PRESENTATION

5주차 주제 디자인 패턴

팩토리 패턴 ~팩토리 메소드 패턴~

by mun

디자인패턴

설계 문제에 대한 해답을 문서화하기 위해 고안된 형식 방법이다.

프로그램 개발에서 자주 나타나는 과제를 해결하기 위한 방법 중 하나로,
과거의 소프트웨어 개발 과정에서 발견된 설계의 노하우를 축적하여 이름을 붙여,
이후에 재이용하기 좋은 형태로 특정의 규약을 묶어서 정리한 것이다. (위키백과)

프로그램을 설계할 때 발생했던 문제점들을 객체 간의 상호 관계 등을 이용하여 해결할 수 있도록 하나의 '규약' 형태로 만들어 놓은 것을 의미한다.

싱글톤 패턴
팩토리 패턴
전략 패턴
옵저버 패턴
프록시 패턴
MVC 패턴
MVP 패턴
MVVM 패턴 등등등 ...

다양한 패턴들 ...

팩토리 패턴

객체를 사용하는 코드에서 객체 생성 부분을 떼어내 추상화한 패턴
상속 관계에 있는 두 클래스에서 상위 틀래스가 중요한 뼈대를 결정하고,
하위 클래스에서 객체 생성에 관한 구체적인 내용을 결정하는 패턴

심플 팩토리 패턴

객체를 생성하는 클래스를 따로 두는 것을 의미

상위 클래스와 하위 클래스가 분리되기 때문에
느슨한 결합을 가진다.

상위 클래스에서는 인스턴스 생성 방식에 대해 전혀
알 필요가 없기 때문에 더 많은 유연성을 갖게 된다.

객체 생성 로직이 따로 떼어져 있기 때문에 코드를
리팩터링 하더라도 한 곳만 고칠 수 있게 되어
유지보수성이 증가된다.

커피 만들기

커피

```
class Coffee {
        String name;
        int price;
}
```

아메리카노

```
class Americano extends Coffee {
        Americano() {
                name = "아메리카노";
                price = 900;
        }
}
```

라떼

```
class Latte extends Coffee {
        Latte() {
                name = "라떼";
                price = 1200;
        }
}
```

커피 만들기

객체를 생성하는
팩토리 클래스 따로 존재

커피공장

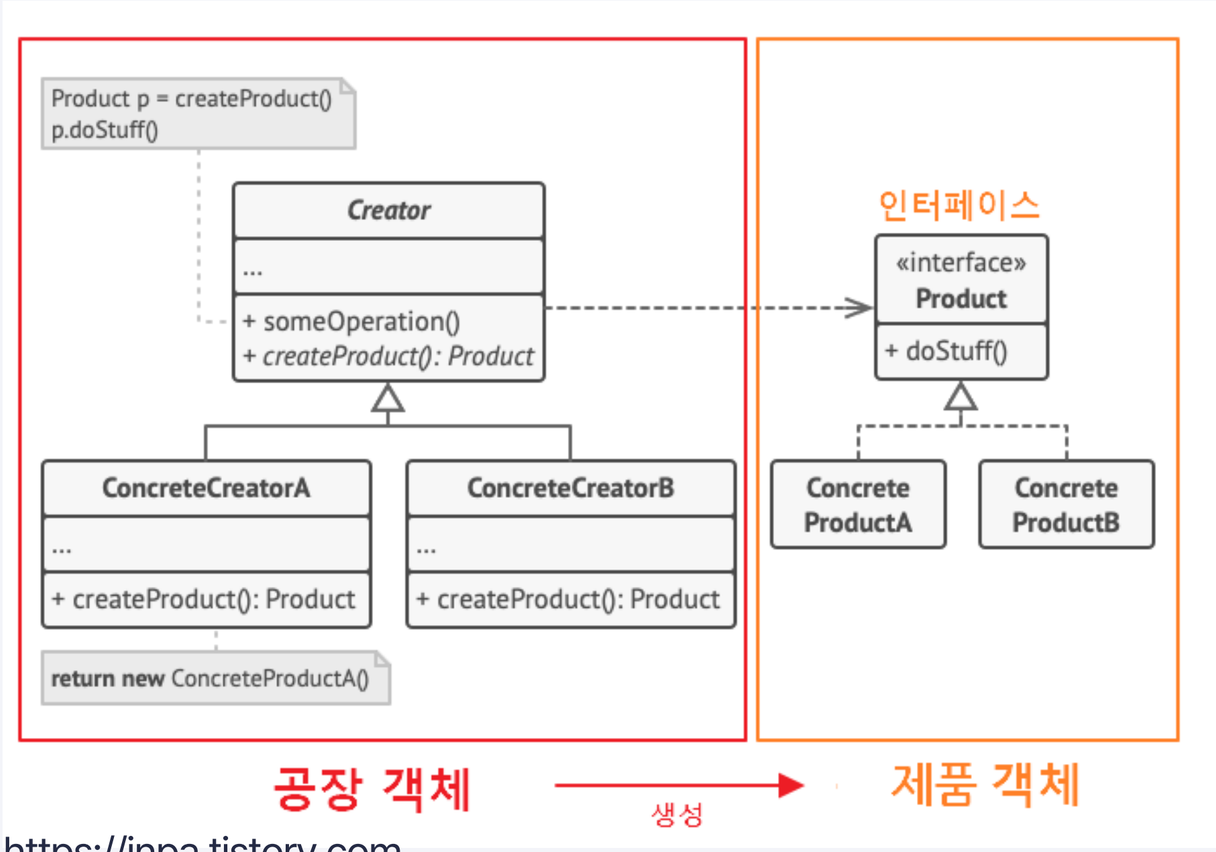
```
public class CoffeFactory {
        public Coffee orderCoffee(String name) {
                Coffee coffee = createCoffee(name);
                return coffee;
        }
        private Coffee createCoffee(String name){
                 return switch (name) {
                        case "Americano" -> new Americano();
                        case "Latte" -> new Latte();
                        default -> null;
                };
        }
}
```

클라이언트

```
class Client {
        public static void main(String[] args) {
                CoffeeFactory factory = new CoffeeFactory();
                Coffee coffee = factory.orderCoffee("Latte");
        }
}
```

팩토리 메소드 패턴

객체 생성을 팩토리 클래스로 캡슐화 처리해 대신 생성하게 하는 디자인 패턴



https://inpa.tistory.com

팩토리에서 직접 객체를 생성 X

팩토리를 상속한 서브클래스에서 객체를 생성하게끔

```
public class CoffeFactory {
    public Coffee orderCoffee(String name) {
        Coffee coffee = createCoffee(name);
        return coffee;
    }
    private Coffee createCoffee(String name){
        return switch (name) {
            case "Americano" -> new Americano();
            case "Latte" -> new Latte();
            default -> null;
        };
    }
}
```

커피 만들기

커피공장

```
abstract class CoffeFactory {
        public Coffee orderCoffee() {
                Coffee coffee = createCoffee(name);
                return coffee;
        }
        abstract protected Coffee createCoffee();
}
```

공통 메소드는 유지,
객체를 만드는 작업은 분리

아메리카노공장

```
class AmericanoFactory extends CoffeFactory {
        @Override
                protected Coffee createCoffee(){
                return new Americano();
        }
}
```

라떼공장

```
class LatteFactory extends CoffeFactory {
        @Override
                protected Coffee createCoffee(){
                return new Latte();
        }
}
```

커피 만들기

심플 팩토리 패턴

```
public class CoffeFactory {
	public Coffee orderCoffee(String name) {
		Coffee coffee = createCoffee(name);
		return coffee;
	}
	private Coffee createCoffee(String name){
		return switch (name) {
			case "Americano" -> new Americano();
			case "Latte" -> new Latte();
			default -> null;
		};
	}
}
```

팩토리 메소드 패턴 적용 후

```
abstract class CoffeFactory {
	public Coffee orderCoffee() {
		Coffee coffee = createCoffee(name);
		return coffee;
	}
	abstract protected Coffee createCoffee();
}
```

상속받은 서브 클래스에서
객체를 생성하게끔 한다.

커피 만들기

적용 전 클라이언트

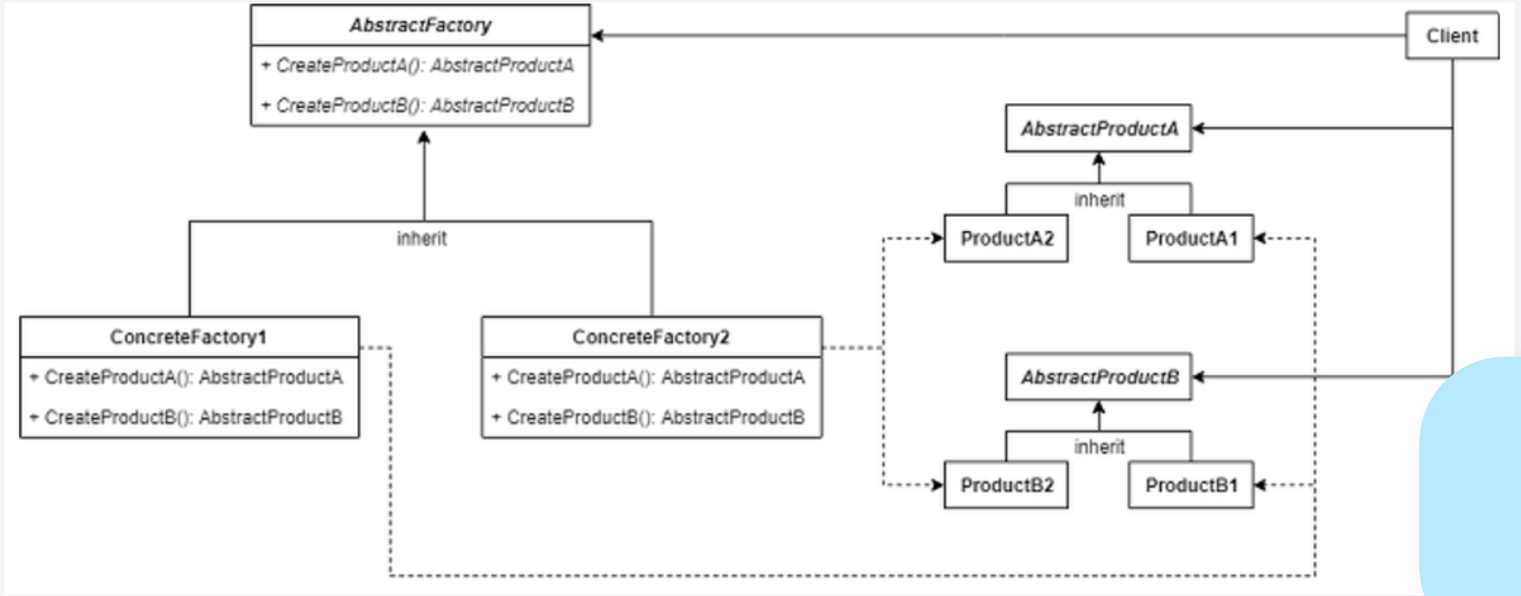
```
class Client {
        public static void main(String[] args) {
                CoffeeFactory factory = new CoffeeFactory();
                Coffee coffee = factory.orderCoffee("Latte");
        }
}
```

팩토리 메소드 패턴 적용 후

```
class Client {
        public static void main(String[] args) {
                LatteFactory ratteFactory = new LatteFactory();
                Coffee latte = ratteFactory.orderCoffee();
        }
}
```

팩토리 패턴 +

추상 팩토리 패턴



https://cjw-awdsd.tistory.com/54

팩토리 메소드 패턴

하나의 팩토리가 하나의 객체 생성

하나의 팩토리에서 연관된
여러 종류의 객체 생성을 지원

좋은 디자인 패턴을
잘 적용하자!

감사합니다

[참고 사이트] https://inpa.tistory.com/, https://jusungpark.tistory.com/14
https://cjw-awdsd.tistory.com/54, 면접을위한CS전공노트(주홍철 저)