[置顶]设计模式是什么鬼（建造者）

凸凹 设计模式 2018年12月27日 876 0 0

本文作者：凸凹

建造者，用于对复杂对象的构造、初始化，与工厂模式不同的是，建造者的目的在于把复杂构造过程从不同对象展现中抽离出来，使得同样的构造工序可以展现出不同的产品对象。

打个比方，我们知道角色扮演类游戏中玩家可以选择不同的职业，各职业攻击力、防御力等等属性设定是不同的，比如战士的力量和体力强，法师的灵力强而体力弱，以及穿戴各种装备引起的属性附加。

假设我们用同一个类来描述这些角色，那么应该怎样新建人物并配备初始武器？交给客户端去完成，把法师配成战士的力量并给脑袋上装备一把屠龙刀吗？诚然，客户端根本不知道怎样去配置（更没有必要知道），游戏人设应该交给专业的团队（建造者模式）去完成，否则会造成不可预知的混乱角色，如同怪物一般的bug存在。

好了，让我们来规划一下专业的建造团队。既然是建造者，那就应该造点建造物了，我们就以盖房子举例。

房子本身有很多个组成部分，各组件息息相关缺一不可，否则房倒屋塌。而其构造过程也是相当复杂的，但大家不必担心，为响应我们简约直观的一贯宗旨，这里只将其简化拆分成地基、墙体、屋顶三部分，首先来看建筑物类。

```
1  public class Building {// 建筑物
2
3      // 用来模拟房子组件的堆叠
4      private List<String> buildingComponents = new ArrayList<>();
5
6      public void setBasement(String basement) {// 地基
7          this.buildingComponents.add(basement);
8      }
9
10     public void setWall(String wall) {// 墙体
11         this.buildingComponents.add(wall);
12     }
13
14     public void setRoof(String roof) {// 房顶
15         this.buildingComponents.add(roof);
16     }
17
18     @Override
19     public String toString() {
20         String buildingStr = "";
21         for (int i = buildingComponents.size() - 1; i >= 0; i--) {
22             buildingStr += buildingComponents.get(i);
23         }
24         return buildingStr;
25     }
26
27 }
```

为了模拟建筑物通用类中各组件的建造顺序，我们在第4行以List来模拟三个组件的堆叠，之后是它们对应的三个建造方法，最后于第19行的toString方法自下而上的打印出最终完成的房子。看起来是不难，但怎样从这个类直

凸凹 官方
关注 私信
凸凹里歐，十余年以上开发经验，《设计
列文章作者，旨在用最通俗的方式诠释设

最新文章
设计模式是什么鬼（命令模式）
设计模式是什么鬼（建造者）
设计模式是什么鬼（抽象工厂）
设计模式是什么鬼（工厂方法）
设计模式是什么鬼（桥接）

极客快讯
Html2excel 1.3.0 版本发布，POI 新
2018年12月30日 40
Apache NetBeans 10.0 正式发布，
2018年12月30日 28
近期国际油价大跌，财报告诉你中石
2018年12月29日 84
“极速抢票”拼不过普通购票 春运抢
坑?
2018年12月29日 82
12月 Web 服务器调查：nginx 增长
份额最高
2018年12月29日 68

```
1  public interface Builder {//施工方接口
2
3      public void buildBasement();
4
5      public void buildWall();
6
7      public void buildRoof();
8
9      public Building getBuilding();
10
11 }
```

既然是施工方的接口，那一定有实现类了，先来写一个高端别墅施工队。

```
1  public class HouseBuilder implements Builder {//别墅施工方
2
3      private Building house;
4
5      public HouseBuilder() {
6          house = new Building();
7      }
8
9      @Override
10     public void buildBasement() {
11         System.out.println("挖地基，部署管道、线缆，水泥加固，搭建围墙、花园。");
12         house.setBasement("╬╬╬╬╬╬╬╬n");
13     }
14
15     @Override
16     public void buildWall() {
17         System.out.println("搭建木质框架，石膏板封墙并粉饰内外墙。");
18         house.setWall("| 田 | 田  田|n");
19     }
20
21     @Override
22     public void buildRoof() {
23         System.out.println("建造木质屋顶、阁楼，安装烟囱，做好防水。");
24         house.setRoof("/◥▆▆◣n");
25     }
26
27     @Override
28     public Building getBuilding() {
29         return house;
30     }
31
32 }
```

嗯，这个施工方看起来是有施工资质的，不管是地基、墙体还是屋顶都讲得（第11行等）头头是道，虽然我们不懂，但建造工艺（第12行等）看起来也都是非常有考究，总之是极其专业的施工方并统统实现了每个组件的建造方法，下来我们同样地再请一个公寓楼的施工队。

```
1  public class ApartmentBuilder implements Builder {// 高层公寓楼施工方
2
3      private Building apartment;
4
5      public ApartmentBuilder() {
6          apartment = new Building();
7      }
8
9      @Override
10     public void buildBasement() {
11         System.out.println("深挖地基，修建地下车库，部署管道、线缆、风道。");
12         apartment.setBasement("╚═══════╝n");
13     }
14
15     @Override
16     public void buildWall() {
17         System.out.println("搭建多层建筑框架，建造电梯井，钢筋混凝土浇灌。");
18         for (int i = 0; i < 8; i++) {// 此处假设固定8层
19             apartment.setWall("╟ □ □ □ □ ╢n");
20         }
21     }
22
23     @Override
24     public void buildRoof() {
25         System.out.println("封顶，部署通风井，做防水层，保温层。");
```


标签聚合

博客	(247)	源码分析
Java源码解析	(181)	python
Java面试题	(96)	python
Java基础	(73)	Oracle
springboot	(67)	笔记
Spring	(62)	设计模式
Linux	(53)	LeetCo
刷题	(46)	Java面试

```
30      public Building getBuilding() {
31          return apartment;
32      }
33
34  }
```

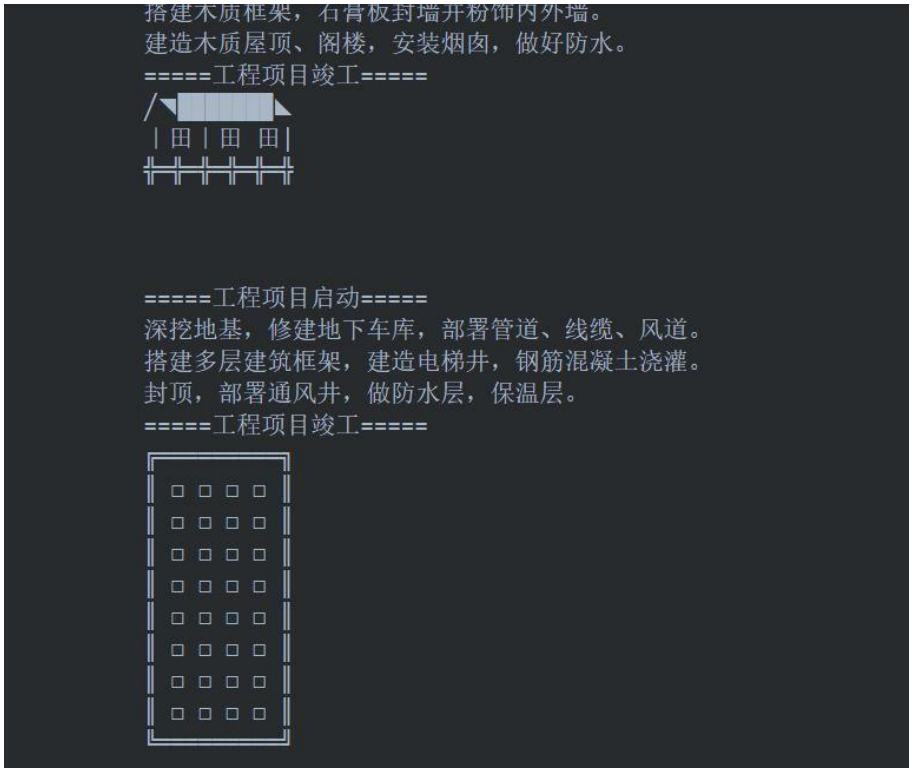
大同小异，公寓楼施工方也是一样的专业，只不过建造工艺好像有些不同，尤其是第16行的建造墙体方法，好像是循环8次造了8层楼的样子，这一定是八层公寓小高层了。到这里，我们好像还是不能交给客户端去亲自调用这三个组件的建造方法，造房子可不是开玩笑的，我们还是得找一个有资质的工程总监去控制整个建造工序流程。

```
1   public class Director {//工程总监
2
3       private Builder builder;
4
5       public Director(Builder builder) {
6           this.builder = builder;
7       }
8
9       public void setBuilder(Builder builder) {
10          this.builder = builder;
11      }
12
13      public Building direct() {
14          System.out.println("=====工程项目启动=====");
15          // 第一步，打好地基；
16          builder.buildBasement();
17          // 第二步，建造框架、墙体；
18          builder.buildWall();
19          // 第三步，封顶；
20          builder.buildRoof();
21          System.out.println("=====工程项目竣工=====");
22          return builder.getBuilding();
23      }
24
25  }
```

我们可以看到，工程总监在宏观上操控着整个施工队的建造流程，在第13行的指导方法中以自下而上的顺序建造房屋，他并不在乎是哪个施工队来造房子，但施工步骤是由他来控制的。是时候满足客户的住房刚需了，组建团队，运行程序。

```
1   public class Client {
2
3       public static void main(String[] args) {
4           //招工，建别墅。
5           Builder builder = new HouseBuilder();
6           //交给工程总监
7           Director director = new Director(builder);
8           System.out.println(director.direct());
9           //替换施工方，建公寓。
10          director.setBuilder(new ApartmentBuilder());
11          System.out.println(director.direct());
12      }
13
14  }
```

可以看到客户端先找了个别墅施工队（第5行），并且安排给总监（第7行），于是造出了别墅，接着又替换了另一个公寓楼施工队（第10行），最终顺利地建了一栋八层公寓，结果如下。



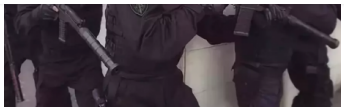
项目终于竣工了，对于复杂对象的构建，专业的建造团队显然是不可或缺的，尤其是产品内部组件间有某种关联性，构建的顺序性，所以我们把制造工序抽离出来交给了工程总监（Director），而产品各种制造工艺则被多态化交给不同的施工方（Builder）去各显神通，最终达成以相同的构造过程生产不同产品的展现的目的，工序不可乱，工艺不可缺。

Java设计模式(17) 设计模式(61) 设计模式是什么鬼(4)

上一篇：ArrayList详解 下一篇：HashMap详解

相关文章

设计模式是什么鬼（代理）



设计模式是什么鬼（策略）

简说桥接模式

23种设计模式介绍-观察者模式

设计模式是什么鬼（单例）

设计模式六大原则（4）：接口隔离原则

发表评论

评分 ☆☆☆☆☆

您的评论一针见血（必填，该内容可在后台设置）

昵称 邮箱

网址

发表评论

繁