Design Patterns - Flyweight Pattern

设计模式-享元模式

Flyweight pattern is primarily used to reduce the number of objects created and to decrease memory footprint（足迹） and increase performance. This type of design pattern comes under structural pattern as this pattern provides ways to decrease object count thus improving the object structure of application.

享元模式主要用于减少创建对象的数量，并且以此来减少内存的占用，从而提高系统的性能。这种类型的设计模式属于结构型模式的一种，该模式提供方法减少对象的数量，从而改善应用的对象结构。

Flyweight pattern tries to reuse already existing similar kind objects by storing them and creates new object when no matching object is found. We will demonstrate this pattern by drawing 20 circles of different locations but we will create only 5 objects. Only 5 colors are available so color property is used to check already existing *Circle* objects.

享元模式尝试以存储对象的方式重新利用那些已经存在的种类相似的对象，并且在没有发现匹配对象的时候会创建新的对象。我们将演示该模式，通过在不同位置画20个的圆，但我们仅仅只会创建5个对象的方式。仅仅会有5种颜色被使用，所以颜色属性可以用来检测已经存在的Circle对象。

Implementation

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

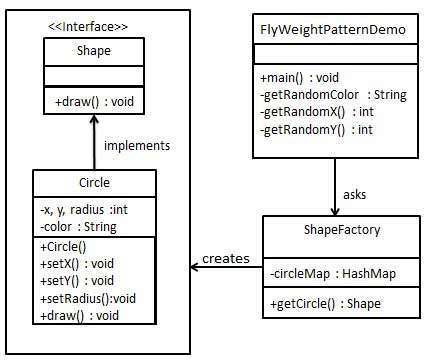
我们将创建一个接口Shape，然后一个具体的类Circle实现Shape接口。接下来我们会声明一个工厂类ShapeFactory。

*ShapeFactory* has a *HashMap* of *Circle* having key as color of the *Circle* object. Whenever a request comes to create a circle of particular color to*ShapeFactory*, it checks the circle object in its *HashMap*, if object of *Circle*found, that object is returned otherwise a new object is created, stored in hashmap for future use, and returned to client.

ShapeFactory有一个存储Circle 的HashMap，该数据结构的key是color，value是Circle对象。无论请求要求ShapeFactory创建何种颜色的Circle对象，它都会检查Circle对象是否存在于HashMap，如果Circle对象被发现，那么就直接返回；否则会创建一个新的对象，并且把它存储在HashMap以备将来使用，然后返回该对象到客户端。

*FlyWeightPatternDemo*, our demo class, will use *ShapeFactory* to get a *Shape*object. It will pass information (*red / green / blue/ black / white*) to*ShapeFactory* to get the circle of desired color it needs.

FlyWeightPatternDemo, 我们的demo类，将通过ShapeFactory来获取Shape对象，demo类通过传送(*red / green / blue/ black / white*)信息到ShapeFactory来获取它所需要的颜色的圆。



Step 1

Create an interface.

创建一个接口Shape

*Shape.java*

public interface Shape {

void draw();

}

Step 2

Create concrete class implementing the same interface.

创建具体的类来实现Shape接口

*Circle.java*

public class Circle implements Shape {

private String color;

private int x;

private int y;

private int radius;//半径

public Circle(String color){

this.color = color;

}

public void setX(int x) {

this.x = x;

}

public void setY(int y) {

this.y = y;

}

public void setRadius(int radius) {

this.radius = radius;

}

@Override

public void draw() {

System.out.println("Circle: Draw() [Color : " + color + ", x : " + x + ", y :" + y + ", radius :" + radius);

}

}

Step 3

Create a factory to generate （生产）object of concrete class based on given information.

创建一个工厂根据给予的信息创建具体类的对象。

*ShapeFactory.java*

import java.util.HashMap;

public class ShapeFactory {

private static final HashMap<String, Shape> circleMap = new HashMap();

public static Shape getCircle(String color) {

Circle circle = (Circle)circleMap.get(color);

if(circle == null) {

circle = new Circle(color);

circleMap.put(color, circle);

System.out.println("Creating circle of color : " + color);

}

return circle;

}

}

Step 4

Use the factory to get object of concrete class by passing an information such as color.

通过向工厂传送color属性来获取具体类的对象

*FlyweightPatternDemo.java*

public class FlyweightPatternDemo {

private static final String colors[] = { "Red", "Green", "Blue", "White", "Black" };

public static void main(String[] args) {

for(int i=0; i < 20; ++i) {

Circle circle = (Circle)ShapeFactory.getCircle(getRandomColor());

circle.setX(getRandomX());

circle.setY(getRandomY());

circle.setRadius(100);

circle.draw();

}

}

private static String getRandomColor() {

return colors[(int)(Math.random()\*colors.length)];

}

private static int getRandomX() {

return (int)(Math.random()\*100 );

}

private static int getRandomY() {

return (int)(Math.random()\*100);

}

}

Step 5

Verify the output.

校验输出

Creating circle of color : Black

Circle: Draw() [Color : Black, x : 36, y :71, radius :100

Creating circle of color : Green

Circle: Draw() [Color : Green, x : 27, y :27, radius :100

Creating circle of color : White

Circle: Draw() [Color : White, x : 64, y :10, radius :100

Creating circle of color : Red

Circle: Draw() [Color : Red, x : 15, y :44, radius :100

Circle: Draw() [Color : Green, x : 19, y :10, radius :100

Circle: Draw() [Color : Green, x : 94, y :32, radius :100

Circle: Draw() [Color : White, x : 69, y :98, radius :100

Creating circle of color : Blue

Circle: Draw() [Color : Blue, x : 13, y :4, radius :100

Circle: Draw() [Color : Green, x : 21, y :21, radius :100

Circle: Draw() [Color : Blue, x : 55, y :86, radius :100

Circle: Draw() [Color : White, x : 90, y :70, radius :100

Circle: Draw() [Color : Green, x : 78, y :3, radius :100

Circle: Draw() [Color : Green, x : 64, y :89, radius :100

Circle: Draw() [Color : Blue, x : 3, y :91, radius :100

Circle: Draw() [Color : Blue, x : 62, y :82, radius :100

Circle: Draw() [Color : Green, x : 97, y :61, radius :100

Circle: Draw() [Color : Green, x : 86, y :12, radius :100

Circle: Draw() [Color : Green, x : 38, y :93, radius :100

Circle: Draw() [Color : Red, x : 76, y :82, radius :100

Circle: Draw() [Color : Blue, x : 95, y :82, radius :100