**工厂方法模式**

**模式定义：**

工厂方法模式(Factory Method Pattern)：工厂方法模式又简称为工厂模式，也叫虚拟构造器(Virtual Constructor)模式或者多态模式，属于类的创建型模式。在工厂方法模式中，父类负责定义创建对象的公共接口，而子类则负责生成具体的对象，这样做的目的是将类的实例化操作延迟到子类中完成，即由子类来决定究竟应该实例化(创建) 哪一个类。

**参与者：**

Product：抽象产品

ConcreteProduct：具体产品

Factory：抽象工厂

ConcreteFactory：具体工厂

**工厂方法模式的优点：**

1）在工厂方法模式中，工厂方法用来创建客户所需要的产品，同时还向客户隐藏了哪种具体产品类被实例化这一细节，用户只需要关心所需产品对应的工厂，无需关心创建产品的细节，甚至无需知道产品类的类名。

2）基于工厂角色和产品角色的多态性设计是工厂方法模式的关键。它能够使工厂可以自主确定创建何种产品对象，而如何创建这个对象的细节则完全封装在具体工厂内部。

3）使用工厂方法模式的另一个优点是在系统中加入新产品时，无需修改抽象工厂和抽象产品提供的接口，无需修改客户端，也无需修改其它的具体工厂和具体产品，而只要添加一个新的具体工厂和具体产品即可。

**工厂方法模式的缺点：**

1）在添加新产品时，需要编写新的具体产品类，而且还要提供与之对应的具体工厂类，类成对增加，将导致系统设计较为庞大，且运行时系统会有额外的开销。

2）由于考虑到系统的可扩展性，需要引入抽象层，在客户端代码中均使用抽象层进行定义，增加了系统的抽象性和理解难度，且在实现的时候可能需要用到反射等技术，增加了系统的实现难度。

**建造者模式**

**模式定义：**

1）建造者模式(Builder Pattern)：将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

2）建造者模式是一步一步创建一个复杂的对象，它允许用户只通过指定复杂对象的类型和内容就可以构建它们，用户不需要知道内部的具体构建细节。建造者模式属于对象创建型模式。根据中文翻译的不同，建造者模式又可以称为生成器模式。

**建造者模式角色：**

Builder：抽象建造者

ConcreteBuilder：具体建造者

Director：指挥者

Product：产品角色

**建造者模式的优点：**

1）在建造者模式中，客户端不必知道产品内部组成的细节，将产品本身与产品的创建过程解耦，使得相同的创建过程可以创建不同的产品对象。

2）每一个具体建造者都相对独立，而与其他的具体建造者无关，因此可以很方便地替换具体建造者或增加新的具体建造者，用户使用不同的具体建造者即可得到不同的产品对象。

3）可以更加精细地控制产品的创建过程。将复杂产品的创建步骤分解在不同的方法中，使得创建过程更加清晰，也更方便使用程序来控制创建过程。

4）增加新的具体建造者无须修改原有类库的代码，指挥者类针对抽象建造者类编程，系统扩展方便，符合“开闭原则”。

**建造者模式的缺点：**

1）建造者模式所创建的产品一般具有较多的共同点，其组成部分相似，如果产品之间的差异性很大，则不适合使用建造者模式，因此其使用范围受到一定的限制。

2）如果产品的内部变化复杂，可能会导致需要定义很多具体建造者类来实现这种变化，导致系统变得很庞大。