一、抽象工厂模式（**Abstract Factory**）

抽象工厂模式提供一个创建一系列相关或相互依赖对象的接口，而无需指定它们具体的类。

抽象工厂（Abstract Factory）模式，又称工具箱（Kit 或Toolkit）模式。

二、创建过程如下

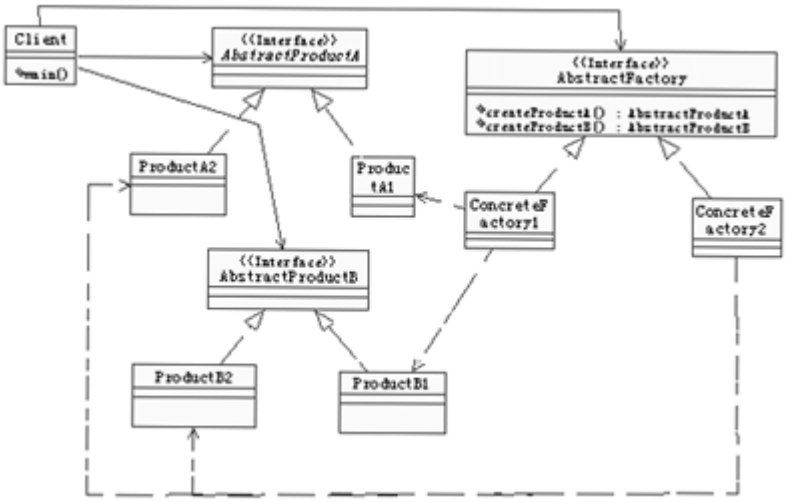
一个具体工厂创建一个产品族，一个产品族是不同系列产品的组合，产品的创建的逻辑分在在每个具体工厂类中。所有的具体工厂继承自同一个抽象工厂。

客户端创建不同产品族的工厂，产品族的工厂创建具体的产品对客户端是不可见的。

增加新的产品族时，需要增加具体工厂类,符合OCP原则。

增加新产品时，需要修改具体工厂类和增加产品类，不符合OCP原则

如果没有应对"多系列对象创建"的需求变化，则没有必要使用抽象工厂模式，这时候使用简单的静态工厂完全可以。



三、一个简单的实例

```
// 产品 Plant接口
public interface IPlant { }
//具体产品PlantA，PlantB
public class PlantA implements IPlant {

public PlantA () {
System.out.println("create PlantA !");
}

public void doSomething() {
System.out.println(" PlantA do something ...");
}
}
public class PlantB implements IPlant {
public PlantB () {
System.out.println("create PlantB !");
}

public void doSomething() {
System.out.println(" PlantB do something ...");
}
}
// 产品 Fruit接口
```

```
public interface IFruit { }
//具体产品FruitA，FruitB
public class FruitA implements IFruit {
public FruitA() {
System.out.println("create FruitA !");
}
public void doSomething() {
System.out.println(" FruitA do something ...");
}
}
public class FruitB implements IFruit {
public FruitB() {
System.out.println("create FruitB !");
}
public void doSomething() {
System.out.println(" FruitB do something ...");
}
}
// 抽象工厂方法
public interface AbstractFactory {
public IPlant createPlant();
public IFruit createFruit() ;
}
//具体工厂方法
public class FactoryA implements AbstractFactory {
public IPlant createPlant() {
return new PlantA();
}
public IFruit createFruit() {
return new FruitA();
}
}
public class FactoryB implements AbstractFactory {
public IPlant createPlant() {
return new PlantB();
}
public IFruit createFruit() {
return new FruitB();
}
}
```