6.13 Mẫu thiết kế FACTORY METHOD

6.13.1 Mẫu thiết kế Factory Method

Trong thế giới ngày nay, mỗi người sử dụng phần mềm hỗ trợ công việc của họ. Mới đây, một công ty sản xuất buộc phải thay đổi cách họ sử dụng để nhận đơn đặt hàng của khách hàng. Công ty bây giờ xem xét việc sử dụng ứng dụng để nhận đơn đặt hàng từ khách hàng. Họ nhận đơn đặt, lỗi đơn đặt, phản hồi đơn đặt trước và trả lời đơn đặt dưới định dạng XML. Công ty yêu cầu bạn phát triển ứng dụng và hiển thị kết quả cho họ.

Thách thức chính đối với bạn là phân tích thông điệp XML và hiển thị nội dung của nó cho người sử dụng. Có các định dạng XML khác nhau phụ thuộc vào các kiểu khác nhau của thông điệp mà công ty nhận được từ khách hàng. Chẳng hạn, XML kiểu đơn có các tập xml tag khác khi so sánh với XML trả lời hoặc XML lỗi. Nhưng công việc lõi là như nhau, đó là hiển thị cho người sử dụng thông điệp được chuyển tải trong XML.

Mặc dù công việc lõi là như nhau, đối tượng này cần được sử dụng đa dạng tùy theo kiểu của XML, mà ứng dụng này nhận được từ khách hàng. Như vậy, đối tượng ứng dụng có vẻ chỉ biết rằng nó cần truy cập đến một lớp từ bên trong một phân cấp lớp (phân cấp của các bộ phân tích cú pháp khác nhau), nhưng không biết chính xác lớp nào từ trong tập các lớp con của lớp cha sẽ được chọn.

Trong trường hợp này, tốt hơn là cung cấp một nhà máy - factory để tạo ra các bộ phân tích, và trong thời gian chạy một parser sẽ được khởi tạo để thực hiện công việc đó, tùy theo kiểu của XML, mà ứng dụng nhận được từ người sử dụng.

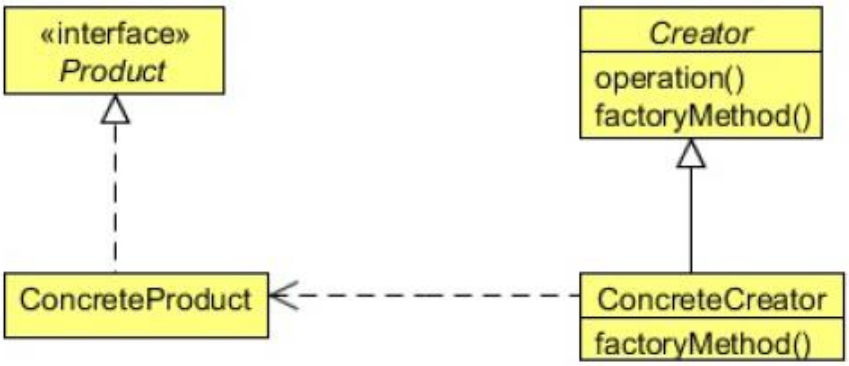
Mẫu Factory Method, phù hợp với tình huống này, định nghĩa một giao diện để tạo ra đối tượng, nhưng cho phép các lớp con quyết định lớp nào được khởi tạo. Factory Method cho phép lớp trì hoãn khởi tạo của lớp con.

Chúng ta xem xét chi tiết về mẫu Factory Method và sau đó sẽ sử dụng nó cài đặt bộ phân tích XML cho ứng dụng.

6.13.2 Mẫu Factory Method là gì

Mẫu *Factory Method* cho chúng ta cách đóng gói khởi tạo các kiểu cụ thể. Mẫu *Factory Method* đóng gói chức năng được yêu cầu để chọn và khởi tạo một lớp phù hợp, bên trong một phương thức chỉ định được tham chiếu như phương thức *factory method. Factory Method* chọn một lớp phù hợp của phân cấp lớp dựa trên ngữ cảnh của ứng dụng và các yếu tố ảnh hưởng khác. Sau đó nó khởi tạo lớp được chọn và trả về nó như khởi tạo của kiểu lớp cha.

Ưu điểm của cách tiếp cận này là các đối tượng ứng dụng có thể sử dụng phương thức *Factory method* để nhận truy cập khởi tạo của lớp phù hợp. Nó loại bỏ nhu cầu mà đối tượng ứng dụng cần xử lý với đa dạng tiêu chí lựa chọn lớp.



Product

- Định nghĩa giao diện của các đối tượng mà *factory method* tạo.

Concrete Product

- Cài đặt giao diện *Product*

Creator

- Khai báo *factory method*, mà trả về đối tượng kiểu *Product*. *Creator* cũng định nghĩa cài đặt mặc định của phương thức *factory method* mà trả về đối tượng *ConcreteProduct* mặc định.
- Có thể gọi phương thức *factory method* để tạo đối tượng *Product*.

ConcreteCreator

- Ghi đè *factory method* để trả về khởi tạo *ConcreteProduct.*

Phương thức *Factory Method* loại bỏ cần thiết gắn kết các lớp chuyên biệt ứng dụng với code của bạn. Code này chỉ làm việc với giao diện *Product*, do đó nó có thể làm việc với bất cứ các lớp *ConcreteProduct* mà người sử dụng định nghĩa nào khác.

6.13.3 Cài đặt mẫu Factory Method

Để cài đặt giải pháp cho ứng dụng đã bàn luận trên, trước hết ta kiểm tra sản phẩm ta có

```
package com.javacodegeeks.patterns.factorymethodpattern;

public interface XMLParser {

        public String parse();

}
```

Giao diện trên sẽ được sử dụng bởi các bộ phân tích XML khác nhau.

```
package com.javacodegeeks.patterns.factorymethodpattern;

public class ErrorXMLParser implements XMLParser{

        @Override
        public String parse() {

                System.out.println("Parsing error XML...");
                return "Error XML Message";
        }

}
```

ErrorXMLParser cài đặt *XMLParser* và được sử dụng để phân tích thông điệp XML lỗi.

```
package com.javacodegeeks.patterns.factorymethodpattern;

public class FeedbackXML implements XMLParser{

        @Override
        public String parse() {
                System.out.println("Parsing feedback XML...");
                return "Feedback XML Message";
        }

}
```

Lớp trên được sử dụng để phân tích thông điệp XML phản hồi.

Các bộ phân tích XML khác là:

```
package com.javacodegeeks.patterns.factorymethodpattern;

public class OrderXMLParser implements XMLParser{

	@Override
	public String parse() {
		System.out.println("Parsing order XML...");
		return "Order XML Message";
	}

}

package com.javacodegeeks.patterns.factorymethodpattern;

public class ResponseXMLParser implements XMLParser{

	@Override
	public String parse() {
		System.out.println("Parsing response XML...");
		return "Response XML Message";
	}

}
```

Để hiển thị các thông điệp được phân tích từ các bộ phân tích, một lớp dịch vụ trừu tượng được tạo mà sẽ mở rộng bởi dịch vụ chuyên biệt như phân tích chuyên biệt, hiện thị các lớp.

```
package com.javacodegeeks.patterns.factorymethodpattern;

public abstract class DisplayService {

	public void display(){
		XMLParser parser = getParser();
		String msg = parser.parse();
		System.out.println(msg);
	}

	protected abstract XMLParser getParser();

}
```

Lớp trên được sử dụng để hiển thị thông điệp được đưa ra từ bộ phân tích XML cho người sử dụng. Lớp trên là lớp trừu tượng mà chứa hai phương thức quan trọng. Phương thức *display* được dùng để hiển thị thông điệp cho người sử dụng. Phương thức *getParser* là *Factory method* mà được cài đặt bởi các lớp con để khởi tạo các đối tượng phân tích và phương thức này được sử dụng bởi phương thức *display* để phân tích XML và nhận thông điệp để hiển thị.

Dưới đây là các lớp con cúa *DisplayService* mà cài đặt phương thức *getParser*.

```
package com.javacodegeeks.patterns.factorymethodpattern;

public class ErrorXMLDisplayService extends DisplayService{

	@Override
	public XMLParser getParser() {
		return new ErrorXMLParser();
	}

}
```

```
package com.javacodegeeks.patterns.factorymethodpattern;

public class FeedbackXMLDisplayService extends DisplayService{

	@Override
	public XMLParser getParser() {
		return new FeedbackXML();
	}

}
```

```
package com.javacodegeeks.patterns.factorymethodpattern;

public class OrderXMLDisplayService extends DisplayService{

	@Override
	public XMLParser getParser() {
		return new OrderXMLParser();
	}

}
```

```
package com.javacodegeeks.patterns.factorymethodpattern;

public class ResponseXMLDisplayService extends DisplayService{

	@Override
	public XMLParser getParser() {
		return new ResponseXMLParser();
	}

}
```

Bây giờ chúng ta sẽ kiểm tra phương thức *Factory Method*.

```
package com.javacodegeeks.patterns.factorymethodpattern;

public class TestFactoryMethodPattern {

	public static void main(String[] args) {
```

```
                DisplayService service = new FeedbackXMLDisplayService();
                service.display();

                service = new ErrorXMLDisplayService();
                service.display();

                service = new OrderXMLDisplayService();
                service.display();

                service = new ResponseXMLDisplayService();
                service.display();

        }

}
```

Nó cho kết quả đầu ra như sau:

```
Parsing feedback XML...
Feedback XML Message
Parsing error XML...
Error XML Message
Parsing order XML...
Order XML Message
Parsing response XML...
Response XML Message
```

Trong lớp trên, bạn có thể thấy rõ là bằng việc cho phép các lớp con cài đặt *factory method* để tạo ra các khởi tạo khác nhau của bộ phân tích mà có thể được sử dụng trong thời gian chạy tùy theo nhu cầu.

6.13.4 Khi nào sử dụng mẫu Factory Method

Sử dụng mẫu *Factory Method* khi

- Một lớp không biết trước được lớp của các đối tượng mà cần được tạo.
- Một lớp muốn các lớp con của nó đặc tả các đối tượng mà nó tạo.
- Các lớp ủy quyền tương ứng cho một trong số các lớp con bổ trợ và bạn muốn cục bộ hóa hiểu biết này của lớp con bổ trợ mà được ủy quyền.