# 什么是设计模式

设计模式(Design Pattern)是一套被反复使用、多数人知晓的、经过 分类编目的、代码设计经验的总结，使用设计模式是为了可重用代码、让代码更容易被他人理解并且保证代码可靠性。

设计模式一般包含模式名称、问题、目的、解决方案、效果等组成要素，其中关键要素是模式名称、问题、解决方案和效果

# **设计模式的作用**

* 复用解决方案
* 保证代码的可靠性
* 大多数设计模式还能使软件更容易修改和维护。

# **设计模式的原则**

设计模式有7大原则：

* 单一职责原则 (Single Responsibility Principle, SRP) ：一个类只负责一个功能领域中的相应职责，当然，我们可以进行变形，让方法职责单一；
* 接口隔离原则 (Interface Segregation Principle, ISP) ：使用多个专门的接口，而不使用单一的总接口，说白了，接口应该尽可能的小。
* 依赖倒转原则 (Dependence Inversion Principle, DIP)：抽象不应该依赖于细节，细节应该依赖于抽象 ，强调的是面向接口编程；
* 里氏代换原则 (Liskov Substitution Principle, LSP) ：所有引用基类对象的地方能够透明地使用其子类的对象，LSP和继承的关系非常密切；
* 开闭原则 (Open-Closed Principle, OCP) ： 软件实体应对扩展开放，而对修改关闭。
* 合成复用原则 (Composite Reuse Principle, CRP) ：尽量使用对象组合，而不是继承来达到复用的目的
* 迪米特法则 (Law of Demeter, LoD)：一个软件实体应当尽可能少地与其他实体发生相互作用，其中门面模式就很好的展现了迪米特法则；

# **设计模式分类**

## 创建型模式（6个）

* 简单工厂模式
* 工厂方法模式
* 抽象工厂模式
* 单例模式
* 原型模式
* 建造者模式

## 结构型模式（7个）

* 适配器模式
* 桥接模式
* 组合模式
* 装饰模式
* 外观模式
* 享元模式
* 代理模式

## 行为型模式（11个）

* 责任链模式
* 命令模式
* 解释器模式
* 迭代器模式
* 中介者模式
* 备忘录模式
* 观察者模式
* 状态模式
* 策略模式
* 模板方法模式
* 访问者模式

# **简单工厂模式**

## 定义

简单工厂模式(Simple Factory Pattern)：定义一个工厂类，它可以根据参数的不同返回不同类的

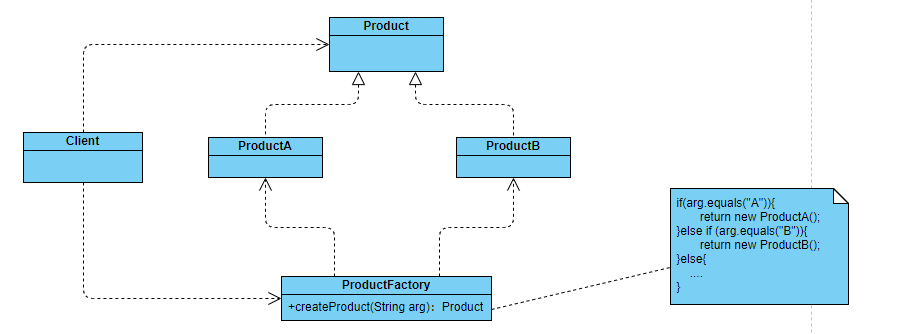
实例，被创建的实例通常都具有共同的父类。因为在简单工厂模式中用于创建实例的方法是

静态(static)方法，因此简单工厂模式又被称为静态工厂方法(Static Factory Method)模式。

## 类图

### 标准类图：

为了让工厂能够统一生产对象，这些对象通常会有一个公共的父类，这个类多半被定义成为抽象类或者是接口（当然这个不是必须的），如下Product，然后让各个具体的类去具体实现或者继承Product。



### 简化类图

为了简化简单工厂，可以将抽象的产品类和工厂类合并在一起，形成如下比较简单的方式，由于工厂需要被实例化，否则不能使用，所以Product不再是抽象类或者接口。



## 总结

### 优点

1. 工厂类包含必要的判断逻辑，可以决定在什么时候创建哪一个产品类的实例，客户端可以免除直接创建产品对象的职责，而仅仅“消费”产品，简单工厂模式实现了对象创建和使用的分离。
2. 客户端无须知道所创建的具体产品类的类名，只需要知道具体产品类所对应的参数即可，对于一些复杂的类名，通过简单工厂模式可以在一定程度减少使用者的记忆量。
3. 可以通过配置文件参数的方式，在不修改代码的情况下传入arg，从而灵活的构建出Product对象；

### 缺点

1. 由于工厂类集中了所有产品的创建逻辑，职责过重，一旦不能正常工作，整个系统都要受到影响。
2. 系统扩展困难，一旦添加新产品就不得不修改工厂逻辑，在产品类型较多时，有可能造成工厂逻辑过于复杂，不利于系统的扩展和维护。
3. 简单工厂模式由于使用了静态工厂方法，造成工厂角色无法形成基于继承的等级结构。

## 开源项目案例