## 一句话概括：

将类的**功能**层次结构与**实现**层次结构分离。

## 补充介绍：

桥接模式（Bridge Pattern）涉及到一个作为桥接的接口，使得实体类的功能独立于接口实现类。这两种类型的类可被结构化改变而互不影响。

**类的功能层次结构：**

父类具有基本功能

在子类中增加新的功能

**类的实现层次结构：**

父类通过声明抽象方法来定义接口

子类通过实现具体方法来实现接口

## 参与角色:

1）功能层次结构父类

2）功能层次结构子类

3）实现层次结构父类（接口）

4）实现层次结构子类

功能层次的父类里面引用了实现层次的父类，并且在功能层次子类里面增加了新的方法，可以通过不同的实现层次子类构建不同的功能层次子类。

## 优点：

1、抽象和实现的分离。 2、优秀的扩展能力。 3、实现细节对客户透明。

## 缺点：

桥接模式的引入会增加系统的理解与设计难度，由于聚合关联关系建立在抽象层，要求开发者针对抽象进行设计与编程。

## 使用案例或场景：

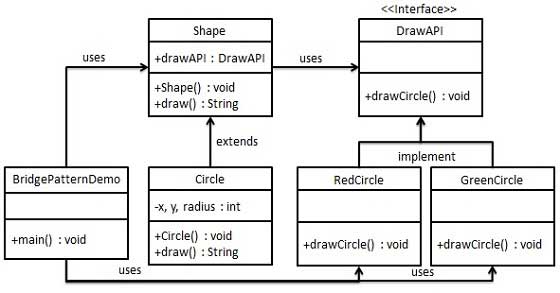
1、如果一个系统需要在构件的抽象化角色和具体化角色之间增加更多的灵活性，避免在两个层次之间建立静态的继承联系，通过桥接模式可以使它们在抽象层建立一个关联关系。 2、对于那些不希望使用继承或因为多层次继承导致系统类的个数急剧增加的系统，桥接模式尤为适用。 3、一个类存在两个独立变化的维度，且这两个维度都需要进行扩展。

注意事项：对于两个独立变化的维度，使用桥接模式再适合不过了。

## 示例程序

**程序简介：**

我们有一个作为桥接实现的 *DrawAPI* 接口和实现了 *DrawAPI* 接口的实体类 *RedCircle*、*GreenCircle*。*Shape* 是一个抽象类，将使用 *DrawAPI* 的对象。*BridgePatternDemo*，我们的演示类使用 *Shape* 类来画出不同颜色的圆。



步骤 1

创建桥接实现接口。

DrawAPI.java

public interface DrawAPI {

public void drawCircle(int radius, int x, int y);

}

步骤 2

创建实现了 DrawAPI 接口的实体桥接实现类。

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 +"]");

}

}

步骤 3

使用 DrawAPI 接口创建抽象类 Shape。

Shape.java

public abstract class Shape {

protected DrawAPI drawAPI;

protected Shape(DrawAPI drawAPI){

this.drawAPI = drawAPI;

}

public abstract void draw();

}

步骤 4

创建实现了 Shape 接口的实体类。

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);

}

}

步骤 5

使用 Shape 和 DrawAPI 类画出不同颜色的圆。

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();

}

}

步骤 6

执行程序，输出结果：

Drawing Circle[ color: red, radius: 10, x: 100, 100]

Drawing Circle[ color: green, radius: 10, x: 100, 100]

参考：

《图解设计模式》【日】结城浩著

《桥接模式》菜鸟教程网站