## 4. 抽象工厂（Abstract Factory）

### Intent

提供一个接口，用于创建 **相关的对象家族** 。

### Class Diagram

抽象工厂模式创建的是对象家族，也就是很多对象而不是一个对象，并且这些对象是相关的，也就是说必须一起创建出来。而工厂方法模式只是用于创建一个对象，这和抽象工厂模式有很大不同。

抽象工厂模式用到了工厂方法模式来创建单一对象，AbstractFactory 中的 createProductA() 和 createProductB() 方法都是让子类来实现，这两个方法单独来看就是在创建一个对象，这符合工厂方法模式的定义。

至于创建对象的家族这一概念是在 Client 体现，Client 要通过 AbstractFactory 同时调用两个方法来创建出两个对象，在这里这两个对象就有很大的相关性，Client 需要同时创建出这两个对象。

从高层次来看，抽象工厂使用了组合，即 Cilent 组合了 AbstractFactory，而工厂方法模式使用了继承。

### Implementation

public class AbstractProductA {
}

public class AbstractProductB {
}

public class ProductA1 extends AbstractProductA {
}

public class ProductA2 extends AbstractProductA {
}

public class ProductB1 extends AbstractProductB {
}

public class ProductB2 extends AbstractProductB {
}

public abstract class AbstractFactory {
 abstract AbstractProductA createProductA();
 abstract AbstractProductB createProductB();
}

public class ConcreteFactory1 extends AbstractFactory {
 AbstractProductA createProductA() {
 return new ProductA1();
 }

 AbstractProductB createProductB() {
 return new ProductB1();
 }
}

public class ConcreteFactory2 extends AbstractFactory {
 AbstractProductA createProductA() {
 return new ProductA2();
 }

 AbstractProductB createProductB() {
 return new ProductB2();
 }
}

public class Client {
 public static void main(String[] args) {
 AbstractFactory abstractFactory = new ConcreteFactory1();
 AbstractProductA productA = abstractFactory.createProductA();
 AbstractProductB productB = abstractFactory.createProductB();
 // do something with productA and productB
 }
}

### JDK

* [javax.xml.parsers.DocumentBuilderFactory](http://docs.oracle.com/javase/8/docs/api/javax/xml/parsers/DocumentBuilderFactory.html)
* [javax.xml.transform.TransformerFactory](http://docs.oracle.com/javase/8/docs/api/javax/xml/transform/TransformerFactory.html#newInstance--)
* [javax.xml.xpath.XPathFactory](http://docs.oracle.com/javase/8/docs/api/javax/xml/xpath/XPathFactory.html#newInstance--)