Design Patterns - Facade Pattern

Facade pattern hides the complexities of the system and provides an interface to the client using which the client can access the system. This type of design pattern comes under structural pattern as this pattern adds an interface to existing system to hide its complexities.

This pattern involves a single class which provides simplified methods required by client and delegates calls to methods of existing system classes.

门面模式隐藏系统的复杂性同时会提供一个接口给用户，使得用户可以使用该系统。这种类型的设计模式属于结构型模式的一种，它将会添加一个接口到现有的系统当中，用户通过该接口使用系统，从而隐藏了系统的复杂性。

该模式涉及一个单独的类，该类会向客户提供简单的方法并且代替用户去调用那些在系统中存在的类的方法。这样用户就不会接触到系统是如何实现的，从而隐藏了系统的复杂性。

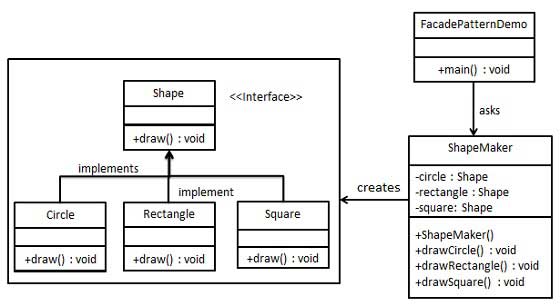
Implementation

We are going to create a *Shape* interface and concrete classes implementing the *Shape* interface. A facade class *ShapeMaker* is defined as a next step.

*ShapeMaker* class uses the concrete classes to delegate user calls to these classes. *FacadePatternDemo*, our demo class, will use *ShapeMaker* class to show the results.

我们将创建一个Shape接口，并且创建具体的类实现它。接下来我们需要声明一个门面类ShapeMaker。

ShapeMaker类代替用户去调用这些具体的类。FacadePatternDemo，我们的demo类将通过使用ShapeMaker类来展示这些结果。



Step 1

Create an interface.

创建一个接口

*Shape.java*

public interface Shape {

void draw();

}

Step 2

Create concrete classes implementing the same interface.

创建具体的类实现Shape接口

*Rectangle.java*

public class Rectangle implements Shape {

@Override

public void draw() {

System.out.println("Rectangle::draw()");

}

}

*Square.java*

public class Square implements Shape {

@Override

public void draw() {

System.out.println("Square::draw()");

}

}

*Circle.java*

public class Circle implements Shape {

@Override

public void draw() {

System.out.println("Circle::draw()");

}

}

Step 3

Create a facade class.

创建一个门面类

*ShapeMaker.java*

public class ShapeMaker {

private Shape circle;

private Shape rectangle;

private Shape square;

public ShapeMaker() {

circle = new Circle();

rectangle = new Rectangle();

square = new Square();

}

public void drawCircle(){

circle.draw();

}

public void drawRectangle(){

rectangle.draw();

}

public void drawSquare(){

square.draw();

}

}

Step 4

Use the facade to draw various types of shapes.

使用门面类画各种各样的图形

*FacadePatternDemo.java*

public class FacadePatternDemo {

public static void main(String[] args) {

ShapeMaker shapeMaker = new ShapeMaker();

shapeMaker.drawCircle();

shapeMaker.drawRectangle();

shapeMaker.drawSquare();

}

}

Step 5

Verify the output.

校验输出

Circle::draw()

Rectangle::draw()

Square::draw()