Builder	BuilderWorker:Build(Builder builder)	70	30
2	Composite	ThemeDistrict:seekSpot(String spotname) Spot:showComments() Spot:getInfo()	70	30
3	Flyweight	ViewSpotTypeFactory: ViewSpotType getViewSpotType(String tag)	70	30
4	Singleton	Casino:Casino getInstance()	70	30
5	Singleton	AlarmSystem:addAlarmBellCommand(String command) AlarmSystem:open() AlarmSystem:close() AlarmSystem:notify(int order)	70	30

二．测试总评：

Builder 模式：在建造者模式的设计和使用中可以非常清晰的感受到作者对建造者模式有着很深的理解和认识了，对指导者和建造者之间的区别把握的十分细腻，此外还能使用抽象基类来减少代码冗余，十分易于阅读。

Composite 模式：在组合模式里，设计者将这一模式嵌入景点的嵌套这样一个场景里是十分合适的，spot 提供了面向参观者的接口，在 spot 下面是子景点 ScenicSpot 和主题园区 ThemeDistrict，设计的十分贴切合理，类与类之间关系清晰。

Flyweight 模式：这里的享元是不同建筑物的标签信息，确实游乐园中建筑存在大量相同的相同属性，最大的相同点就是扮演的角色功能，这样设计可以有效减少给建筑物打标签或添加属性时的操作步骤。

Singleton 模式：这里的单例模式有两个应用场景，一个是游乐园类 Casino，通过懒汉式方式创建，提供对外的接口获得游乐园实例进行各种操作；另一个是警报系统 AlarmSystem，虽然全游乐园共用一套警报系统我认为有点不太现实，但从设计的角度上来说式可行的，通过饿汉方式创建警报系统，只提供一个接口来获取这个单例，在设计上是完全没有问题的。