Design Pattern - Factory Pattern

设计模式-工厂模式

Factory pattern is one of most used design pattern in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

工厂模式是Java中最常用的设计模式之一。这种类型的设计模式属于创建模式下，创建一个对象最好的方式之一。

在工厂模式中，我们不会把创建对象的逻辑暴露给客户端，同时通过使用通用接口来创建对象引用。

Implementation

实现

We're going to create a *Shape* interface and concrete classes implementing the *Shape* interface. A factory class *ShapeFactory* is defined as a next step.

*FactoryPatternDemo*, our demo class will use *ShapeFactory* to get a *Shape*object. It will pass information (*CIRCLE / RECTANGLE / SQUARE*) to *ShapeFactory* to get the type of object it needs.

我们将创建一个Shape接口以及实现这个Shape接口的类。接下来一步，我们会定义一个工厂类ShapeFactory。

FactoryPatternDemo，我们的demo类将通过图形工厂来获得图形对象。通过传送信息（圆、矩形、正方形）到图形工厂来获得我们所需要的对象。



Step 1

第一步

Create an interface.

创建一个接口

*Shape.java*

public interface Shape {

void draw();

}

Step 2

第二步

Create concrete classes implementing the same interface.

创建具体的实现类来实现相同的接口（shape interface）。

*Rectangle.java*

public class Rectangle implements Shape {

@Override

public void draw() {

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

}

}

*Square.java*

public class Square implements Shape {

@Override

public void draw() {

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

}

}

*Circle.java*

public class Circle implements Shape {

@Override

public void draw() {

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

}

}

Step 3

第三步

Create a Factory to generate object of concrete class based on given information.

创建一个工厂基于传送的信息生成具体类的对象。

*ShapeFactory.java*

public class ShapeFactory {

//use getShape method to get object of type shape

public Shape getShape(String shapeType){

if(shapeType == null){

return null;

}

if(shapeType.equalsIgnoreCase("CIRCLE")){

return new Circle();

} else if(shapeType.equalsIgnoreCase("RECTANGLE")){

return new Rectangle();

} else if(shapeType.equalsIgnoreCase("SQUARE")){

return new Square();

}

return null;

}

}

Step 4

第四步

Use the Factory to get object of concrete class by passing an information such as type。

使用工厂通过传送过来的类型信息获得具体类的对象。

*FactoryPatternDemo.java*

public class FactoryPatternDemo {

public static void main(String[] args) {

ShapeFactory shapeFactory = new ShapeFactory();

//get an object of Circle and call its draw method.

Shape shape1 = shapeFactory.getShape("CIRCLE");

//call draw method of Circle

shape1.draw();

//get an object of Rectangle and call its draw method.

Shape shape2 = shapeFactory.getShape("RECTANGLE");

//call draw method of Rectangle

shape2.draw();

//get an object of Square and call its draw method.

Shape shape3 = shapeFactory.getShape("SQUARE");

//call draw method of circle

shape3.draw();

}

}

Step 5

第五步

Verify the output.

校验输出。

Inside Circle::draw() method.

Inside Rectangle::draw() method.

Inside Square::draw() method.