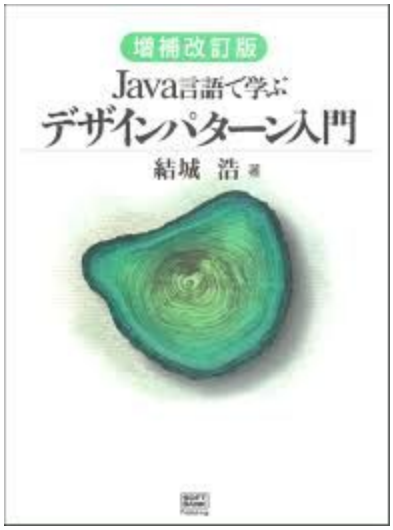
Java言語で学ぶデザインパターン入門

Factory Method
インスタンス作成をサブクラスにまかせる



FactoryMethodパターン

Template Methodパターンでは、スーパークラス側で処理の枠組みを作り、サブクラス側で具体的な処理の肉付けを行った。このパターンをインスタンス生成の場面に適用したものが、Factory Methodパターンである。Factory Methodパターンではインスタンスの作り方をスーパークラスの側で定めるが、具体的なクラス名までは定めない。具体的な肉付けはサブクラス側で行う。これによってインスタンス生成のための枠組みと、実際のインスタンス生成のクラストを分けて考えることができるようになる。

サンプルプログラム

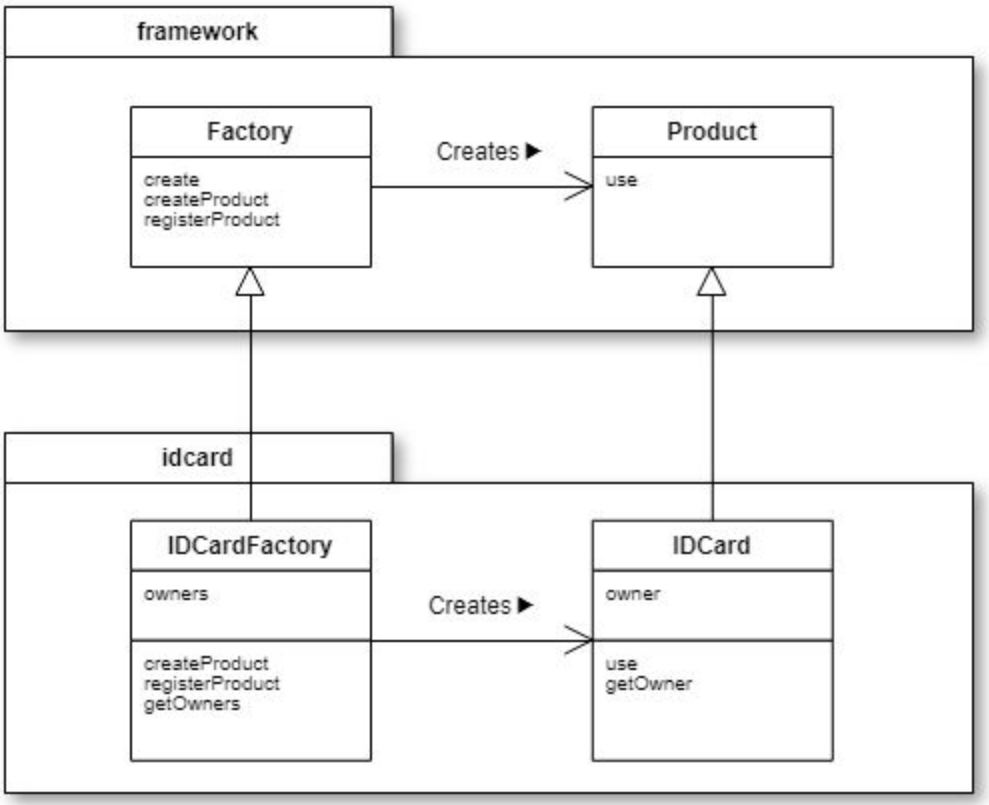
身分証明書カードを作る工場を題材とする。

- インスタンス生成のフレームワークの側（frameworkパッケージ）
- 肉付けを行っている部分（idcardパッケージ）

クラス一覧

パッケージ	名前	解説
framework	Product	抽象メソッドuseのみ定義されている抽象クラス
framework	Factory	メソッドcreateを実装している抽象クラス
idcard	IDCard	メソッドuseを実装しているクラス
idcard	IDCardFactory	メソッドcreateProduct,registerProductを実装しているクラス
無名	Main	動作テスト用のクラス

クラス図



Productクラス

```
package framework;

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

Productクラス

このフレームワークでは、製品とは「何はともあれ、useできる（使える）もの」と規定していることになる。

Factoryクラス

```
package framework;

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

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

Factoryクラス

Template Methodパターンが用いられている。このフレームワークでは、工場（Factory）とは「createメソッドでProductのインスタンスを生成するもの」と規定している。そしてcreateメソッドは「createProductで製品を作って、registerProductで製品を登録する」という手順として実装されている。

IDCardクラス

```java
package idcard;

import framework.Product;

public class IDCard extends Product{
	private String owner;
	IDCard(String owner) {
		System.out.println(owner + "のカードを作ります。");
		this.owner = owner;
	}
	public void use() {
		System.out.println(owner + "のカードを使います。");
	}
	public String getOwner() {
		return owner;
	}
}
```

IDCardクラス

IDCardのインスタンスは、idcardパッケージ外からnewを使って生成できないことを示している。Javaでは、public,protected,privateなどのアクセス制御が何もついていないコンストラクタやメソッドは、同じパッケージ内のクラスからのみ利用できる。

IDCardFactoryクラス

```java
package idcard;

import java.util.ArrayList;
import java.util.List;

import framework.Factory;
import framework.Product;

public class IDCardFactory extends Factory{
	private List owners = new ArrayList();

	protected Product createProduct(String owner) {
		return new IDCard(owner);
	}
```

```
	protected void registerProduct(Product product) {
		owners.add(((IDCard)product).getOwner());
	}
	public List getOwners() {
		return owners;
	}

}
```

IDCardFactoryクラス

Mainクラス

```
import framework.Factory;
import framework.Product;
import idcard.IDCardFactory;

public class Main {
	public static void main(String[] args) {
		Factory factory = new IDCardFactory();
		Product card1 = factory.create("石川");
		Product card2 = factory.create("田中");
		Product card3 = factory.create("中田");

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

Mainクラス