



链滴

设计模式:Abstract Factory(抽象工厂)-- 创建型模式

作者: [sirwsl](#)

原文链接: <https://ld246.com/article/1591780103806>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

一、介绍:

提供一个接口以创建一系列相关或相互依赖得对象，而无需指定他们具体得类。也就是说用户在使用过程中，通过接口创建一系列的组件，而并不需要知道哪些类实现了哪些组件。

用户只与抽象接口进行交互而不适用具体的类接口。

二、适用性

- 一个系统要独立与它的产品创建、组合和表示。
- 一个系统要由多个产品系列中的一个来配置。
- 要对一系列相关的产品对象进行创建以便联合使用。
- 提供一个产品类库，但只是想显示他们的接口而不是实现。

三、参与者

AbstractFactory

- 声明创建抽象产品的操作接口

ConcreteFactory

- 实现接口，创建具体的产品对象的操作

AbstractProduct

- 声明产品的接口

ConcentrateProduct

- 实现AbstractProduct接口
- 定义一个被具体工厂创建的产品对象

Client

- 使用AbstractFactory与AbstractProduct声明的接口

四、效果

- 分离了具体的类

抽象工厂模式封制了创建产品对象的过程，能够控制应用创建对象的类，实现了客户与类分类的原则使得产品的类名在具体工厂中被隔离。

- 易于交换唱片的系列

一个具体工厂在一个应用中仅在初始化的时候得到，这使得改变一个应用的具体工厂很容易、在改变过程中只需要改变工厂就可使用不同产品的配置

- 有利于产品的一致性

当一个系列中的产品被设计在一起工作的时候，一个应用只能能够只用同一个系列中的对象。

- 难以支持新种类的产品

抽象工厂接口确定了可以被创建产品的集合，想要增加新品种时难以扩展

五、实现

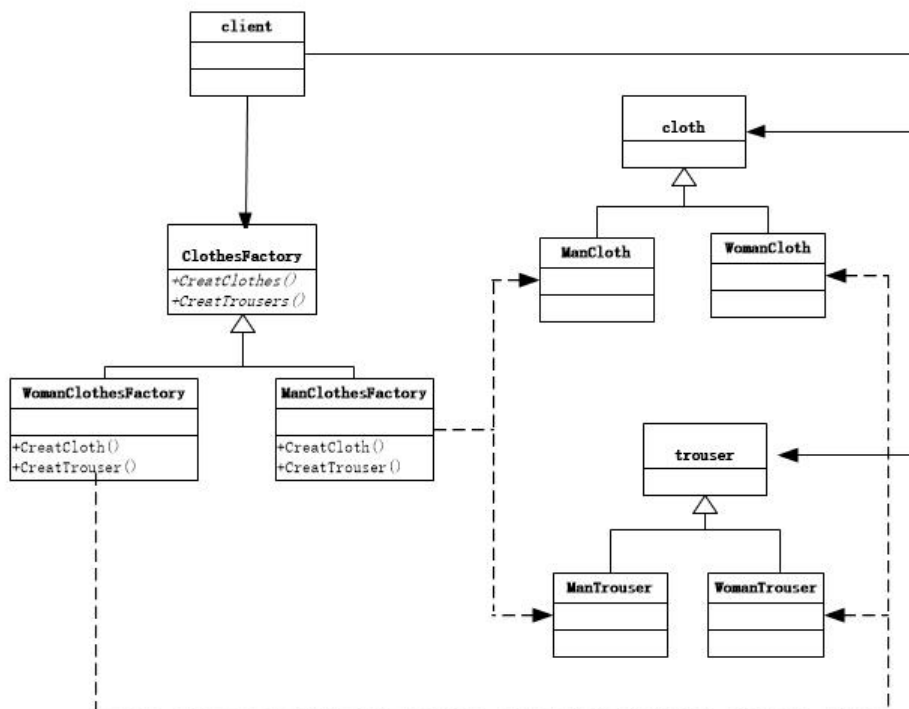
- 1.将工厂作为单件
- 2.创建产品
- 3.定义可拓展的工厂

1. 场景

有一个生产服装的工厂，但是服装又有裤子与衣服的分，而男性与女性也是不一样的。而我们都知所有的衣服各裤子都有共性，但是工厂在生产时需要在不同的车间中进行生产。

也就是，工厂总部（trouser）只需要知道每个部门名字（manTrousers、WomanTrousers）就可以让他们创建产品（cloth），并不需要关注他们具体是怎么实现的。

2. UML建模



3. 编码：

1. 创建产品的接口

创建一个衣服接口

```
public interface cloth {  
    void clothStyle();  
}
```

创建一个裤子的接口

```
public interface trouser {  
    void trouserStyle();  
}
```

2. 产品接口实现

男士衣服接口实现

```
public class manCloth implements cloth{  
  
    @Override  
    public void clothStyle() {  
        System.out.println("男士衣服样式为-红色西服");  
    }  
}
```

女士衣服的接口实现

```
public class womanCloth implements cloth{  
    @Override  
    public void clothStyle() {  
        System.out.println("女士衣服样式为-白色T恤");  
    }  
}
```

男士裤子的接口实现

```
public class manTrouser implements trouser{  
    @Override  
    public void trouserStyle() {  
        System.out.println("男士裤子-样式为黑色西裤");  
    }  
}
```

女士裤子的接口实现

```

public class womanTrousar implements trouser{
    @Override
    public void trouserStyle() {
        System.out.println("女士裤子样式为-白色裙子");
    }
}

```

3. 创建抽象工厂接口

```

public interface clothesFactory {
    //衣服接口
    cloth getCloth();
    //裤子接口
    trouser getTrousar();
}

```

4. 抽象工厂实现

男士服装工厂

```

public class manClothesFactory implements clothesFactory{
    @Override
    public cloth getCloth() {
        return new manCloth();
    }

    @Override
    public trouser getTrousar() {
        return new manTrousar();
    }
}

```

女士服装工厂

```

public class womanClothesFactory implements clothesFactory{
    @Override
    public cloth getCloth() {
        return new womanCloth();
    }

    @Override
    public trouser getTrouser() {
        return new womanTrouser();
    }
}

```

5. 使用工厂生产

```

public class Demo {
    public static void main (String[] args){
        clothesFactory man = new manClothesFactory();
        clothesFactory woman = new womanClothesFactory();
        cloth mc=man.getCloth();
        trouser mt=man.getTrouser();
        cloth wc = woman.getCloth();
        trouser wr = woman.getTrouser();
        mc.clothStyle();
        mc.clothStyle();
        wc.clothStyle();
        wr.trouserStyle();
    }
}

```

结果:

```

男士衣服样式为-红色西服
男士衣服样式为-红色西服
女士衣服样式为-白色T恤
女士裤子样式为-白色裙子

```

4. 源码: 参见我的github中DesignPatten

- <https://github.com/sirwsl/DesignPatten>