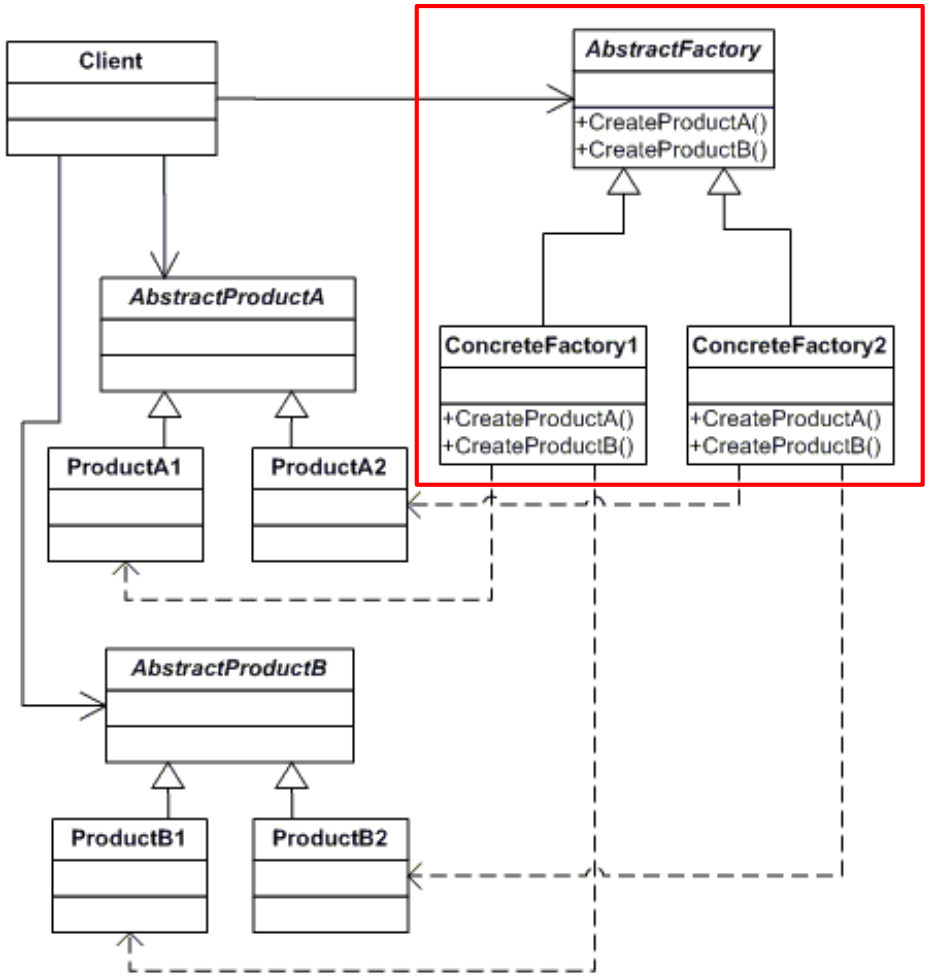
C# 추상팩토리패턴

추상팩토리 패턴

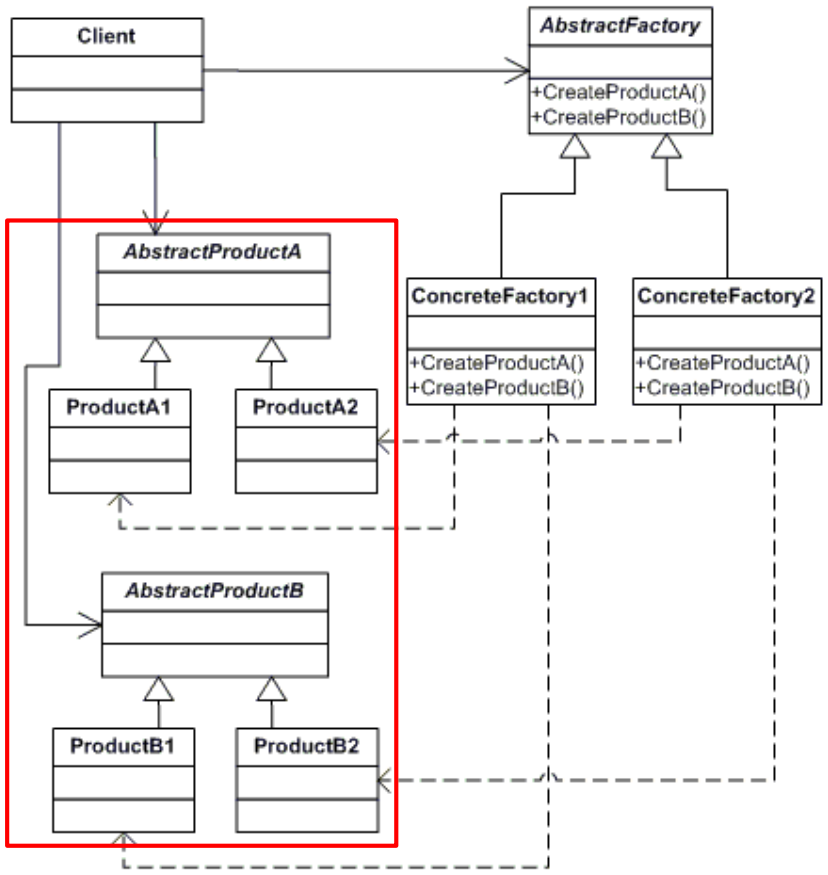
- 객체를 구성하는 부분을 추상화 하여 여러 서브클래스가 상속/구현 받아서 객체를 구성
- 객체를 구성하는 클래스들의 생성 공정을 추상화
- 장점 : 객체 생성 과정에 일관성이 있다.
- 단점 : 객체의 구현 내용이 추가 되면 추상 클래스까지 수정해야 한다.

추상 팩토리 패턴



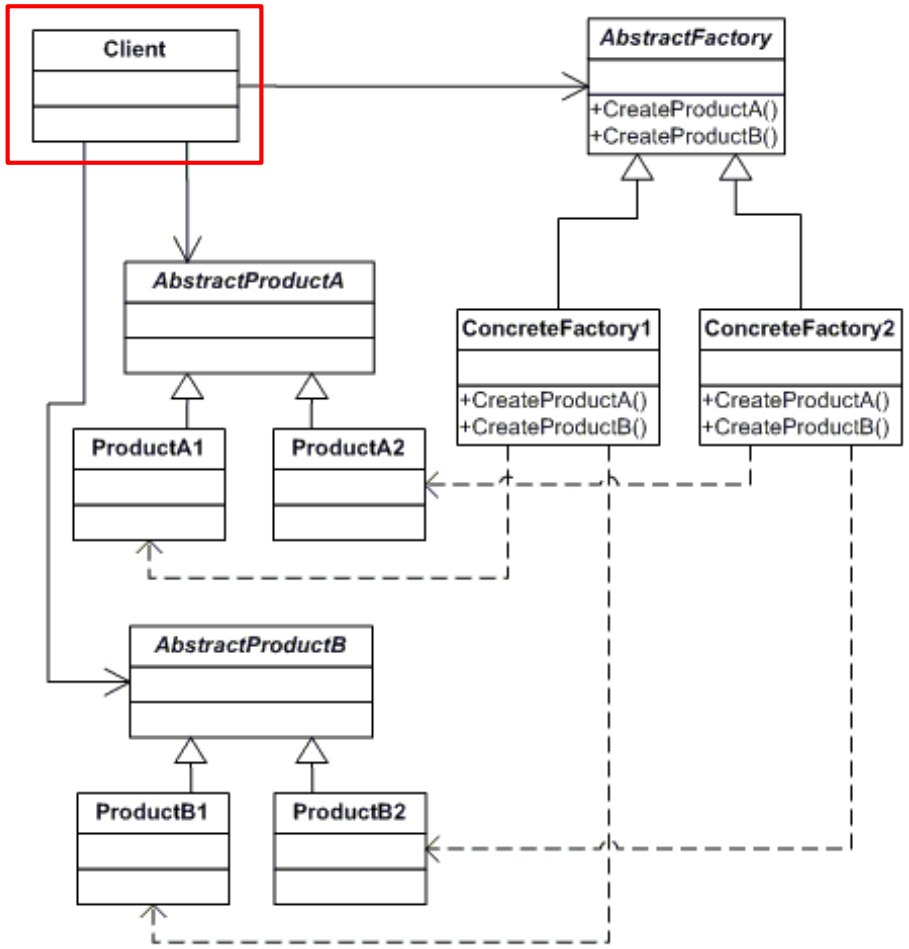

```
abstract class AbstractFactory
{
    public abstract AbstractProductA CreateProductA();
    public abstract AbstractProductB CreateProductB();
}

class ConcreteFactory1 : AbstractFactory
{
    public override AbstractProductA CreateProductA()
    {
        return new ProductA1();
    }

    public override AbstractProductB CreateProductB()
    {
        return new ProductB1();
    }
}

class ConcreteFactory2 : AbstractFactory
{
    public override AbstractProductA CreateProductA()
    {
        return new ProductA2();
    }

    public override AbstractProductB CreateProductB()
    {
        return new ProductB2();
    }
}
```

추상 팩토리 패턴




```
abstract class AbstractProductA
{
}

abstract class AbstractProductB
{
    public abstract void Interact(AbstractProductA a);
}

class ProductA1 : AbstractProductA
{
}

class ProductB1 : AbstractProductB
{
    public override void Interact(AbstractProductA a)
    {
        Console.WriteLine(this.GetType().Name + " interacts with " + a.GetType().Name);
    }
}

class ProductA2 : AbstractProductA
{
}

class ProductB2 : AbstractProductB
{
    public override void Interact(AbstractProductA a)
    {
        Console.WriteLine(this.GetType().Name + " interacts with " + a.GetType().Name);
    }
}
```

추상 팩토리 패턴




```
class Client
{
    private AbstractProductA _abstractProductA;
    private AbstractProductB _abstractProductB;

    public Client(AbstractFactory factory)
    {
        _abstractProductB = factory.CreateProductB();
        _abstractProductA = factory.CreateProductA();
    }

    public void Run()
    {
        _abstractProductB.Interact(_abstractProductA);
    }
}

class Program
{
    static void Main(string[] args)
    {
        AbstractFactory factory1 = new ConcreteFactory1();
        Client client1 = new Client(factory1);
        client1.Run();

        AbstractFactory factory2 = new ConcreteFactory2();
        Client client2 = new Client(factory2);
        client2.Run();

        Console.ReadKey();
    }
}
```