**基于类的adapter模式**

interface ITravel{

void travel();//出差

}

class KongZhong{//空中网有两名员工

public void A() {//员工A

System.out.println(“我是员工A”);

}

public void B() {//员工B

System.out.println(“我是员工B”);

}

}

class AdapterTravel extends KongZhong implements ITravel {

public void Travel() {

this.A(); //调入我们公司的A员工到南京

this.B(); //调入我们公司的B员工到南京

System.out.println(“我们要去出差了”);

}

}

//执行开会的结果 派员工出差

static void Main(String[] args){

ITravel adapter = new AdapterTravel();

adapter.Travel(); //叫两位员工到南京出差

}

执行结果：

我是员工A

我是员工B

我们要去出差了

**基于对象的adapter模式**

class AdapterTravel implements ITravel {

KongZhong kong;

public AdapterTravel() {

KongZhong kong = new KongZhong();

}

public void Travel() {

kong.A(); //调入我们公司的A员工到南京

kong.B(); //调入我们公司的B员工到南京

System.out.println(“我们要去出差了”);

}

}

类适配器模式是以继承方式实现，增加模块耦合度

对象适配器以组合方式(建立对象实例)试下，降低了耦合度

3D图形的接口和类

public interface Shape3D {

public void get3DLocation(); //获得图形位置

public void set3DLocation(); //设置图形位置

public void display3D(); //显示3D图形

}

public class Ball implements Shape3D{  
 public void get3DLocation();{ System.out.println(“获取[3D球形]位置”); }

public void set3DLocation();{ System.out.println(“设定[3D球形]位置”); }

public void display3D(); { System.out.println(“显示[3D球形]位置”); }

}

有一个新的2D接口和类

public interface Shape2D {

public void get2DLocation(); //获得图形位置

public void set2DLocation(); //设置图形位置

public void display2D(); //显示2D图形

}

public class Circle implements Shape2D{  
 public void get2DLocation();{ System.out.println(“获取[2D圆形]位置”); }

public void set2DLocation();{ System.out.println(“设定[2D圆形]位置”); }

public void display2D(); { System.out.println(“显示[2D圆形]位置”); }

}

三维适配器

public class Adapter2DTo3D implements Shape2D{

Shape3D shape3D;

public Adatper2DTo3D(Shape3D shape3D{

System.out.println(“2D图形转换成3D图形”);

this.shape3D = shape3D;

}

public void display2D(); { shape3D.display3D(); }

public void get2DLocation();{ shape3D. get3DLocation(); }

public void set2DLocation();{ shape3D.set3DLocation(); }

}

使用

public class MainRun{

public static void main(String[] args) {  
 Shape3D ball = new Ball();

Adapter2DTo3D adapter2DTo3D = new Adapter2DTo3D(ball);

adapter2DTo3D.display2D();

adapter2DTo3D.get2DLocation();

adapter2DTo3D.set2DLocation();

}

二维适配器

public class Adapter3DTo2D implements Shape3D{

Shape2D shape2D;

public Adatper3DTo2D(Shape2D shape2D{

System.out.println(“3D图形转换成2D图形”);

this.shape2D = shape2D;

}

public void display3D(); { shape2D.display2D(); }

public void get3DLocation();{ shape2D. get2DLocation(); }

public void set3DLocation();{ shape2D.set2DLocation(); }

}

http://www.docin.com/p-429901154.html