04 外观模式、建造者模式

C-3 创建: 张林伟, 最后修改: 张林伟 2018-10-19 09:18

外观模式

为子系统中的一组接口提供一个一致的界面，此模式定义了一个高层接口，这个接口使得这一子系统更加容易使用。

首先，在设计初期阶段，应该要有意识地将不同的两个层次分离；

其次，在开发阶段，子系统往往因为不断地重构演化而变得越来越复杂，增加外观 Facade 可以提供一个简单的接口，减少它们之间的依赖；

第三，在维护一个遗留的大型系统时，可能这个系统以及非常难以维护和扩展了，为新系统开发一个外观 Facade 类，来提供设计粗糙或高度复杂的遗留代码的比较清晰简单的接口，让新系统与 Facade 对象交互，Facade 与遗留代码交互所有复杂的工作。

Demo

子系统

代码块 Java

```java
// Blue.java
public class Blue {
    public void draw() {
        System.out.println("draw blue");
    }
}

// Red.java
public class Red {
    public void draw() {
        System.out.println("draw red");
    }
}
```

外观类

代码块 Java

```java
public class Facade {

    private Blue blue;
    private Red red;

    public Facade() {
        blue = new Blue();
        red = new Red();
    }

    public void drawBlue() {
        blue.draw();
    }

    public void drawRed() {
        red.draw();
    }
}
```

客户端

代码块 Java

```java
public class Client {

    public static void main(String[] args) {
        Facade facade = new Facade();

        facade.drawBlue();
        facade.drawRed();
    }
```

```
}
```

建造者模式（生成器模式）

将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

建造者模式是在当创建复杂对象的算法应该独立于该对象的组成部分以及它们的装配方式时使用的模式。

Demo

eg.模拟饭店订单生成

商品打包

代码块 Java

```java
// Packing.java
public interface Packing {
    public void pack();
}

// Wrapper.java
public class Wrapper implements Packing {
    @Override public void pack() {
        System.out.println("纸打包");
    }

    @Override
    public String toString() {
        return "纸打包";
    }
}

// Bowl.java
public class Bowl implements Packing {
    @Override public void pack() {
        System.out.println("碗打包");
    }

    @Override
    public String toString() {
        return "碗打包";
    }
}

// Bottle.java
public class Bottle implements Packing {
    @Override public void pack() {
        System.out.println("瓶打包");
    }

    @Override
    public String toString() {
        return "瓶打包";
    }
}
```

商品

代码块 Java

```java
// Item.java
public interface Item {
    /**
     * 商品名称
     */
    String getName();
```

```
    /**
     * 商品价格
     */
    float getPrice();
    /**
     * 打包
     */
    void pack();
    /**
     * 获取打包方式
     */
    String getPack();
}
// Burger.java
public class Burger implements Item {

    private final String name = "汉堡";
    private final float price = 12.5f;
    private final Wrapper pack = new Wrapper();

    @Override public String getName() {
        return name;
    }
    @Override public float getPrice() {
        return price;
    }
    @Override public void pack() {
        pack.pack();
    }
    @Override public String getPack() {
        return pack.toString();
    }
}

// Milk.java
public class Milk implements Item {

    private final String name = "牛奶";
    private final float price = 4f;
    private final Bottle pack = new Bottle();

    @Override public String getName() {
        return name;
    }
    @Override public float getPrice() {
        return price;
    }
    @Override public void pack() {
        pack.pack();
    }
    @Override public String getPack() {
        return pack.toString();
    }
}

// BeefNoodle.java
public class BeefNoodle implements Item {

    private final String name = "牛肉面";
    private final float price = 18f;
    private final Bowl pack = new Bowl();

    @Override public String getName() {
        return name;
```

```java
    }
    @Override public float getPrice() {
        return price;
    }
    @Override public void pack() {
        pack.pack();
    }
    @Override public String getPack() {
        return pack.toString();
    }
}

// Beer.java
public class Beer implements Item {

    private final String name = "啤酒";
    private final float price = 5.6f;
    private final Bottle pack = new Bottle();

    @Override public String getName() {
        return name;
    }
    @Override public float getPrice() {
        return price;
    }
    @Override public void pack() {
        pack.pack();
    }
    @Override public String getPack() {
        return pack.toString();
    }
}
```

订单

代码块 Java

```java
public class Order {

    private List<Item> items;

    public Order() {
        items = new ArrayList<>();
    }

    public void addItem(Item item) {
        items.add(item);
    }

    public void clean() {
        items.removeAll(items);
    }

    public float getCost() {
        float cost = 0;
        for(Item item: items) {
            cost += item.getPrice();
        }
        return cost;
    }

    public void showItems() {
        for(Item item: items) {
            System.out.println("Item: " + item.getName() + " " + item.getPrice() + " " + item.getPack());
        }
    }
```

```
}
```

建造类

代码块 Java

```java
public class OrderBuilder {

    private Order order;

    public OrderBuilder() {
        order = new Order();
    }

    public Order buildBreakfast(Integer num) {
        order.clean();
        for (int i = 0; i < num; i++) {
            order.addItem(new Burger());
            order.addItem(new Milk());
        }
        return order;
    }

    public Order buildLunch(Integer num) {
        order.clean();
        for (int i = 0; i < num; i++) {
            order.addItem(new BeefNoodle());
            order.addItem(new Beer());
        }
        return order;
    }

}
```