

Design Patterns

Engr. Abdul-Rahman Mahmood

DPM, MCP, QMR(ISO9001:2000)

 armahmood786@yahoo.com


 alphapeeler.sf.net/pubkeys/pkey.htm

 pk.linkedin.com/in/armahmood

 www.twitter.com/alphapeeler

 www.facebook.com/alphapeeler

 abdulmahmood-sss  alphasecure

 armahmood786@hotmail.com

 <http://alphapeeler.sf.net/me>



 alphasecure@gmail.com

 <http://alphapeeler.sourceforge.net>

 <http://alphapeeler.tumblr.com>

 armahmood786@jabber.org

 alphapeeler@aim.com

 mahmood_cubix  48660186

 alphapeeler@icloud.com

 <http://alphapeeler.sf.net/acms/>

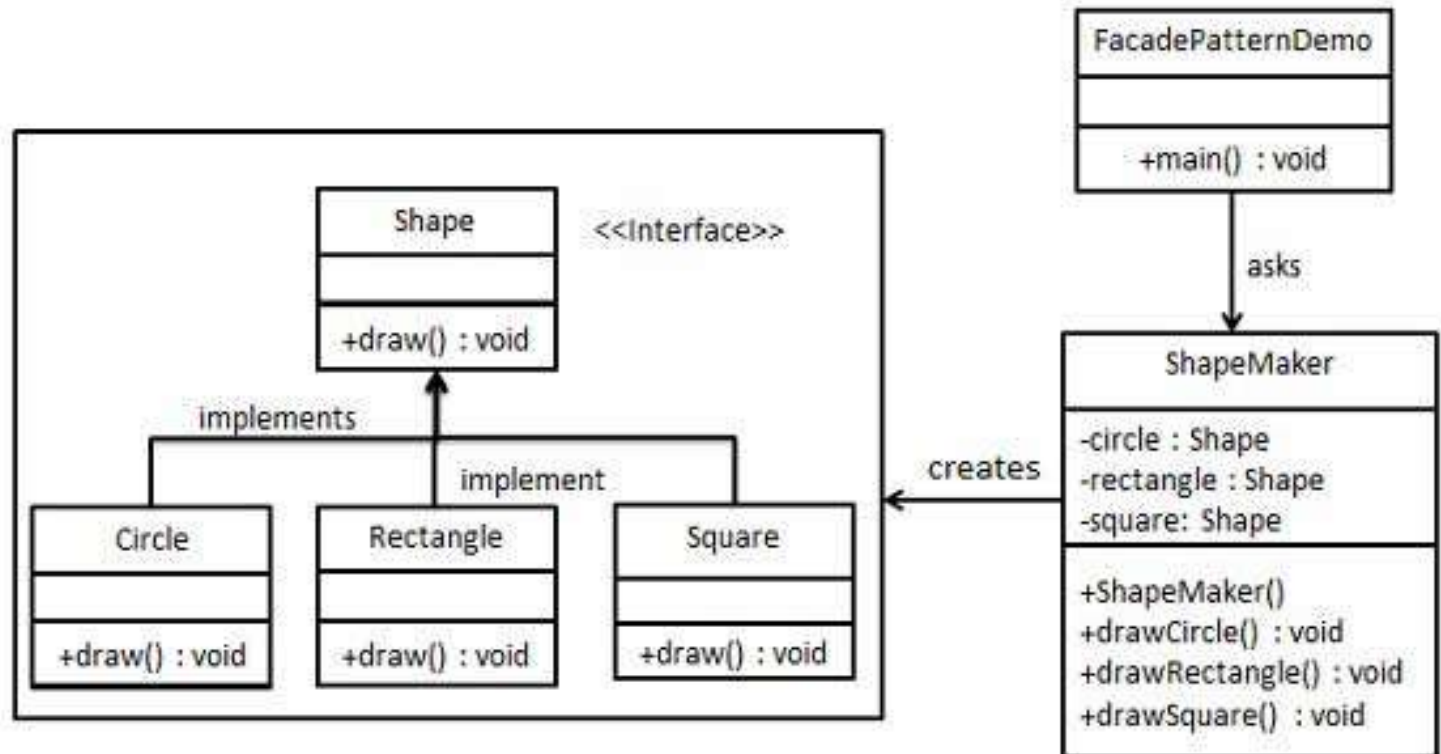
Façade Design Pattern

Façade Design 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.

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.



Step 1

- Create an interface.
- *Shape.java*

```
public interface Shape {  
    void draw();  
}
```

Step 2

- Create concrete classes implementing same interface.

```
public class Rectangle implements Shape {  
    @Override  
    public void draw() {  
        System.out.println("Rectangle::draw()"); }  
}
```

Rectangle.java

```
public class Square implements Shape {  
    @Override  
    public void draw() {  
        System.out.println("Square ::draw()"); }  
}
```

Square.java

```
public class Circle implements Shape {  
    @Override  
    public void draw() {  
        System.out.println("Circle::draw()"); }  
}
```

Circle.java

Step 3

- Create a facade class.
- *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.
- *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.

Circle::draw()

Rectangle::draw()

Square::draw()