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

Aula Prática - 7 Padrões Estruturais - Bridge

Intenção

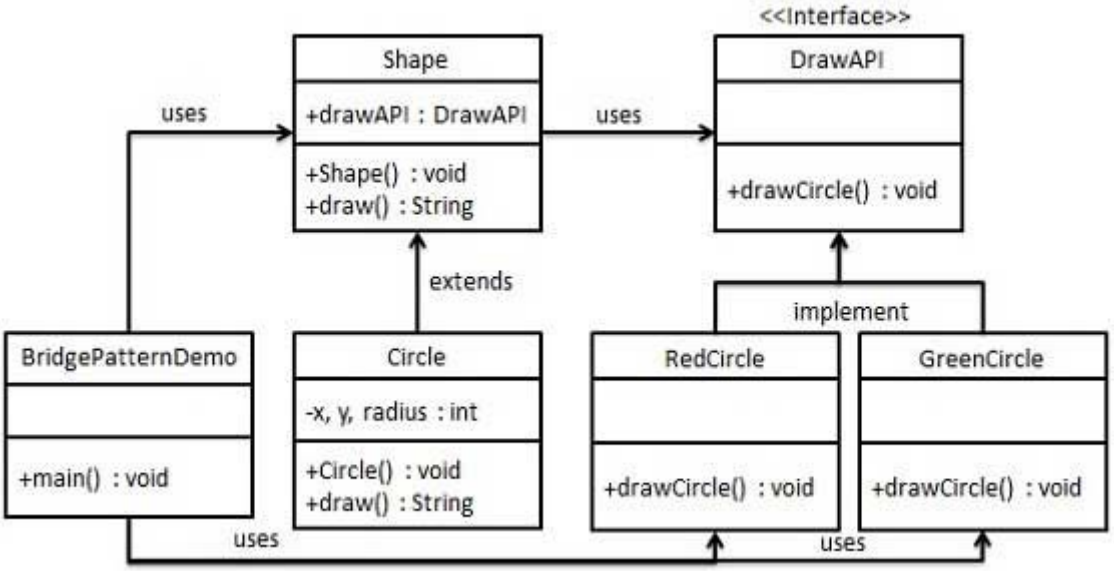
Desacoplar uma abstração de sua implementação para que ambos possam variar independentemente.

Usar este padrão quando…

- Quiser evitar uma ligação permanente entre a abstração e a implementação;
- Tanto a abstração quanto a implementação possuem subclasses;
- Mudanças na implementação não devem afetar o código do cliente;
- Sua atual solução gera uma proliferação de classes (exemplo)

Vantagens e desvantagens

- Desacopla a implementação:
  - Podendo até mudá-la em tempo de execução.
- Melhora a extensibilidade:
  - É possível extender a abstração e a implementação separadamente.
- Esconde detalhes de implementação:
  - Clientes não precisam saber como é implementado.



Passo 1
Crie a interface do implementador de pontes.

DrawAPI.java

```
public interface DrawAPI {
   public void drawCircle(int radius, int x, int y);
}
```

Passo 2
Crie classes do implementador de ponte concreta implementando a interface DrawAPI.

RedCircle.java

```
public class RedCircle implements DrawAPI {
   @Override
   public void drawCircle(int radius, int x, int y) {
      System.out.println("Drawing Circle[ color: red, radius: " + radius + ", x: " + x + ", " + y + "]");
   }
}
```

GreenCircle.java

```
public class GreenCircle implements DrawAPI {
   @Override
   public void drawCircle(int radius, int x, int y) {
      System.out.println("Drawing Circle[ color: green, radius: " + radius + ", x: " + x + ", " + y + "]");
   }
}
```

Passo 3

Crie uma classe abstrata Shape usando a interface DrawAPI.

Shape.java

```
public abstract class Shape {
   protected DrawAPI drawAPI;

   protected Shape(DrawAPI drawAPI){
      this.drawAPI = drawAPI;
   }
   public abstract void draw();
}
```

Passo 4

Crie uma classe concreta implementando a interface Shape.

Circle.java

```
public class Circle extends Shape {
   private int x, y, radius;

   public Circle(int x, int y, int radius, DrawAPI drawAPI) {
      super(drawAPI);
      this.x = x;
      this.y = y;
      this.radius = radius;
   }

   public void draw() {
      drawAPI.drawCircle(radius,x,y);
   }
}
```

Passo 5

Use as classes Shape e DrawAPI para desenhar diferentes círculos coloridos.

BridgePatternDemo.java

```
public class BridgePatternDemo {
   public static void main(String[] args) {
      Shape redCircle = new Circle(100,100, 10, new RedCircle());
      Shape greenCircle = new Circle(100,100, 10, new GreenCircle());

      redCircle.draw();
      greenCircle.draw();
   }
}
```

Passo 6

Teste sua implementação!