Java로 배우는 디자인패턴 입문
Chapter 4. Factory Method
인스턴스 생성을 하위 클래스에게 위임

교재: 자바언어로배우는디자인패턴입문(개정판)/YukiHiroshi저/김윤정역/영진닷컴

덕성여자대학교 컴퓨터학과
최 승 훈

01. Factory Method 패턴

- Factory Method 패턴
  - Template Method를 변형한 패턴
    - 인스턴스 만드는 방법은 상위 클래스에서 결정하고
    - 인스턴스를 실제로 생성하는 일은 하위 클래스에서 결정한다.
  - '구체적인 제품 생성'을 '공장'을 통해서 한다.

02. 예제 프로그램

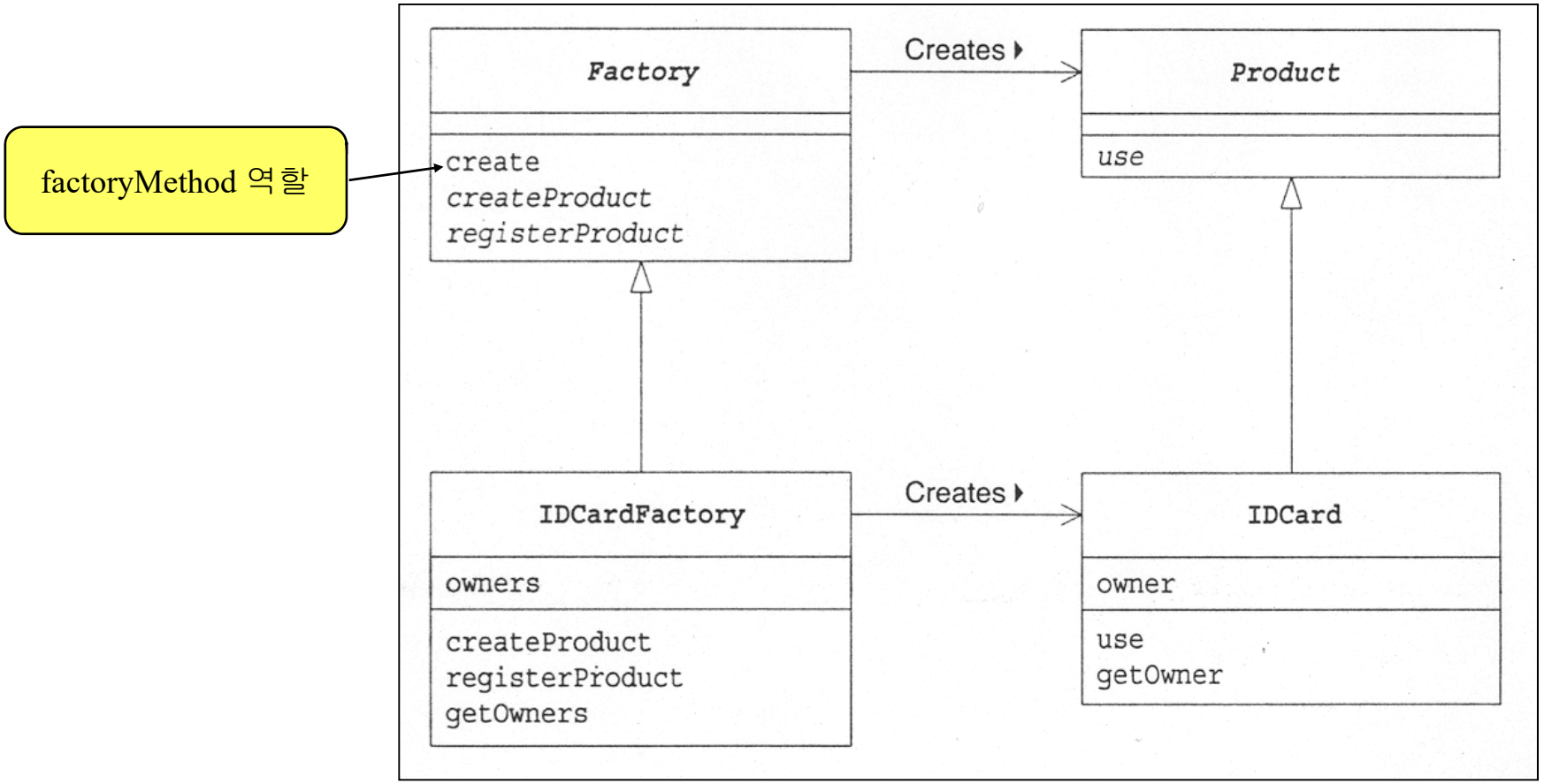
- 신분증(ID 카드)을 만드는 공장
  - framework 패키지 (응용 프로그램의 기본 틀을 제공함)
    - Product / Factory
  - idcard 패키지
    - IDCard(제품) / IDCardFactory(공장)
  - IDCard 객체를 Main 클래스에서 직접 생산할 수도 있다.
  - 그러나, Factory Method 패턴을 이용해서,
    - IDCard 객체가 필요하면, IDCardFactory를 통해서 IDCard 제품을 생산하겠다.

02. 예제 프로그램

패키지	이름	해설
framework	Product	추상 메소드 use만 정의되어 있는 추상 클래스
framework	Factory	메소드 create를 구현하고 있는 추상클래스
idcard	IDCard	메소드 use를 구현하고 있는 클래스
idcard	IDCardFactory	메소드 createProduct, registerProduct를 구현하고 있는 클래스
Anonymous	Main	동작 테스트용 클래스

02. 예제 프로그램

- 클래스 다이어그램



02. 예제 프로그램

- Product 클래스
  - '제품'을 표현한 추상 클래스
  - use( )의 구현은 하위 클래스에 맡겨짐

- Factory 클래스
  - create( )
    - Template Method 패턴 사용됨
      - 추상 메소드인 createProduct(제품생산)와 registerProduct(제품등록)를 사용함
    - 제품을 만들고, 등록한 후, 생성된 제품을 반환한다.
  - createProduct() / registerProduct()
    - 하위 클래스에서 구현한다.
    - factory method 역할을 담당한다.

02. 예제 프로그램

- IDCard 클래스
  - Product 클래스의 하위 클래스
  - 상위 클래스의 use( ) 메소드를 구현함
  - getOwner( )를 추가함

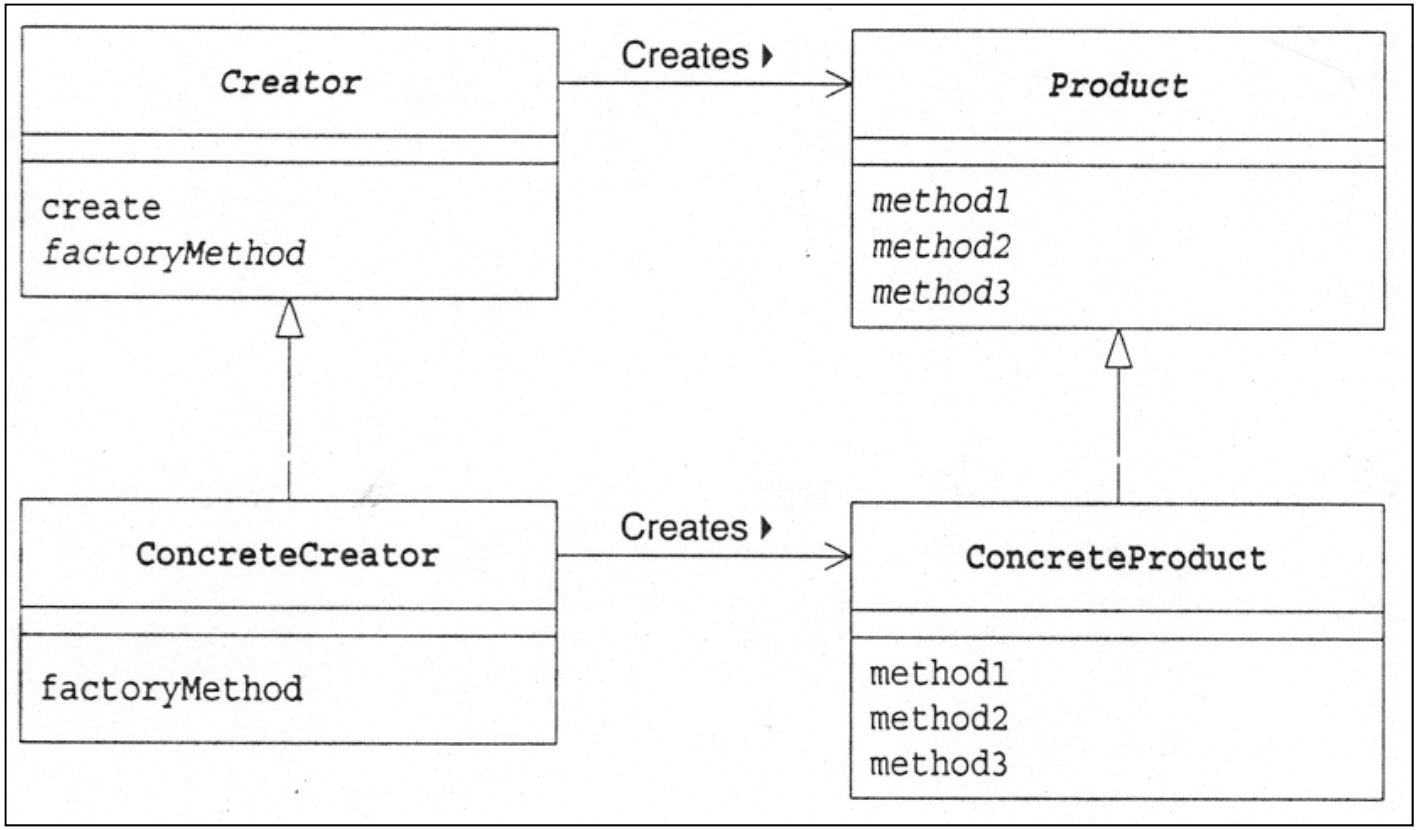
- IDCardFactory 클래스
  - createProduct와 registerProduct를 구현
  - createProduct( )
    - IDCard 제품을 실제로 생성함
    - 어떤 제품을 생산할 지 결정한다
  - registerProduct( )
    - IDCard의 소유주를 owners 필드에 추가함(등록함)

02. 예제 프로그램

- Main 클래스
  - framework 패키지와 idcard 패키지를 이용해서 IDCard를 생성해서 사용함
  - 필요한 IDCard 공장을 만들고, IDCard 공장의 create( ) 메소드를 호출해서 원하는 IDCard 제품을 얻는다.

04. Factory Method 패턴에 등장하는 역할

- 일반적인 클래스 다이어그램



04. Factory Method 패턴에 등장하는 역할

- Product(제품)의 역할
  - 생성된 제품(인스턴스)이 가지고 있어야 할 인터페이스(API)를 결정하는 추상 클래스
  - 구체적인 역할은 하위 클래스인 ConcreteProduct 역할이 결정한다.
  - 예제에서는, Product 클래스가 해당됨

- Creator(생산자)의 역할
  - Product 클래스를 생성하는 추상 클래스
  - Creator는 실제 제품을 생성하는 일을 하는 ConcreteCreator 역할에 대해서는 아무것도 모른다.
  - 예제에서는, Factory 클래스가 해당됨

04. Factory Method 패턴에 등장하는 역할

- ConcreteProduct(구체적 제품)의 역할
  - 구체적인 제품을 나타내는 클래스
  - 예제에서는, IDCard 클래스가 해당됨

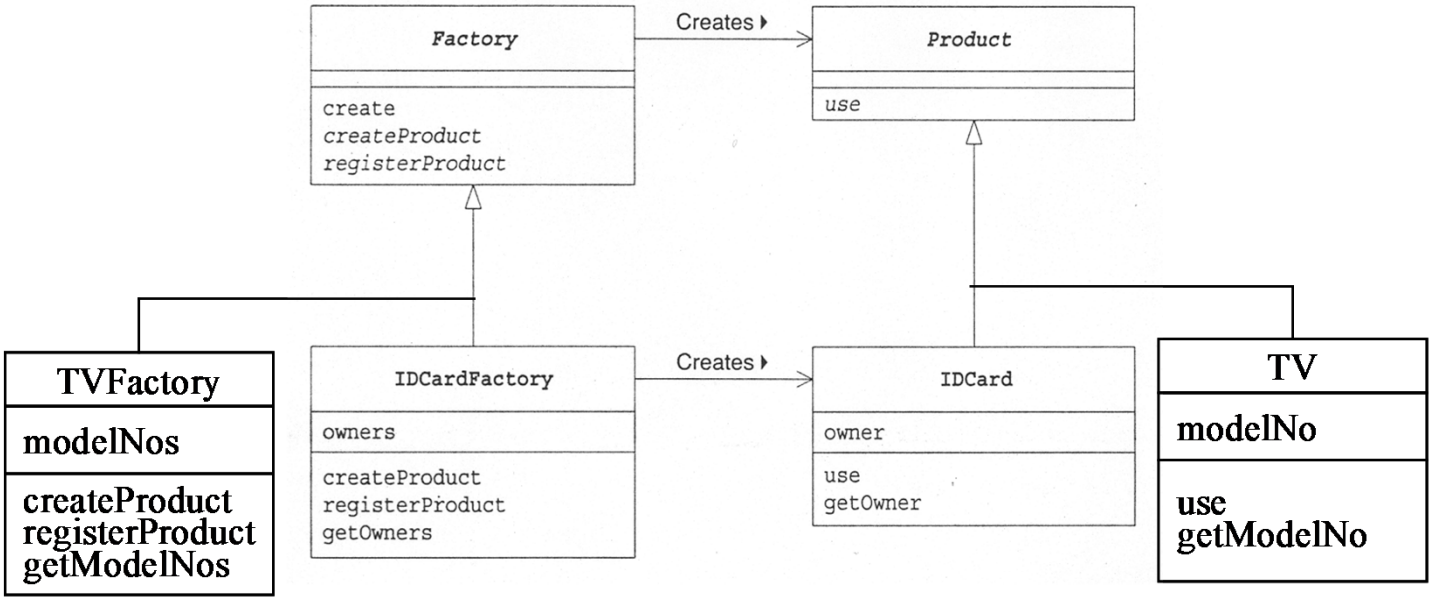
- ConcreteCreator(구체적 생산자)의 역할
  - 구체적인 제품을 만드는 클래스
  - 예제에서는, IDCardFactory 클래스가 해당됨

05. 독자의 사고를 넓혀주는 힌트

- 프레임워크와 구체적인 공장 및 제품을 분리
  - 같은 프레임워크를 이용해서(즉, 프레임워크 수정 없이), 다른 '공장'과 다른 '제품'을 추가로 정의할 수 있다.
    - 예: TV공장 + TV

05. 독자의 사고를 넓혀주는 힌트

- 예: TV공장 + TV



Factory와 Product 수정 없이, 다른 종류의 '제품'과 '공장'을 추가로 만들수 있다.

=> Main 클래스(클라이언트)에서는, TV 제품이 필요한 경우, TVFactory 객체를 생성한 후에 TVFactory 객체의 create( ) 메소드를 호출하기만 하면 된다.

05. 독자의 사고를 넓혀주는 힌트

- 각 공장이 어떤 제품을 어떻게 생산하는지에 대해서, 클라이언트는 신경 쓰지 않는다(모른다)
  - 예: IDCardFactory 공장이, IDCard 제품 대신에, IDCard2 라는 제품을 생산하도록 바뀌어도, Main 클래스의 코드는 바뀌지 않는다.
    - 단지, IDCardFactory 객체를 생성하고, create( ) 메소드를 호출하기만 하면 된다.
  - Factory Method 패턴을 사용하지 않는다면,
    - new IDCard( ) 부분을 모두 찾아서
    - new IDCard2( )로 바꿔주어야 한다.

05. 독자의 사고를 넓혀주는 힌트

- 인스턴스 생성 – 메소드 구현 방법
  - Factory 클래스의 createProduct 메소드 구현 방법에는 3가지가 있다
  - (1) 추상 메소드로 한다. (예제 프로그램의 경우)
  - (2) 디폴트 구현을 준비해 둔다
    - 하위 클래스에서 구현을 준비하지 않았을 경우에 이 디폴트 구현이 실행된다.

```
class Factory {
    public Product createProduct(String name) {
        return new Product(name);
    }
}
```

05. 독자의 사고를 넓혀주는 힌트

- 인스턴스 생성 – 메소드 구현 방법

  - (3) 에러로 처리한다.
    - 디폴트 구현의 내용을 예외를 발생시키는 문장으로 한다.
    - 하위 클래스에서 구현하지 않으면, 이 디폴트 구현이 실행되어서 예외가 발생한다.

```
class Factory {
    public Product createProduct(String name)
            throws    FactoryMethodRuntimeException {
        throw new FactoryMethodRuntimeException();
    }
}
```

```
public class FactoryMethodRuntimeException
            extends RuntimeException {

}
```

06. Factory Method 패턴과 관련된 패턴

- Template Method 패턴 (3장)
- Singleton 패턴 (5장)
- Composite 패턴 (11장)
- Iterator 패턴 (1장)
  - iterator( ) 메소드가, Iterator 인스턴스를 생성할 때, Factory Method 패턴을 사용하는 경우가 있다.

07. 요약

- 제품 생성에 대한 프레임워크와 구체적인 공장 및 제품을 분리해서 구현한다.
  - 제품이 필요한 경우, 클라이언트는 구체적인 공장에 대한 객체를 생성한 후 create( ) 메소드를 호출하기 하면 된다.

연습문제 (각자 공부할 것)

- 4-1
  - IDCard 클래스의 생성자에 public이 붙어 있지 않은 이유는?

- 4-2. 프로그램 작성
  - IDCard 제품 생성시 serial 번호를 붙인다.
  - 생산된 IDCard 제품의 owner와 serial 쌍을 저장한다.
    - Hashtable 클래스 이용
  - IDCard 생산 방법이 바뀌어도, framework와 클라이언트(Main 클래스) 쪽 코드는 변경이 없음에 주목해야 한다.

- 4-3
  - 추상 클래스 정의 시, 생성자를 선언하면 어떻게 되는가?
    - 생성자는 상속되지 않는다. 따라서, abstract한 생성자는 무의미함

연습문제 (각자 공부할 것)

- 4-3 (계속): 생성자 호출
  - 한 객체가 생성될 때는 기본적으로 부모의 인자 없는 생성자가 가장 먼저 호출된다.
  - 한 객체가 생성될 때 부모의 인자 있는 생성자를 호출하려면
    - 첫 문장을 인자 있는 부모 생성자를 호출하는 문장으로 한다.
      - 예: super("csh", 12345)
    - 그러면, 부모의 인자 없는 생성자 대신 인자 있는 부모 생성자가 호출된다.