Centro Universitário UniBH
Ciência da Computação
Práticas de Programação
Professor: Lucas Schmidt

Aula Prática - 11 Padrões Estruturais - Flyweight

Intenção

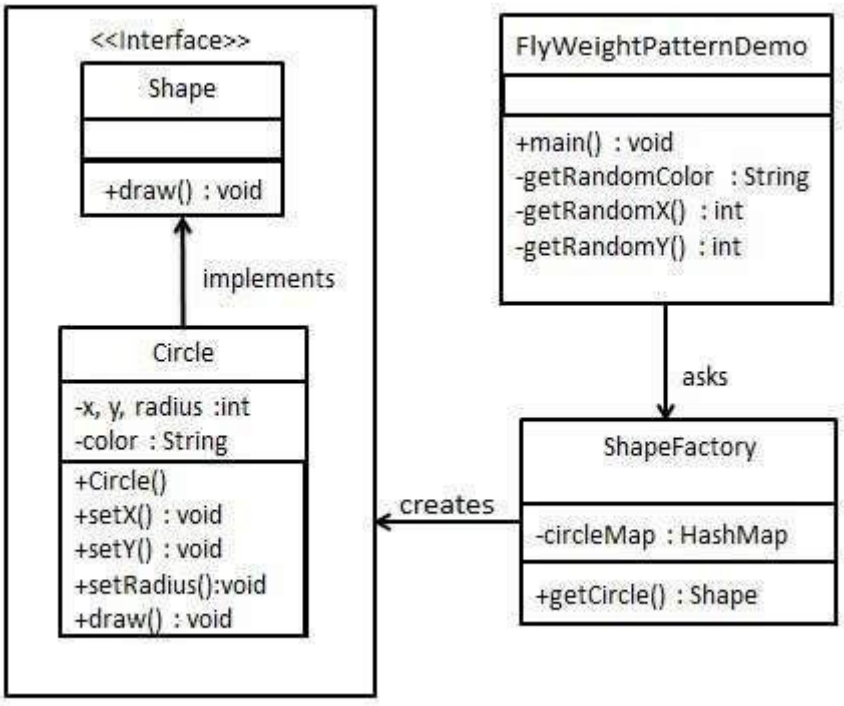
Implantar compartilhamento de objetos de granularidade muito pequena para dar suporte ao uso eficiente de grande quantidade deles.

Usar este padrão quando…

- Todas as condições forem verdadeiras:
  - A aplicação usa um grande número de objetos;
  - O custo de armazenamento é alto por causa desta quantidade;
  - O estado dos objetos pode ser externalizado;
  - Objetos podem ser compartilhados assim que seu estado é externalizado;
  - A aplicação não depende da identidade.

Vantagens e desvantagens

- Custo x benefício:
  - Custo de recuperar o objeto compartilhado e transferir seu estado externalizado;
  - Benefício de economia de recursos.



Passo 1
Crie uma interface.

Shape.java

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

Passo 2
Crie uma classe concreta implementando a mesma interface.

Circle.java

```
public class Circle implements Shape {
   private String color;
   private int x;
   private int y;
   private int radius;

   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);
   }
}
```

Passo 3
Crie uma fábrica para gerar objeto de classe concreta com base em informações dadas.

ShapeFactory.java

```
import java.util.HashMap;

public class ShapeFactory {

   // Uncomment the compiler directive line and
   // javac *.java will compile properly.
   // @SuppressWarnings("unchecked")
   private static final HashMap 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;
   }
}
```

Passo 4
Use a fábrica para adquirir o objeto da classe concreta passando uma informação como cor.

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);
   }
}
```

Passo 5

Teste sua implementação!