设计模式学习笔记

2016-12-01

**6大原则：**

**单一职责：**一个类只做一件事，这个需要根据程序员的经验来判断，不同程序员对单一职责的评判标准不同。

**开闭原则：**设计的类对扩展开放，对修改关闭

**里氏替换原则：**所有的基类出现的地方可以用他的子类来代替

**依赖倒转原则：**针对接口编程，依赖于抽象而不是依赖于具体实现（自定义类的成员变量最好是接口）

**接口隔离原则：**使用多个隔离的接口比使用单个接口要好

**迪米特法则：**一个实体应当尽量少的与其他实体之间发生相互作用，使得系统模块相对独立

**聚合复用原则：**应尽量使用组合聚合，少使用继承

2016-11-30

**工厂方法模式**

工厂方法模式分为三种：

**普通工厂模式**，就是建立一个工厂类，对实现了同一接口的一些类进行实例的创建。

***多个工厂方法模式***，是对普通工厂方法模式的改进，在普通工厂方法模式中，如果传递的字符串出错，则不能正确创建对象，而多个工厂方法模式是提供多个工厂方法，分别创建对象。

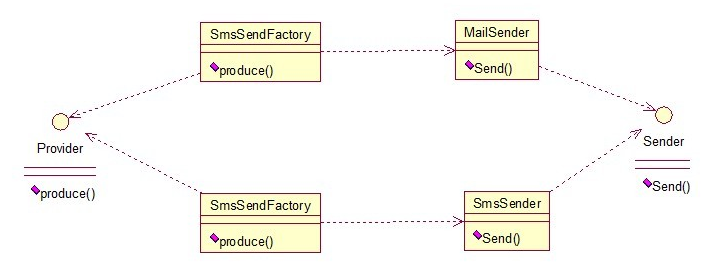
***静态工厂方法模式***，将上面的多个工厂方法模式里的方法置为静态的，不需要创建实例，直接调用即可。

***总体来说***，工厂模式适合：凡是出现了大量的产品需要创建，并且具有共同的接口时，可以通过工厂方法模式进行创建。在以上的三种模式中，第一种如果传入的字符串有误，不能正确创建对象，第三种相对于第二种，不需要实例化工厂类，所以，大多数情况下，我们会选用第三种——静态工厂方法模式。

**抽象工厂模式**

对于工厂方法模式来说，当新加一个接口的实现，则要修改工程类，违背了开闭原则，为了解决这个问题，抽象工厂模式便产生了

每一个接口实现对应一个工厂，全部的工厂都继承自**抽象的接口（也就是对工厂的抽象）**



**单例模式**

单例对象（Singleton）是一种常用的设计模式。在Java应用中，单例对象能保证在一个JVM中，该对象只有一个实例存在。

public static Singleton getInstance() {

        if (instance == null) {

            synchronized (instance) {

                if (instance == null) {

                    instance = new Singleton();

                }

            }

        }

        return instance;

    }

静态内部类实现

private static class SingletonFactory{

        private static Singleton instance = new Singleton();

    }

    public static Singleton getInstance(){

        return SingletonFactory.instance;

    }