نقش الگو

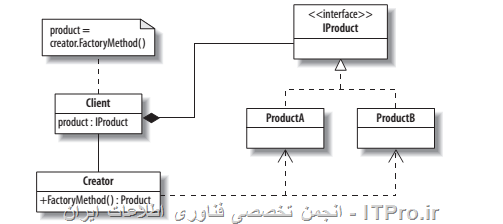
الگوی Factory Method راهی برای ساخت اشیا می باشد و برای تولید اشیا از کلاس ها مورد استفاده قرار می گیرد. اما تفاوتی که وجود دارد این است که این الگو به زیرکلاس ها (subclass) اجازه می دهد که از کلاسی که مورد نظرشان است شی را بسازد. زیرکلاس های متفاوت ممکن است که پیاده سازی های متفاوتی داشته باشند. الگوی Factory Method با توجه به اطلاعات کاربر از آن زیرکلاسی که مدنظر است شیئی را می سازد.

شرح الگو

اگر یک میوه فروشی موجود باشد که در کل روزهای سال میوه آوکادو می فروشد. این میوه فروشی باید مطمئن باشد که می تواند در طول سال می تواند این میوه را از کشاورزان تهیه نماید. این میوه فروشی باید از نواحی و کشورهای مختلف این میوه را در طول سال خریده و در داخل فروشگاه عرضه نماید. این میوه در کشورهای اسپانیا، آفریقای جنوبی و کنیان در فصل های متفاوتی رشد می کند. الگوی Factory Method شبیه به این میوه فروشی عمل می کند.

طراحی الگو

طراحی این الگو به این شکل است که کاربر یک متغیر Product معرفی می کند و از Factory Method می خواهد که آن را مقدار دهی نماید. شیئی که ساخته می شود به این بستگی دارد که چه تصمیمی برای ساخت شی product گرفته شده است. نمودار uml این الگو به شکل زیر است. دقت کنید که در این نمودار دو نوع product به نام های productA و productB وجود دارد.



عناصری که در این نمودار وجود دارند به شرح زیر هستند: 

* IProduct: اینترفیسی برای کلاس Product
* productA, productB: کلاس هایی که اینترفیس IProduct را پیاده سازی می کنند.
* Creator کلاسی که الگوی Factory Method را می سازد
* Factory Method تصمیم می گیرد که از کدام کلاس شی ساخته شود.

طراحی این الگو باعث می شود که بتوانیم در مورد این که کدام شی ساخته شود در یک زمان تصمیم گیری کنیم. نیازی نیست که کاربر بداند که زیرکلاس های مختلف را چگونه استفاده کند و فقط از اشیا ساخته شده استفاده می کند.

پیاده سازی و مثال: فراهم کردن میوه آوکادو

اگر بخواهیم برای این الگو یک مثال بیاوریم همان فراهم کردن میوه آوکادو را پیاده سازی می کنیم. اگر هرکدام از کشورهای تولید کننده آوکادو را به عنوان یک کلاس در نظر بگیریم و هرکدام اینترفیس IProduct را پیاده سازی می کند. بنابراین این کشورها می توانند آوکادو برای میوه فروشی فراهم کنند. کشورهای تولید کننده آوکادو با توجه به این که در کدام ماه از سال قرار داشته باشیم انتخاب می شوند. کد پیاده سازی و نتیجه آن به این شکل است. 

using System;

using System.Collections;

class FactoryPattern {

interface IProduct {

string ShipFrom( );

}

class ProductA : IProduct {

public String ShipFrom ( ) {

return " from South Africa";

}

}

class ProductB : IProduct {

public String ShipFrom ( ) {

return "from Spain";

}

}

class DefaultProduct : IProduct {

public String ShipFrom ( ) {

return "not available";

}

}

class Creator {

public IProduct FactoryMethod(int month) {

if (month >= 4 && month <=11)

return new ProductA( );

else

if (month == 1 || month == 2 || month == 12)

return new ProductB( );

else return new DefaultProduct( );

}

}

static void Main( ) {

Creator c = new Creator( );

IProduct product;

for (int i=1; i<=12; i++) {

product = c.FactoryMethod(i);

Console.WriteLine("Avocados "+product.ShipFrom( ));

}

}

}

و نتیجه اجرای برنامه به شکل زیر خواهد شد 

Avocados from Spain

Avocados from Spain

Avocados not available

Avocados from South Africa

Avocados from South Africa

Avocados from South Africa

Avocados from South Africa

Avocados from South Africa

Avocados from South Africa

Avocados from South Africa

Avocados from South Africa

Avocados from Spain

همانگونه که در کد دیده می شود این ما در متد main فقط از creator استفاده کرده ایم که این creator به اقتضای زمان تصمیم می گیرد که از کدام کلاس استفاده کند و هر کدام از کلاس ها یک متد ShipFrom دارند که آن را با توجه به نیاز خود پیاده سازی کرده اند.

موارد استفاده

از این الگو در موارد زیر استفاده می شود. 

1. زمانی که در برنامه ما انعطاف پذیری مهم باشد.
2. اشیا بتوانند در زیرکلاس ها گسترش پیدا کنند
3. دلیل مشخصی وجود دارد که یک زیرکلاس به زیرکلاس دیگر ارجحیت دارد- این منطق قسمتی از الگوی FactoryMethod است. **Itpro باشید**

s