Padrão de Projeto Factory Method em Java

Veja neste artigo os principais conceitos, funcionamento e implementação prática, na linguagem Java, do Padrão de Projeto Factory Method.

Introdução

Os Design Patterns (Padrões de Projetos) têm sua origem no trabalho de um arquiteto chamado Christopher Alexander em meados da década de 70. Ele escreveu dois livros de bastante sucesso que exemplificava o uso e descrevia seu raciocínio para documentar os padrões para a arquitetura. Em 1995, um grupo de quatro profissionais (que ficou conhecido como Group Of Four ou Grupo dos Quatro) escreveu e lançou o livro "Design Patterns: Elements of Reusable Object-Oriented Software" [Gamma95] que continha um catálogo com 23 padrões de projetos (Design Patterns) orientados a software. A ideia de documentar problemas recorrentes que acontecia nos softwares surgiu através da ideia de Christopher Alexander que também percebeu essa necessidade na sua área.

Os Design Patterns são uma coleção de padrões de projeto de software que contém soluções para problemas conhecidos e recorrentes no desenvolvimento de software descrevendo uma solução comprovada para um problema de projeto recorrente. A

Documentação desses padrões permite o reuso e o compartilhamento dessas informações sobre a melhor maneira de se resolver um problema de projeto de software.

Neste artigo descreveremos um dos Padrões de projetos mais utilizados pelos desenvolvedores de software, sendo bastante recorrente ,que é o Factory Method que será detalhado nas seções subsequentes do artigo.

Funcionamento

De forma geral todos os padrões Factory (Simple Factory, Factory Method, Abstract Factory) encapsulam a criação de objetos. O padrão Factory Method por sua vez encapsula a criação de objetos, no entanto, a diferença é que neste padrão encapsula-se a criação de objetos deixando as subclasses decidirem quais objetos criar.

O Diagrama de classe abaixo mostra mais detalhes sobre o funcionamento do padrão Factory Method.

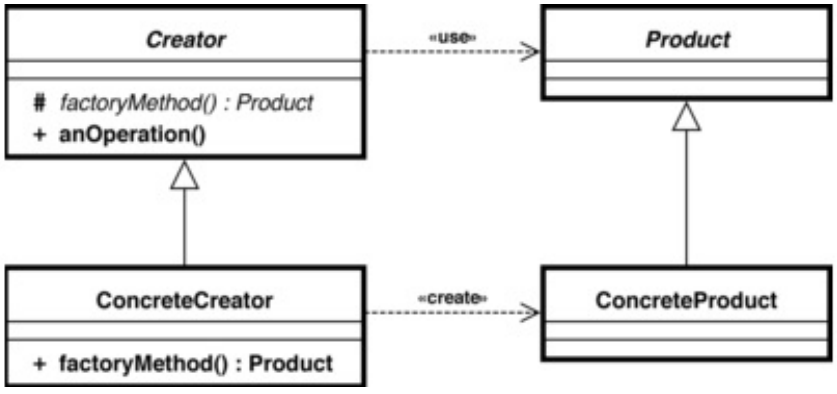

Figura 1: Diagrama de classe do Padrão Factory Method

No diagrama de classe acima temos a classe de criador abstrata que é a Creator que define um método fábrica abstrata que as subclasses implementam para criar um produto (factoryMethod) e pode possuir um ou mais métodos com seus devidos

comportamentos que chamarão o factoryMethod. Normalmente o método factoryMethod do Creator também possui um Product abstrato que é produzido por uma subclasse (ConcreteCreator). Nota-se que cada ConcreteCreator produzirá seu próprio método de criação.

Segundo o GOF (Group Of Four) o padrão Factory Method é: “Um padrão que define uma interface para criar um objeto, mas permite às classes decidirem qual classe instanciar. O Factory Method permite a uma classe deferir a instanciação para subclasses”.

Exemplo de Implementação

Segue abaixo um exemplo de implementação em Java utilizando o Padrão Factory Method. Inicialmente define-se abaixo os produtos abstratos e concretos que serão usados pela factory.

Listagem 1: Exemplo de implementação dos produtos

```
public abstract class Pessoa {

        public String nome;
        public String sexo;

}

class Homem extends Pessoa {

        public Homem(String nome) {
                this.nome = nome;
                System.out.println(“Olá Senhor ” + this.nome);
        }
}

class Mulher extends Pessoa {
```

```
        public Mulher(String nome) {
                this.nome = nome;
                System.out.println(“Olá Senhora ” + this.nome);
        }

}
```

Acima temos a implementação da primeira parte do padrão Factory Method. Nesse exemplo criou-se os Produtos (abstratos e concretos) que executam a decisão tomada na factory.

Em tempo de execução não sabemos quem será chamado, ao invés de termos if’s e else’s no cliente, temos toda a lógica de decisão na factory que é mostrada abaixo.

Listagem 2: Exemplo de implementação do Method Factory

```
class FactoryPessoa {

        public Pessoa getPessoa(String nome, String sexo) {
                if (sexo.equals(“M”))
                        return new Homem(nome);
                if (sexo.equals(“F”))
                        return new Mulher(nome);
        }

}
```

Abaixo segue um exemplo de execução desse padrão descrito acima:

Listagem 3: Exemplo de implementação do Factory Method em Java

```
public class TesteApp {

        public static void main(String args[]) {
                FactoryPessoa factory = new FactoryPessoa();
                String nome = “Carlos”;
                String sexo = “M”;
```

```
                factory.getPessoa(nome, sexo);
        }
}
```

Acima criou-se uma factory com os dados acima. Baseado na condição “sexo” temos a criação do objeto Homem que faz a saudação correta. Veja que toda a parte de decisão, ou a sujeira, fica tudo na fábrica para que ela possa decidir o que fazer.

Vantagens do Padrão Factory Method

O Factory Method é bastante utilizado em diversos projetos, até mesmo nos casos em que temos apenas um Creator (diagrama acima), pois mesmo nessas condições o padrão nos oferece um meio de desligar a implementação de um Product. Adicionando ou alterando Products isso não irá afetar o Creator, pois eles não estão fortemente ligados.

Com o padrão Factory Method podemos encapsular o código que cria objetos. É muito comum termos classes que instanciam classes concretas e essa parte do código normalmente sofre diversas modificações, portanto nesses casos usamos um Factory Method que encapsula esse comportamento de instanciação.

Usando o Factory Method temos o nosso código de criação em um objeto ou método, evitando assim a duplicação e além disso temos um local único para fazer manutenção. O padrão também nos dá um código flexível e extensível para o futuro.

Conclusão

Como foi possível estudar neste artigo, o padrão Factory Method oferece um modo de encapsular as instanciações de tipos concretos. O Creator nos oferece um método para criação de objetos, os demais métodos operam em cima das subclasses de

Creator, ou seja, os ConcreteCreator, fabricados pelo factoryMethod. Além disso apenas os ConcreteCreator implementam o método de fábrica e criam Products como pode-se observar no diagrama de classe do padrão.

Bibliografia

- Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra. Head First Design Patterns. O'Reilly Media, 2004.
- Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, 2010.