KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

Factory Method

- 하위 클래스에서 인스턴스를 만든다

[KB] IT's Your Life

KB 국민은행

Factory Method 패턴

- Template Method 패턴을 인스턴스 생성 장면에 적용한 것
  - 인스턴스를 생성하는 공장을 Template Method 패턴으로 구성한 것

- 인스턴스 생성 방법을 상위 클래스에서 결정하되, 구체적인 클래스 이름까지는 결정하지 않음
- 구체적인 살은 모두 하위 클래스에서 정의

→ 인스턴스 생성을 위한 뼈대(프레임워크)와 실제 인스턴스를 생성하는 클래스로 나누어 생각

Factory Method 패턴

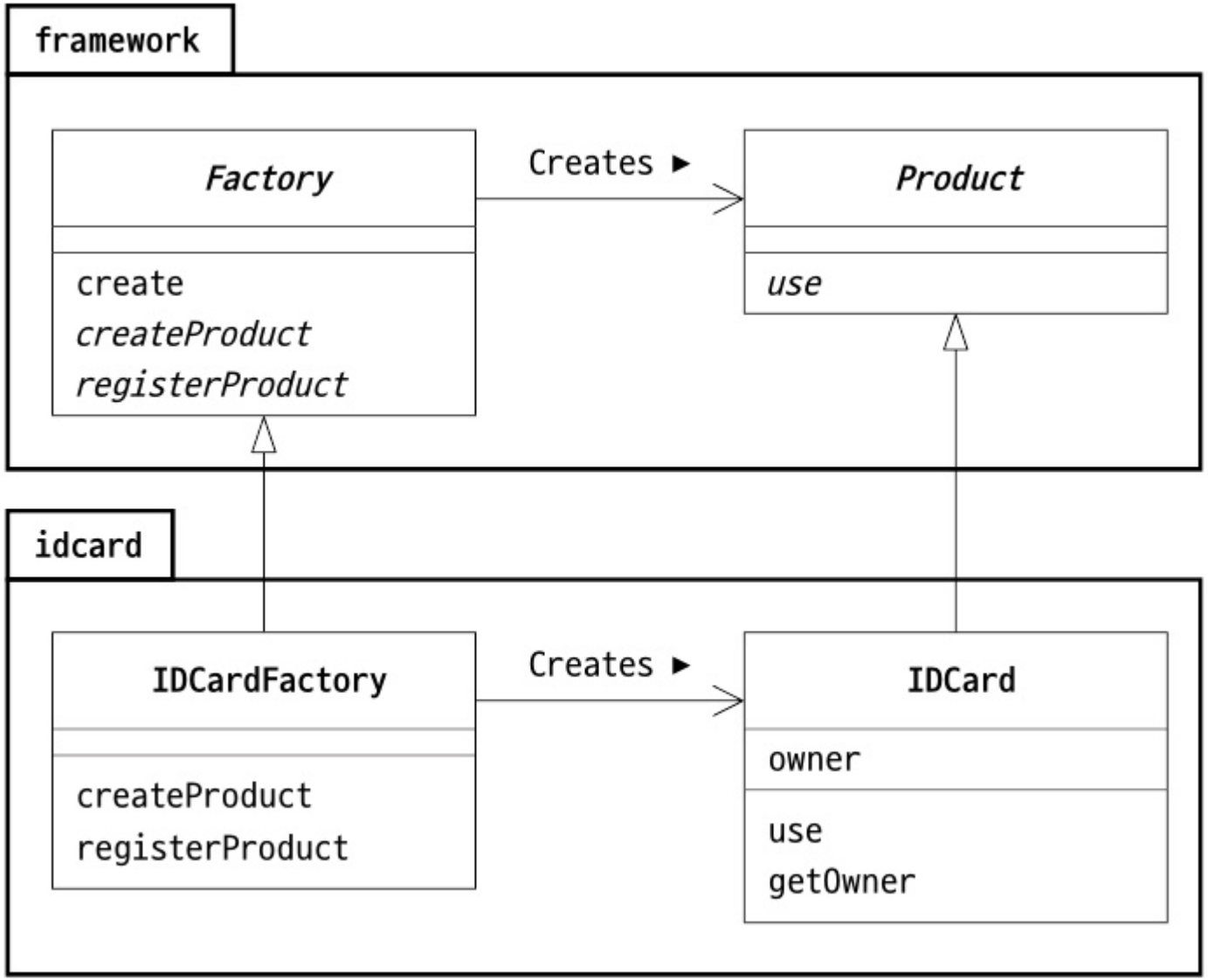
예제 프로그램

- 신분증 카드(ID 카드)를 만드는 공장
- 프레임워크 역할
  - Product 클래스와 Factory 클래스
- 구체적인 내용 역할
  - IDCard 클래스와 IDCardFactory 클래스

패키지	이름	설명
framework	Product	추상 메소드 use만 정의한 추상 클래스
framework	Factory	메소드 create를 구현한 추상 클래스
idcard	IDCard	메소드 use를 구현한 클래스
idcard	IDCardFactory	메소드 createProduct, registerProduct를 구현한 클래스
이름 없음	Main	동작 테스트용 클래스

Factory Method 패턴

- 예제 프로그램의 클래스 다이어그램



framework/Product.java

```
public abstract class Product {
    public abstract void use();
}
```

framework/Factory.java

```
public abstract class Factory {
    public final Product create(String owner) {
        Product p = createProduct(owner);
        registerProduct(p);
        return p;
    }

    protected abstract Product createProduct(String owner);
    protected abstract void registerProduct(Product product);
}
```

idcard/IDCard.java

```
public class IDCard extends Product {
    private String owner;

    public IDCard(String owner) {
        System.out.println(owner + "의 카드를 만듭니다.");
        this.owner = owner;
    }

    @Override
    public void use() {
        System.out.println(this + "을 사용합니다.");
    }

    @Override
    public String toString() {
        return "[IDCard:" + owner + "]";
    }
}
```

idcard/IDCardFactory.java

```
public class IDCardFactory extends Factory {
    @Override
    protected Product createProduct(String owner) {
        return new IDCard(owner);
    }

    @Override
    protected void registerProduct(Product product) {
        System.out.println(product + "을 등록했습니다.");
    }
}
```

Factory Method 패턴

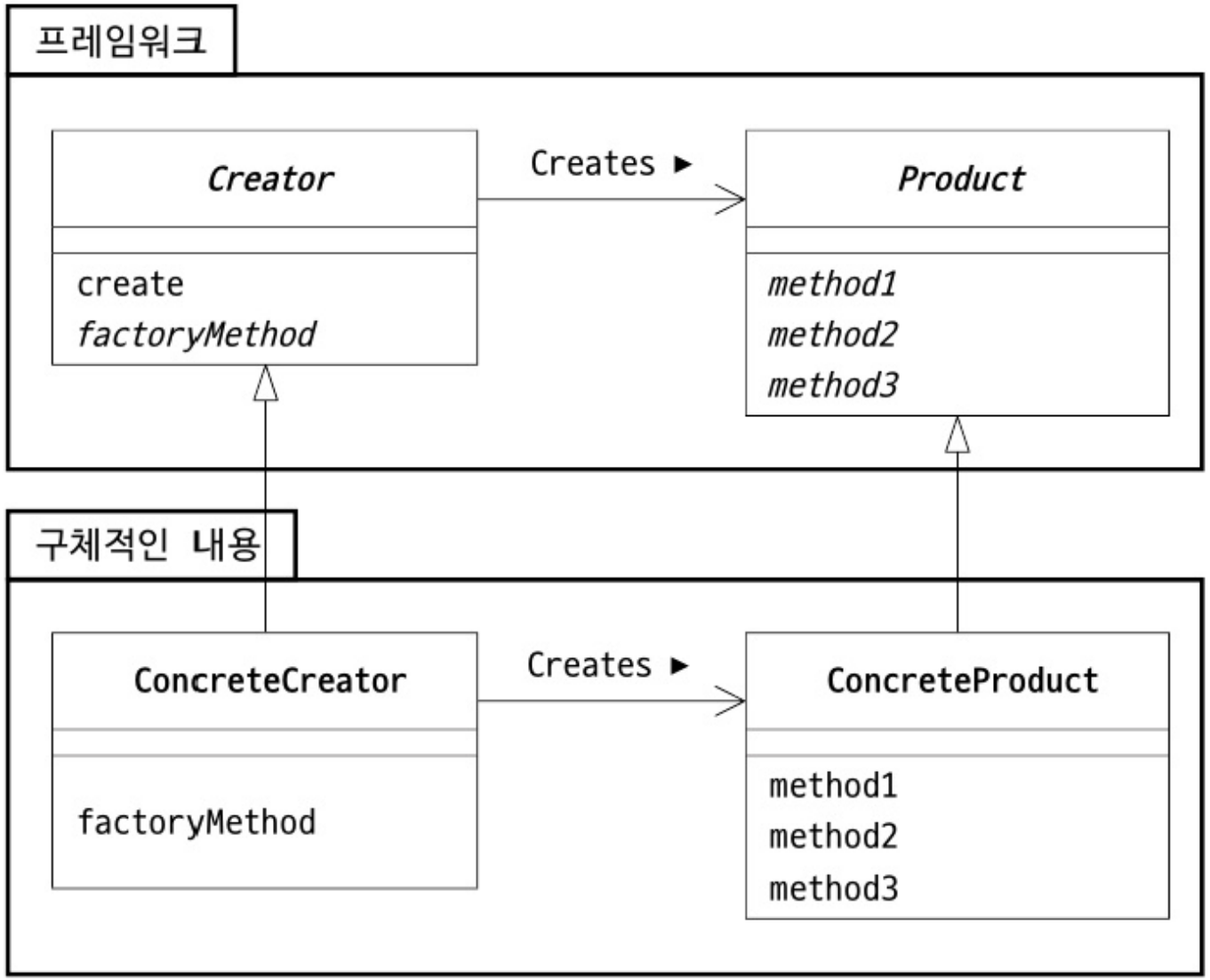
Main.java

```
public class Main {
    public static void main(String[] args) {
        Factory factory = new IDCardFactory();
        Product card1 = factory.create("Youngjin Kim");
        Product card2 = factory.create("Heunmin Son");
        Product card3 = factory.create("Kane");
        System.out.println();

        card1.use();
        card2.use();
        card3.use();
    }
}
```

```
Youngjin Kim의 카드를 만듭니다.
[IDCard:Youngjin Kim]을 등록했습니다.
Heunmin Son의 카드를 만듭니다.
[IDCard:Heunmin Son]을 등록했습니다.
Kane의 카드를 만듭니다.
[IDCard:Kane]을 등록했습니다.

[IDCard:Youngjin Kim]을 사용합니다.
[IDCard:Heunmin Son]을 사용합니다.
[IDCard:Kane]을 사용합니다.
```

Factory Method 패턴

Factory Method 패턴의 클래스 다이어그램



Factory Method 패턴

Factory Method 패턴의 사용

- 같은 프레임워크를 사용하여 전혀 다른 제품과 공장을 만드는 경우

→ framework 패키지의 내용은 수정하지 않아도 전혀 다른 제품과 공장을 만들 수 있음

사용측은 framework의 내용으로 처리하므로 사용 측의 코드 변경 없음

→ OCP