软件体系结构

《软件体系结构作业十三》

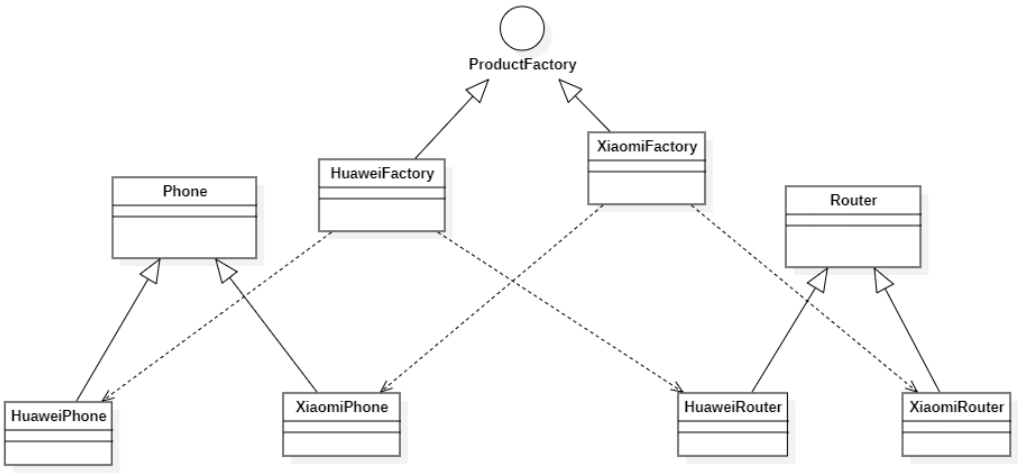
学　号 22920212204396

姓　名 黄子安

2024 年 5 月 6 日

一、阅读 Abstract Factory 的例子的代码，举例说明使用 Abstract Factory 模式的其他应用

先给出类图，如下所示总共有两家公司，他们都生产手机和路由器，华为手机和小米手机在抽象意义上都是手机，华为路由器和小米路由器在抽象意义上都是路由器，在这个案例中，通过抽象工厂模式可以实现产品的抽象，实现对产品族中相关联的多等级产品共同管理，而不必专门引入多个新的类来进行各自管理



再给出代码，首先给出两种产品和他们的抽象表示

```
public interface Phone {
    void callup();
    void sendSMS();
}
```

```
public class HuaweiPhone implements Phone{

    @Override
    public void callup() {
        System.out.println("华为手机打电话");
    }

    @Override
    public void sendSMS() {
        System.out.println("华为手机发短信");
    }
}
```

```java
public class XiaomiPhone implements Phone{

    @Override
    public void callup() {
        System.out.println("小米手机打电话");
    }

    @Override
    public void sendSMS() {
        System.out.println("小米手机发短信");
    }
}
```

再给出对应的抽象工厂

```java
public interface ProductFactory {
    Phone getPhone();
    Router getRouter();
}
```

```java
public class HuaweiFactory implements ProductFactory {
    @Override
    public Phone getPhone() {
        return new HuaweiPhone();
    }

    @Override
    public Router getRouter() {
        return new HuaweiRouter();
    }
}
```

```java
public class XiaomiFactory implements ProductFactory {

    @Override
    public Phone getPhone() {
        return new XiaomiPhone();
    }

    @Override
    public Router getRouter() {
        return new XiaomiRouter();
    }
}
```

java

最后编写一个客户端用于测试

```
public class Main {
    public static void main(String[] args) {
        System.out.println("=============小米系列=============");
        XiaomiFactory xiaomiFactory = new XiaomiFactory();
        Phone phone = xiaomiFactory.getPhone();
        phone.callup();
        phone.sendSMS();
        Router router = xiaomiFactory.getRouter();
        router.openWifi();

        System.out.println("=============华为系列=============");
        HuaweiFactory huaweiFactory = new HuaweiFactory();
        phone = huaweiFactory.getPhone();
        phone.callup();
        phone.sendSMS();
        router = huaweiFactory.getRouter();
        router.openWifi();
    }
}
```

运行结果如下所示：

```
D:\Java17\bin\java.exe "-javaagent:D:\IdeaU\IntelliJ IDEA 2023.1.3\lib\idea_rt.jar=58359:D:\Ideal
=============小米系列=============
小米手机打电话
小米手机发短信
小米路由器打开WiFi
=============华为系列=============
华为手机打电话
华为手机发短信
华为路由器打开WiFi

Process finished with exit code 0
```

由此可以通过对产品的抽象可以避免工厂方法所产生子类工厂过多的问题，保证代码的简洁与逻辑清晰，对整个产品族进行扩展也会比较方便，比如新加入一个苹果公司造手机和路由器只需要创建一个新的 AppleFactory 即可，符合开闭原则，无需修改之前的代码同时又能保证结构清晰