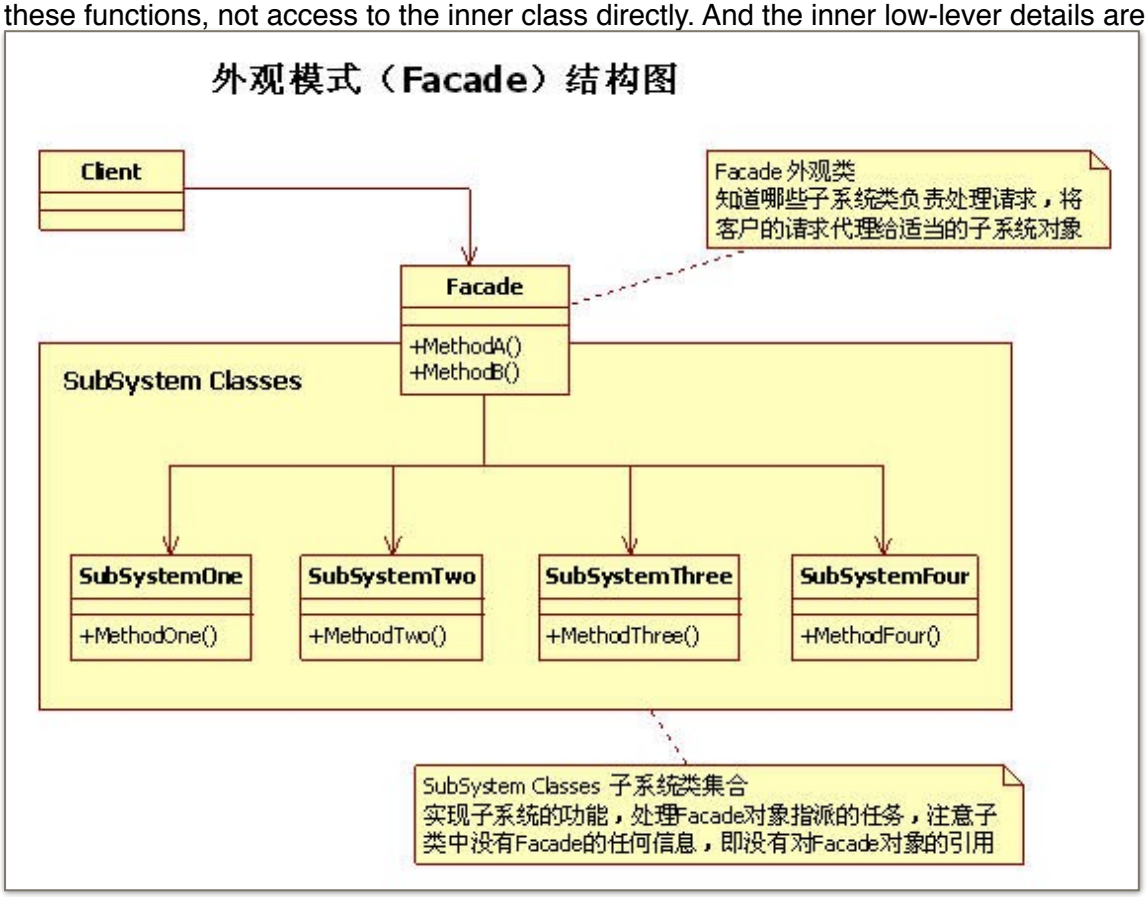
Assignment 2 : Design Patterns

3120102146 葛现隆

Explain

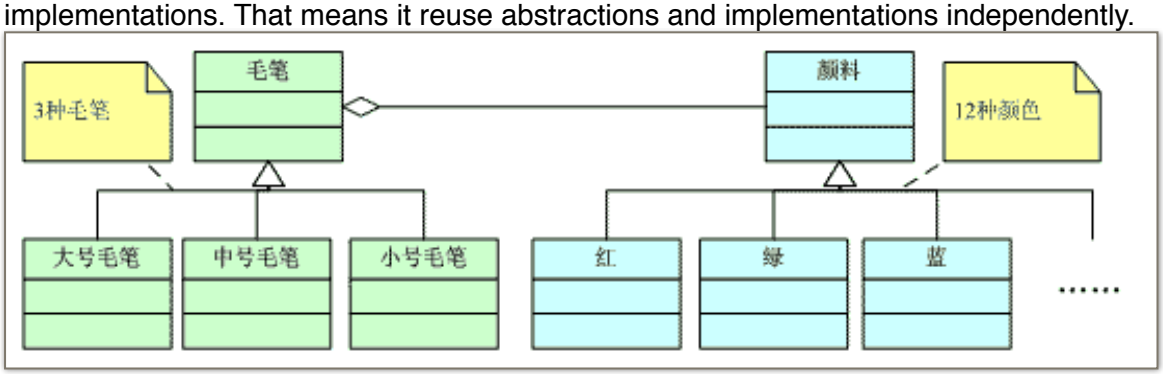
1. facade pattern

Facade pattern aims to reduce the communication an dependencies between subsystems. Inner part gives applications access to Facade, and the outer systems can only access to Facade for these functions, not access to the inner class directly. And the inner low-lever details are hided.



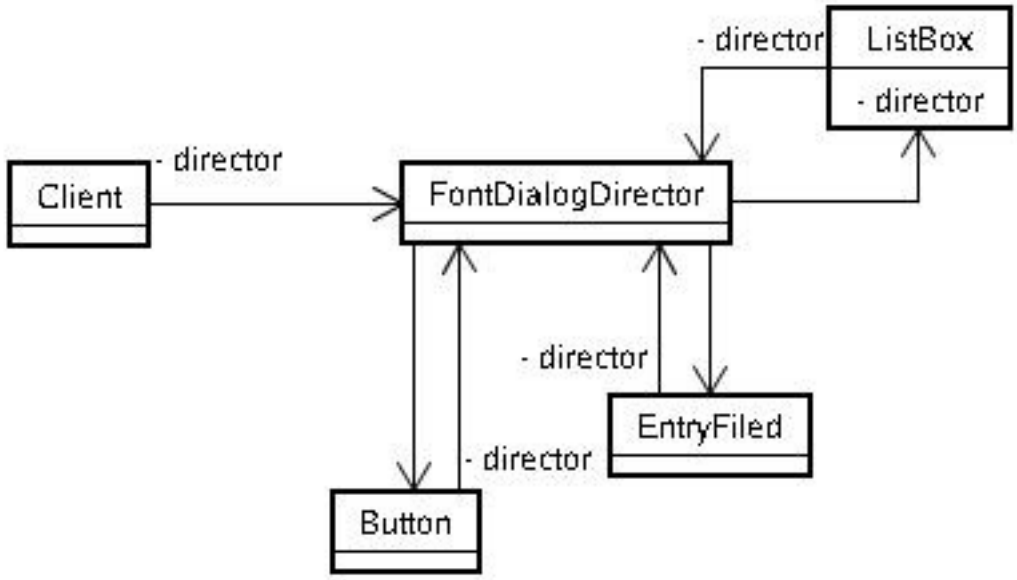
2. bridge pattern

Bridge pattern based on minimum design principles. It uses inheritance to accommodate several possible implementations of an abstraction. With an abstract class defining the interface to the abstraction, bridge pattern always build concrete subclasses implement the possible implementations. That means it reuse abstractions and implementations independently.



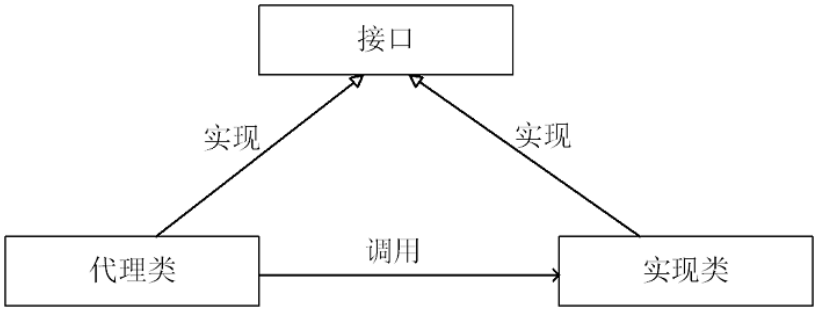
3. mediator pattern

As the name shows, we use a mediator to connect different objects interactions to avoid the many complex connections between objects. The mediator is responsible for controlling and coordinating interactions of a group of objects, keeping the group from referring to each other explicitly.



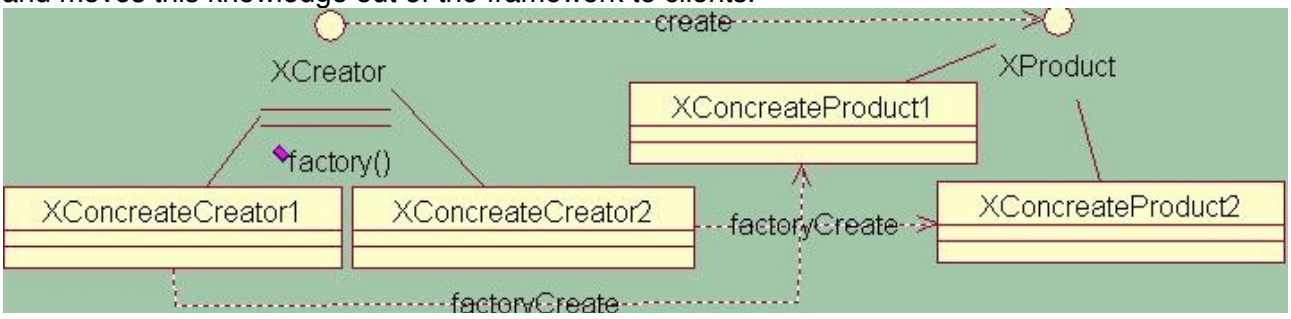
4. proxy pattern

In some cases, when we include an big/complex object, we will cost a lot of time and space for the full object.Yet, in most case, the objects are not aimed to be visible at the same time. So we use other object to temporally act as the stand-in for the complex object, which will efficiently defer the full cost.



5. factory pattern

Factory pattern define an **abstract** interface for creating an object, and let subclasses decide which class to instantiate(simple factory just use parent class to instantiate), which can defer instantiation to subclasses. Factory method encapsulates the knowledge of which class to create and moves this knowledge out of the framework to clients.



Illustrating

Here is a facade pattern. System A has class A1, A2 & A3. The client need to call A1.do(), A2.do() and A3.do() to finish the function. The code as follows:

```
//System A
class A1{
	public void do();
}

class A2{
	public void do();
}

class A3{
	public void do();
}

//Our Facade
public class Facade{
	public void do{
		A1 a1 = new A1();
		A2 a2 = new A2();
		A3 a3 = new A3();

		a1.do();
		a2.do();
		a3.do();
	}
}

//Client
public class Client{
	public static void main(String [] args){
		Facade fc = new Facade();
		fc.do();
	}
}
```