工厂模式 - 简单工厂模式

寂然

分类介绍

- 简单工厂模式
- 工厂方法模式
- 抽象工厂模式

披萨订购的案例 --》一般实现方式 优化 --》简单工厂模式   升级--》工厂方法模式

案例引出 - 披萨订购

> 有这样一个披萨店的需求，披萨的种类很多(比如 GreekPizza、CheesePizza 等)
>
> 披萨的制作有 prepare，bake，cut，box 等
>
> 要求：完成披萨店订购功能，便于披萨种类的扩展，便于维护

?
? ?

? ? ?

解决方案一 - 一般实现方式

方案分析

- 优势：容易理解
- 劣势：违反了开闭原则？ 调用方

- 假设:新增一个种类的披萨?

```
//提供方
public class FruitPizza extends Pizza{
    @Override
    public void prepare() {
        System.out.println("准备水果披萨的原材料");
    }
}

//调用方
else if (orderType.equals("FruitPizza")){
                pizza = new FruitPizza();
                pizza.setName("FruitPizza");
            }
```

改进思路

基本介绍 - 简单工厂模式

> 简单工厂模式：定义了一个创建对象的类，由这个类来封装实例化对象的行为，在简单工厂模式中，可以根
>
> 据参数的不同返回不同类的实例

解决方案二 - 简单工厂模式

简单工厂模式：定义一个创建对象的类（工厂类）--》负责封装实例化对象的行为

```
       if (orderType.equals("GreekPizza")){

            pizza = new GreekPizza();

            pizza.setName("GreeKPizza");
        } else if (orderType.equals("FruitPizza")){

            pizza = new FruitPizza();

            pizza.setName("FruitPizza");
        } else if (orderType.equals("CheesePizza")){

            pizza = new CheesePizza();
```

```
            pizza.setName("CheesePizza");
        }
```

```
package com.kkb.factorypattern.simplefactory.order;

import com.kkb.factorypattern.simplefactory.pizza.Pizza;

import java.util.Scanner;

/**
 * @Classname OrderPizza
 * @Created by 寂然
 * @Description 订购披萨的业务逻辑 调用方
 */
public class OrderPizza {

    SimpleFactory simpleFactory;

    public void setSimpleFactory(SimpleFactory simpleFactory) {
        this.simpleFactory = simpleFactory;

        while (true){

            orderType = getType();

            pizza = this.simpleFactory.createPizza(orderType);

            if (pizza != null){//订购成功

                pizza.prepare();
                pizza.bake();
                pizza.cut();
                pizza.box();

            } else {

                System.out.println("抱歉，这里没有你想要的披萨");
                break;
            }
        }


    }

    Pizza pizza = null;

    String orderType; //订购披萨的类型

    public OrderPizza(SimpleFactory simpleFactory){

        setSimpleFactory(simpleFactory);
    }
```

```
    //获取客户希望的披萨类型
    public String getType(){

        System.out.println("请输入你要订购的披萨类型");

        Scanner scanner = new Scanner(System.in);
        String next = scanner.next();
        return next;
    }
}
```

方案分析

使用简单工厂模式，工厂类是只有一个，只需要改动这一个地方

避免了业务逻辑的代码和创建对象的代码大量耦合 --》降低耦合

让代码更加条理清晰，更加优雅

静态工厂模式 明确一下

披萨案例 --》升级 --》工厂方法模式 抽象工厂模式 一起对比分析