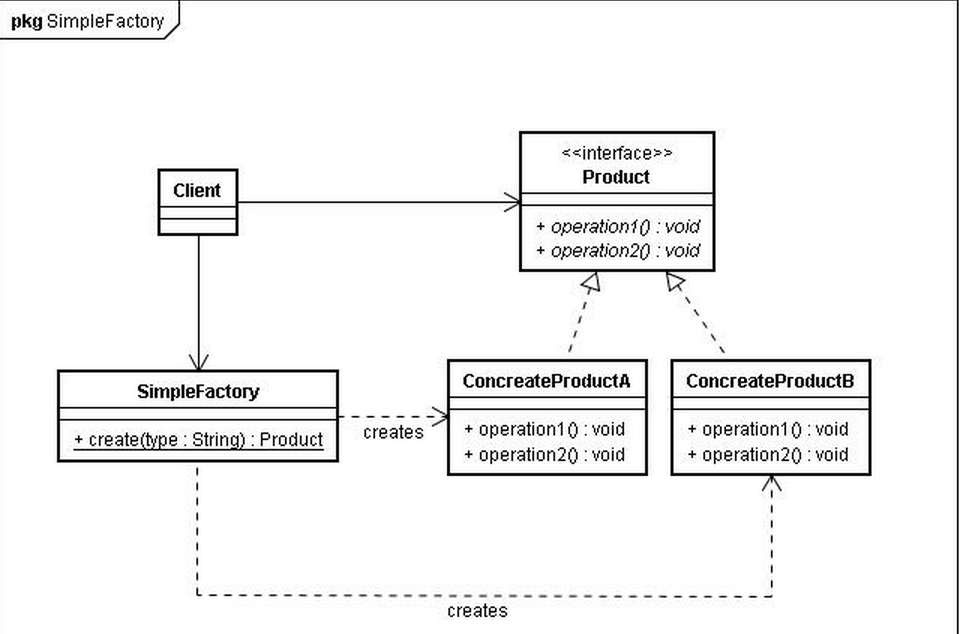
工厂模式：专门负责将大量有共同接口的类实例化。工厂模式可以动态决定将哪一个类实例化，不必事先知道每次要实例化哪一个类。工厂模式的几种形态：

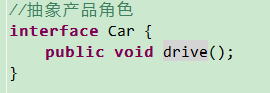
1. 简单工厂(Simple Factory)模式，又称静态工厂方法模式(Static Factory Method Pattern)
2. 简单工厂模式：它属于类创建型模式。在简单工厂模式中，可以根据自变量的不同返回不同类的实例。简单工厂模式专门定义一个类来负责创建其他类的实例，被创建的实例通常都具有共同的父类。
3. 类图



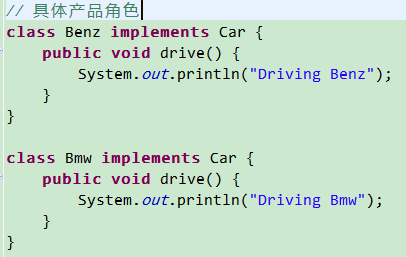
* 1. 工厂类（SimpleFactory）角色：担任这个角色的是工厂方法模式的核心，含有与应用紧密相关的商业逻辑。工厂类在客户端的直接调用下创建产品对象，它往往由一个具体Java 类实现。
  2. 抽象产品（Product）角色：担任这个角色的类是工厂方法模式所创建的对象的父类，或它们共同拥有的接口。抽象产品角色可以用一个Java 接口或者Java 抽象类实现。
  3. 具体产品（Concrete Product）角色：工厂方法模式所创建的任何对象都是这个角色的实例，具体产品角色由一个具体Java 类实现。

1. 例

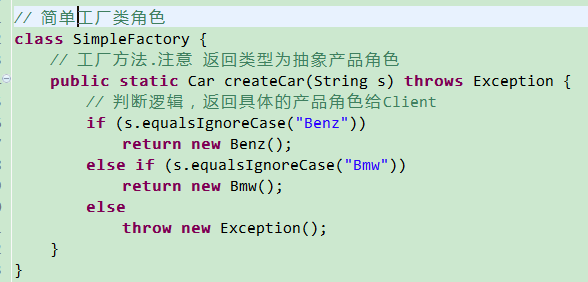
定义Product接口



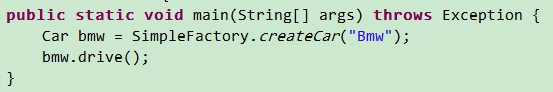
定义具体ConcreteProductX



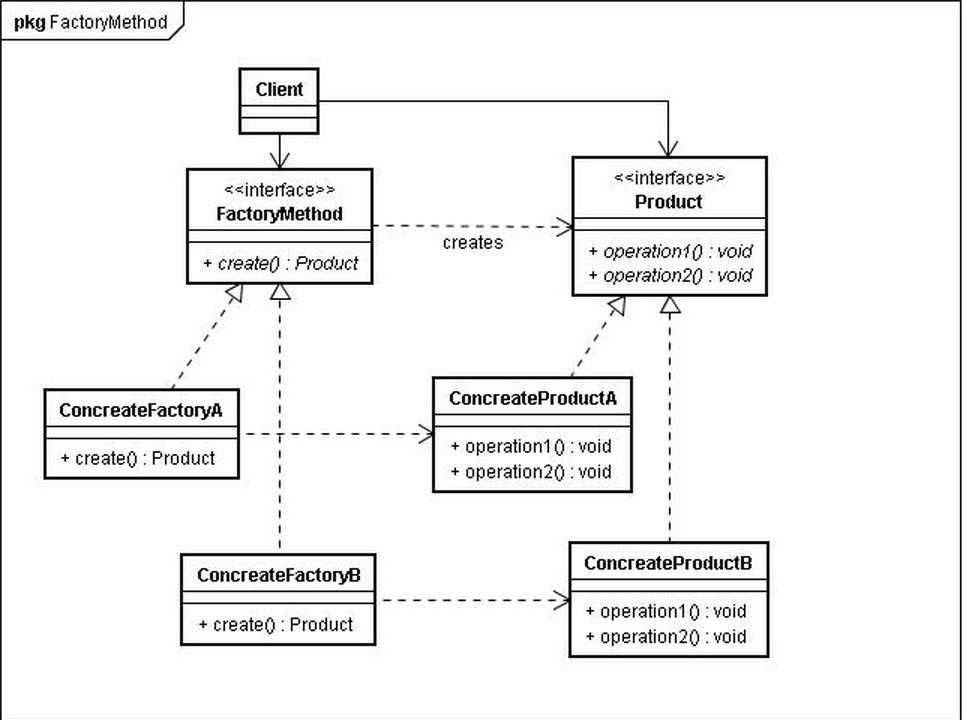
定义简单工厂类



定义Client测试

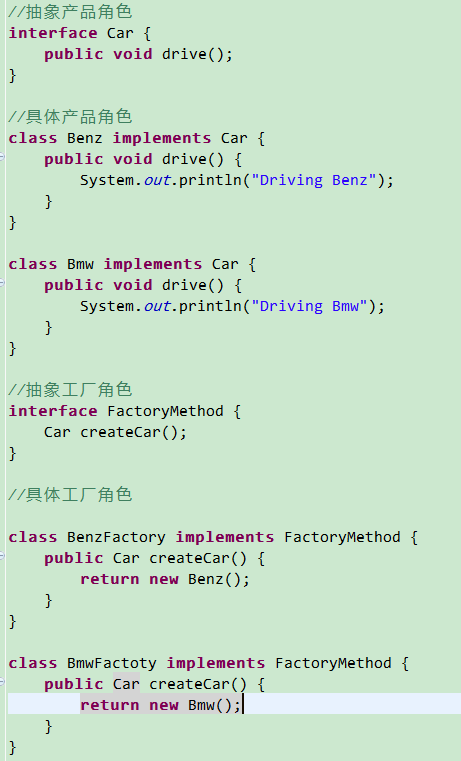


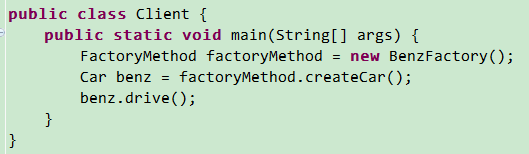
1. ：工厂方法(Factory Method)模式，又称多态性工厂(Polymorphic Factory)模式或虚拟构造子(Virtual Constructor)模式
2. 工厂方法模式：定义一个用于创建对象的接口，让子类决定实例化哪一个类，工厂方法使一个类的实例化延迟到其子类。
3. 类图



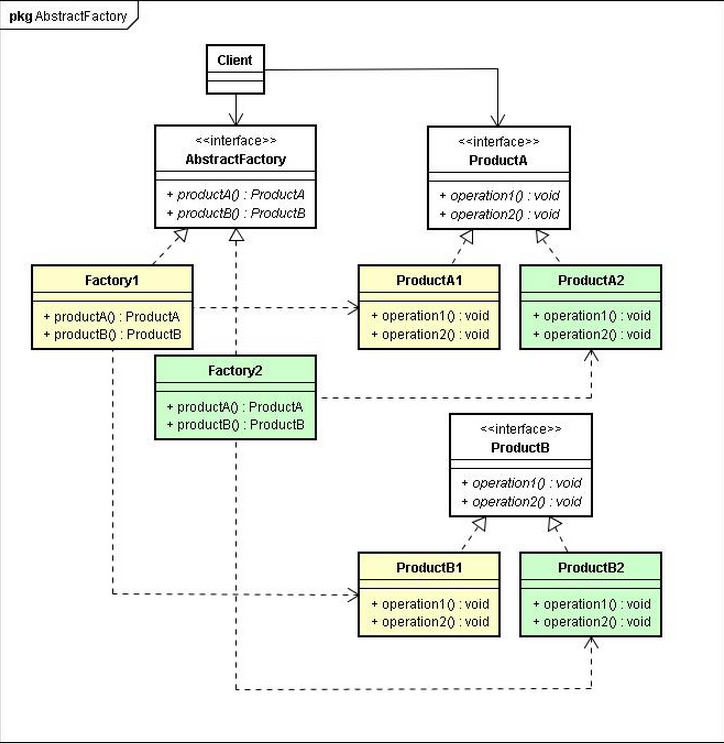
* 1. 抽象工厂（FactoryMethod）角色：担任这个角色的是工厂方法模式的核心，它是与应用程序无关的。任何在模式中创建对象的工厂类必须实现这个接口。在实际的系统中，这个角色也常常使用抽象Java 类实现。
  2. 具体工厂（ConcreteFactoryX）角色：担任这个角色的是实现了抽象工厂接口的具体Java 类。具体工厂角色含有与应用密切相关的逻辑，并且受到应用程序的调用以创建产品对象。
  3. 抽象产品（Product）角色：工厂方法模式所创建的对象的超类型，也就是产品对象的共同父类或共同拥有的接口。在实际的系统中，这个角色也常常使用抽象Java 类实现。
  4. 具体产品（ConcreteProductX）角色：这个角色实现了抽象产品角色所声明的接口。工厂方法模式所创建的每一个对象都是某个具体产品角色的实例。

1. 例

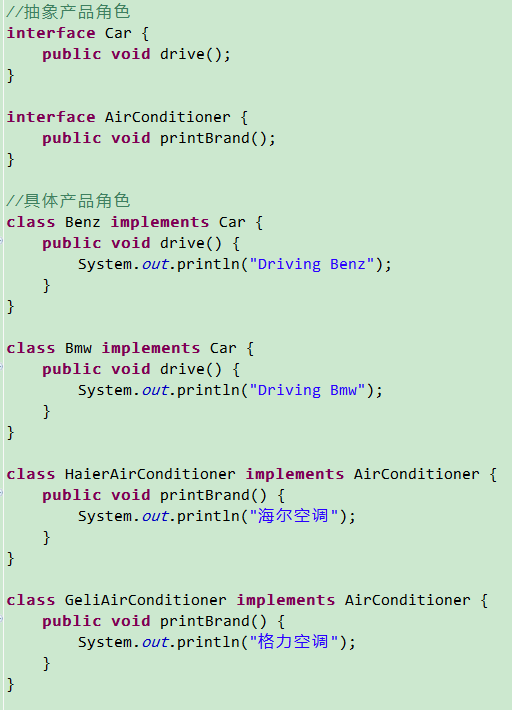


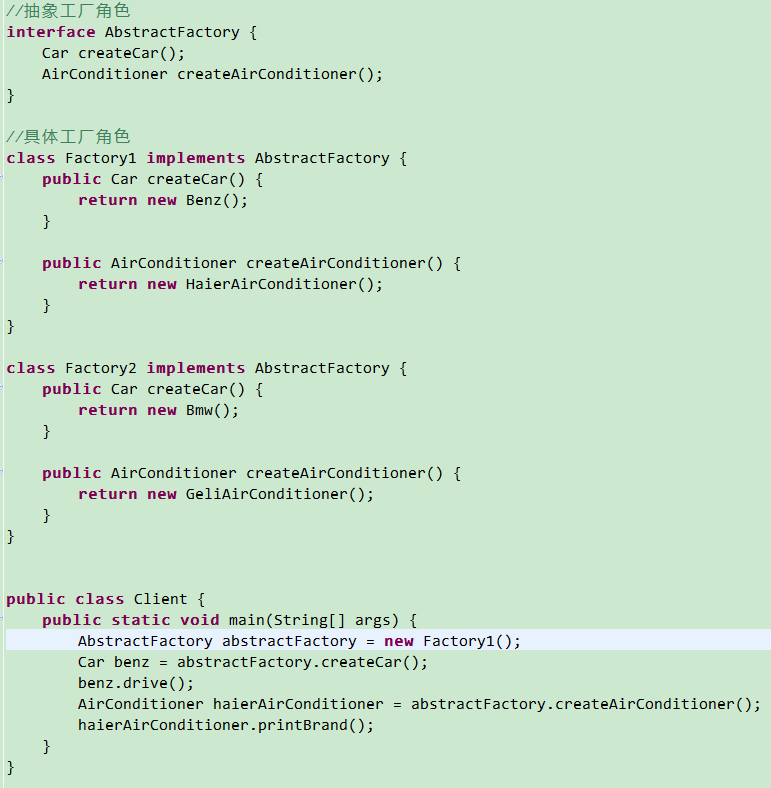


1. 抽象工厂(Abstract Factory)模式，又称工具箱(Kit或Toolkit)模式
2. 抽象工厂模式提供一个创建一系列或相互依赖的对象的接口，而无需指定它们具体的类。
3. 类图



1. 例





1. JDK中java.util.Calendar就使用了简单工厂模式. java.util.Calendar#getInstance(java.util.TimeZone, java.util.Locale)