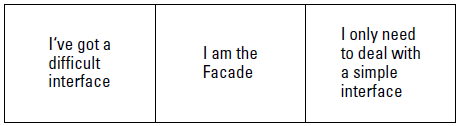
**FACADES PATTERN**

Chúng ta đã học Adapter pattern, một pattern tương tự nhưng đơn giản hơn là Façade pattern. 2 pattern này có cùng mục đích tuy nhiên Adapter pattern làm cho code của chúng ta làm việc được với code khác trong khi Facade pattern làm cho code của chúng ta dễ dàng sử dụng hơn. Facades pattern có thể được nói như sau **“Provide a unified interface to s set of interfaces in a system. Facede defines a higher-level interface that makes the subsystem easier to use”**

Ý tưởng xây dựng facede rất đơn giản, chúng ta chỉ cần đơn giản hóa các interface phức tạp thành 1 interface đơn giản.



Chúng ta thường dùng Façade pattern khi ta không thể viết lại code cho nó đơn giản. Mặc dù dùng Façade có thể giải quyết được vấn đề, tuy nhiên nếu code bên dưới có thay đổi gì thì Façade pattern cũng phải được thay đổi.

Có 1 nguyên tắc thiết kế trong OOP là tính encapsulation. Ý tưởng là chúng ta làm sao cho mỗi class hoặc object có thể hoạt động độc lập và trong bản thân mỗi object hay gói là 1 khối thiết lập vững chắc. Tuy nhiên không phải lúc nào nó cũng có thể hoạt động độc lập mà cần phải có kết nối hay thông tin về các object khác, lúc này Facede pattern có thể được sử dụng.

Một ví dụ được đưa ra như sau : chúng ta cần 1 class có thể đặt tên cho các vật thể, ý tưởng ban đầu đưa ra là tạo cho mỗi vật thể 1 thuộc tính để đặt tên.

public class DifficultProduct

{

char nameChars[] = new char[7];

public DifficultProduct()

{

}

public void setFirstNameCharacter(char c)

{

nameChars[0] = c;

}

public void setSecondNameCharacter(char c)

{

nameChars[1] = c;

}

public void setThirdNameCharacter(char c)

{

nameChars[2] = c;

}

public void setFourthNameCharacter(char c)

{

nameChars[3] = c;

}

public void setFifthNameCharacter(char c)

{

nameChars[4] = c;

}

public void setSixthNameCharacter(char c)

{

nameChars[5] = c;

}

public void setSeventhNameCharacter(char c)

{

nameChars[6] = c;

}

**public String getName()**

**{**

**return new String(nameChars);**

**}**

}

Như ta thấy ở trên có 7 thuộc tính cho 7 nameChar. Điều này là phức tạp và nếu có hàm trang nameChar thì ta không thể tạo ra hàng trăm thuộc tính. Do đó chúng ta sẽ xây dựng class mới SimpleProductFacade theo phong cách Façade để giái quyết vấn đề trên

public class SimpleProductFacade

{

public SimpleProductFacade()

{

}

.

.

.

}

Class mới này sẽ bao bọc và xử lý các vấn đề còn tồn tại trong class DifficultProduct đó là gom các thuộc tính setFirstNameCharacter, setSecondNameCharacter …. Vào chung thành 1 hàm.

public class SimpleProductFacade

{

DifficultProduct difficultProduct;

public SimpleProductFacade()

{

difficultProduct = new DifficultProduct();

}

**public void setName(String n)**

**{**

**char chars[] = n.toCharArray();**

**if(chars.length > 0){**

**difficultProduct.setFirstNameCharacter(chars[0]);**

**}**

**if(chars.length > 1){**

**difficultProduct.setSecondNameCharacter(chars[1]);**

**}**

**if(chars.length > 2){**

**difficultProduct.setThirdNameCharacter(chars[2]);**

**}**

**if(chars.length > 3){**

**difficultProduct.setFourthNameCharacter(chars[3]);**

**}**

**if(chars.length > 4){**

**difficultProduct.setFifthNameCharacter(chars[4]);**

**}**

**if(chars.length > 5){**

**difficultProduct.setSixthNameCharacter(chars[5]);**

**}**

**if(chars.length > 6){**

**difficultProduct.setSeventhNameCharacter(chars[6]);**

**}**

**public String getName()**

**{**

**return difficultProduct.getName();**

**}**

}

Bây giờ thì class chỉ còn lại 2 hàm setname và getname rất thuật tiện cho việc sử dụng. Bây giờ chúng ta làm thử 1 kiểm tra về façade pattern mà chúng ta đã xây dựng. Chúng ta đơn giản là tạo 1 class SimpleProductFacade và gọi hàm setName để thấy được tác dụng của nó.

public class TestFacade

{

public static void main(String args[])

{

TestFacade t = new TestFacade();

}

**public TestFacade()**

**{**

**SimpleProductFacade simpleProductFacade =**

**new SimpleProductFacade();**

**simpleProductFacade.setName(“printer”);**

**System.out.println(“This product is a “ +**

**simpleProductFacade.getName());**

**}**

}