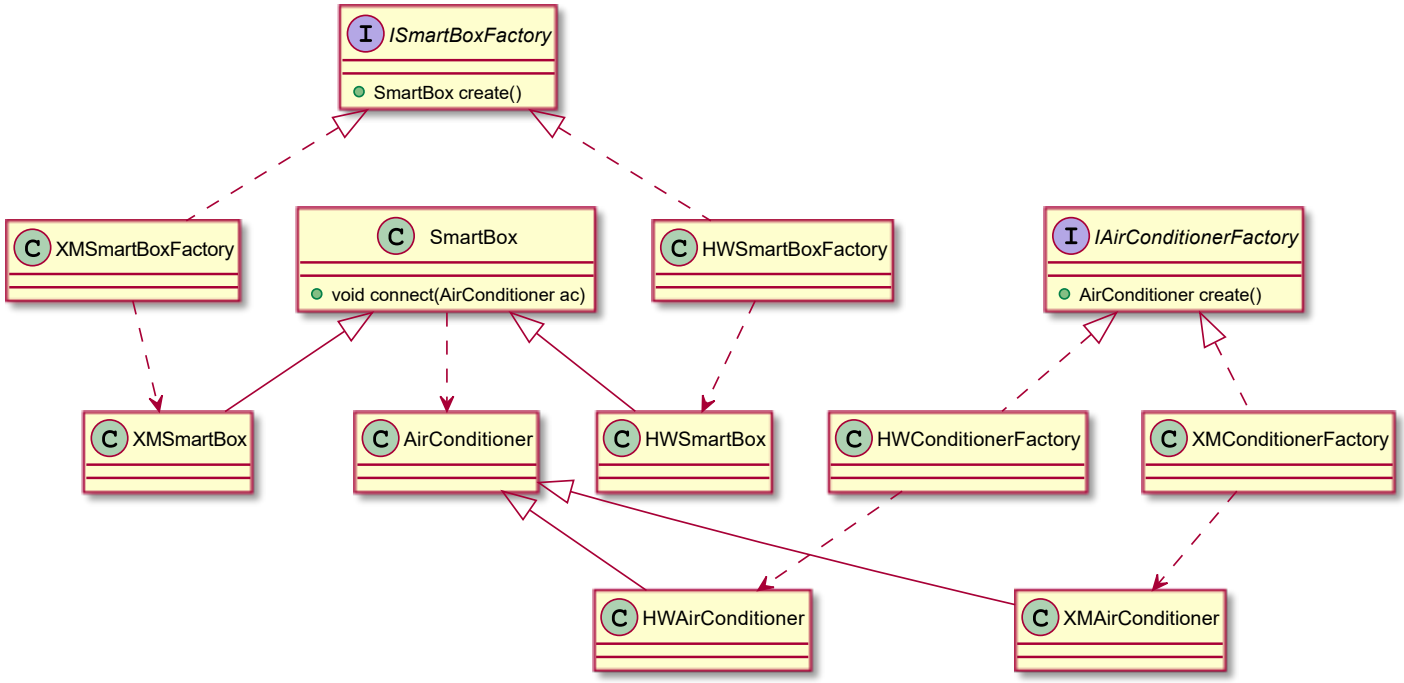
第2章 抽象工厂模式

提出问题

> 问题描述：用户需要布置智能家居设备，例如通过智能音箱（SmartBox）连接（connect）其他智能设备（例如空调AirConditioner），假设现在市场上主流的厂家有小米、华为、海信等，而且相同厂家的设备才能互联。请设计程序模拟该应用场景。

这里涉及多个厂家的两种产品`SmartBox`和`AirConditioner`，`SmartBox`具备连接其他设备的功能，根据工厂方法模式进行设计，类图如下：



编写客户端代码，模拟购买智能设备并进行互联：

```
public class Test{
    public static void main(String[] args){
        ISmartBoxFactory sboxf = new XMSmartBoxFactory();
        IAirConditionerFactory acf =  new XMConditionerFactory();
        SmartBox box = sboxf.create();
        AirConditioner ac = acf.create();
        box.connect(ac);
    }
}
```

这里存在一个问题，客户端需要对设备的适配进行管理，例如要创建相同厂家的工厂进行生产的设备才能互联。如何将产品之间的关联关系内化到底层设计中，从而降低应用层对底层模块的使用难度？本节的抽象工厂模式能解决这个问题。

模式名称

抽象工厂：`Abstract Method` 或 `Kit`

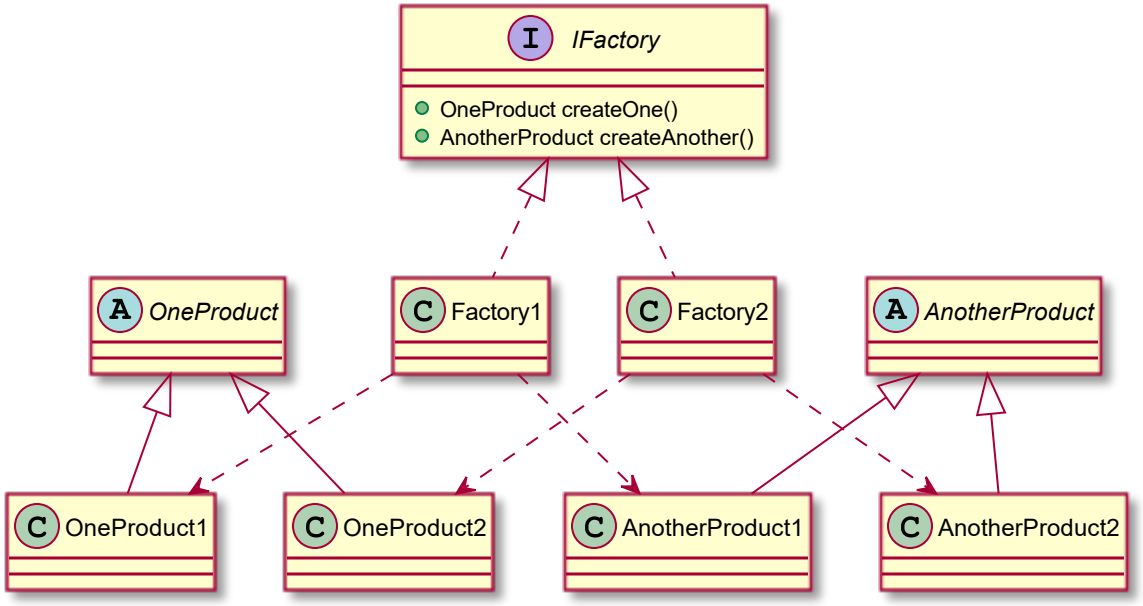
设计意图

提供一个创建一系列相关或相互依赖对象的接口，而无须预先定义或指定它们具体的类。该模式增加了产品簇（Product Family），同一个产品簇中的不同类型之间的产品相互关联或依赖。

> Provide an interface for creating families of related or dependent objects without specifying their concrete classes.

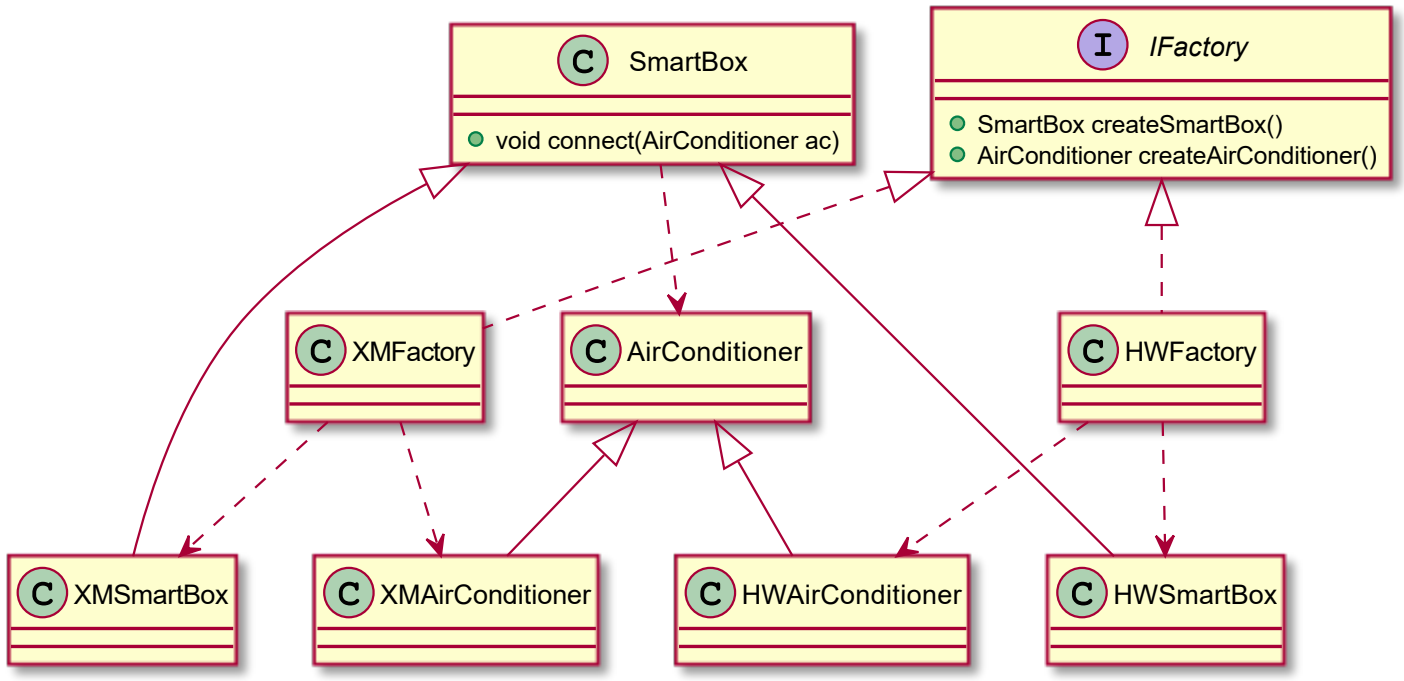
设计结构

相对于工厂方法模式，抽象工厂模式中的工厂类接口为产品簇中的每一类产品定义一个创建方法，也就是说一个工厂需要生产一系列产品。产品方面，多了产品簇上的类型维度。



解决问题

根据抽象工厂模式的定义和结构修改原程序类图结构为：



客户端程序修改为：

```
public class Test {
        public static void main(String[] args) {
                IFactory xmf = new XMFactory();
                SmartBox sbox = xmf.createSmartBox();
                AirConditioner ac = xmf.createAirConditioner();
                sbox.connect(ac);
        }
}
```

可以看出，同厂家的产品均由一个工厂生产，使得产品簇中的各类产品更顺利地一起工作，客户端也不必刻意去管理设备适配问题。

效果与适用性

效果：

- 继承了工厂方法的一些优点，降低客户端与产品类之间的关联；
- 有利于保证产品的一致性，促进同一家族产品在一起能顺利协作；
- 产品簇易于扩展，但产品簇中产品类型扩展时需要修改原接口和工厂类。

适用性：

- 当系统需要由多个产品簇中的一个来配置时；
- 当产品簇中的产品被设计为在一起协同工作时；
- 当需要提供一个产品库，对外只提供应用接口而隐藏实现细节时；
- 当系统独立与具体产品的创建、组成和表示。