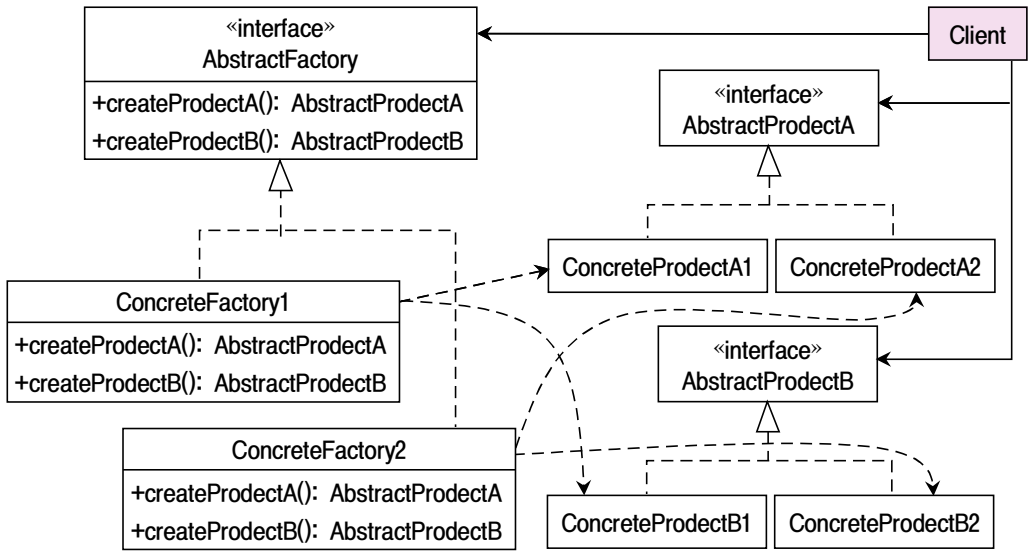
5. 추상 팩토리 패턴

추상 팩토리 패턴(abstract factory pattern) : **구성요소** 별로 '**객체의 집합**'을 생성

〈추상 팩토리 패턴 클래스 다이어그램〉



- 추상 팩토리 패턴은 **서로 관련이 있는 구성요소** 별로 **객체 집합을 생성**한다.
- 서로 독립적인 객체들의 집합을 생성할 수 있는 인터페이스를 제공하는 패턴이다.
- 많은 수의 연관된 **자식클래스**를 특정 그룹으로 묶어 한번에 교체할 수 있도록 한다.

추상 팩토리 적용 예	키보드, 마우스, 모니터를 묶은 전자 **제품군**이 있고 이들 제품이 **삼성전자** 제품이냐? **애플** 제품이냐? 등에 따라 제품 집합이 여러 브랜드명으로 구분될 때, 복잡하게 묶이는 제품군들 **관리** 및 **확장**이 용이하도록 구현한 것이 추상 팩토리 패턴이다. 추상 팩토리 패턴은 **제품군 집합**을 타입별로 객체를 생성할 수 있다.

	팩토리 메서드 패턴	**추상 팩토리 패턴**
공통점	객체 생성 과정을 추상화한 **인터페이스** 제공 객체 생성을 캡슐화하여 구체적인 타입을 감추고 **느슨한 결합구조** 제공	
차이점	객체 생성을 다른 클래스 **위임**	**연관**된 여러 종류의 객체 생성(**제품군**)
	포커스를 **메서드**에 맞춤	포커스를 **클래스**에 맞춤

예제	이전 페이지에 있는 추상 팩토리 패턴의 클래스 다이어그램을 코딩한 것

```
interface AbstractProductA { }     //제품군 A
class ConcreteProductA1 implements AbstractProductA { }
class ConcreteProductA2 implements AbstractProductA { }

interface AbstractProductB { }     //제품군 B
class ConcreteProductB1 implements AbstractProductB { }
class ConcreteProductB2 implements AbstractProductB { }

interface AbstractFactory {          //공장 클래스
  AbstractProductA createProductA();       //제품군 A
  AbstractProductB createProductB();       //제품군 B
}
class ConcreteFactory1 implements AbstractFactory { //제1공장-제품 A1, B1 생산
  public AbstractProductA createProductA(){ return new ConcreteProductA1(); }
  public AbstractProductB createProductB(){ return new ConcreteProductB1(); }
}
class ConcreteFactory2 implements AbstractFactory { //제2공장-제품 A2, B2 생산
  public AbstractProductA createProductA(){ return new ConcreteProductA2(); }
  public AbstractProductB createProductB(){ return new ConcreteProductB2(); }
}

public class Client {
  public static void main(String[] args) {
    AbstractFactory factory = null;          //공장 건설
    factory = new ConcreteFactory1();    //제1공장 가동-제품 A1, B1 생산
    AbstractProductA A1 = factory.createProductA();
    System.out.println("A1");
    AbstractProductB B1 = factory.createProductB();
    System.out.println("B1");
    factory = new ConcreteFactory2();    //제2공장 가동-제품 A2, B2 생산
    AbstractProductA A2 = factory.createProductA();
    System.out.println("A2");
    AbstractProductB B2 = factory.createProductB();
    System.out.println("B2");
  }
}
```