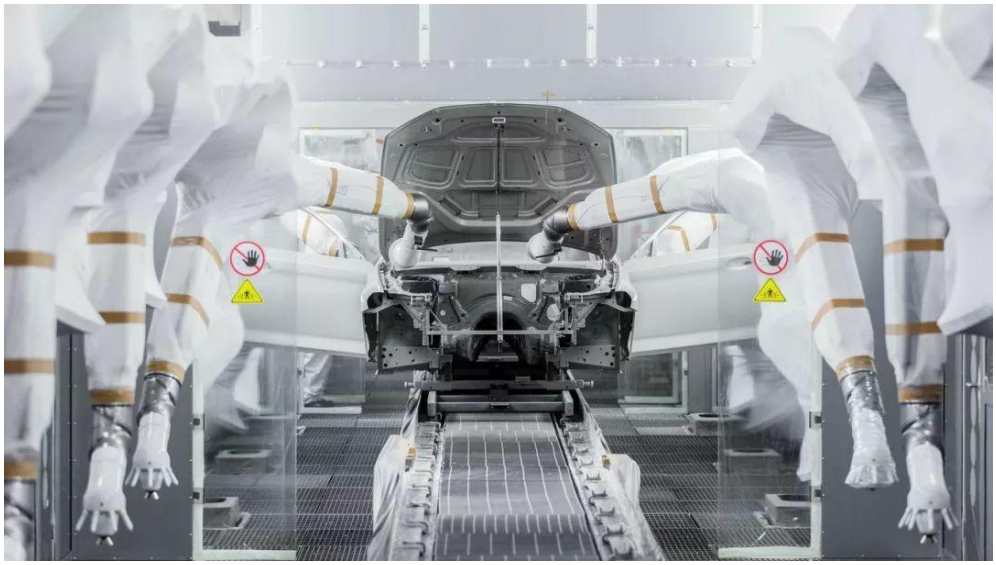
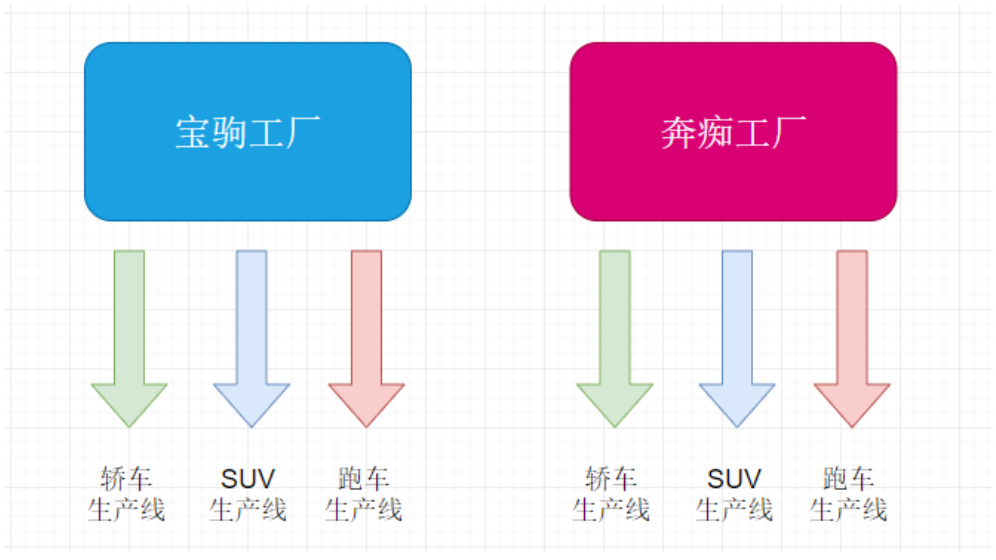
[置顶]设计模式是什么鬼（抽象工厂）

凸凹 设计模式 2018年12月22日 804 0 0

抽象工厂，意味着工厂的泛化，也就是说对多个工厂共通行为的抽取及概括。这和我们之前讲过的工厂方法模式十分类似，不同之处在于抽象工厂定义了更多的抽象行为，也就是多个工厂方法于抽象工厂中，其实它就是工厂方法的变种而已，所以建议读者先理解好工厂方法模式再回来看本章。



我们都知道，在工厂方法模式中的每个实际的工厂只定义了一个工厂方法，如果这些产品可以进行归类分族的话，那么我们便可以顺理成章的定义多个工厂方法，如此可以避免过多的产品造成工厂泛滥的问题。比如宝驹汽车有轿车、SUV、跑车三个等级的产品，而奔痴汽车也同样包括以上三类等级产品，如此便形成了两个产品族，分别由宝驹工厂和奔痴工厂生产，每个工厂都有三个等级的生产线，以及后加入的四环汽车产品族同样可以符合这个规范。



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）

关注	粉丝	点赞
0	27	26

极客快讯

- Html2excel 1.3.0 版本发布，POI 新
  2018年12月30日 32
- Apache NetBeans 10.0 正式发布，
  2018年12月30日 22
- 近期国际油价大跌，财报告诉你中石
  2018年12月29日 68
- “极速抢票”拼不过普通购票 春运抢
  坑?
  2018年12月29日 78
- 12月 Web 服务器调查：nginx 增长
  份额最高
  2018年12月29日 50

繁

我们以一款即时战略游戏来举例，假设游戏中有两个种族，人类与外星怪兽，其中人类拥有各种高科技军工制造技术，而外星怪兽则是以血肉之躯的不断进化与人类抗衡。

在开始代码之前我们先对两族兵种进行分析归纳，显而易见人类兵工厂和怪兽兵工厂（母巢）产出兵种都可以被简单归纳为初、中、高三个等级，如下表所示。

种族\等级	初级兵种	中级兵种	高级兵种
人类族	海军陆战队	变形坦克	巨型战舰
怪兽族	蟑螂	毒液	猛犸

产品族已经定义清楚了，开始建立数据模型。首先定义产品的父类抽象兵种Unit，这里我们使用抽象类。

```
public abstract class Unit {// 兵种

    protected int attack;// 攻击力
    protected int defence;// 防御力
    protected int health;// 血量
    protected int x;// 横坐标
    protected int y;// 纵坐标

    public Unit(int attack, int defence, int health, int x, int y) {
        this.attack = attack;
        this.defence = defence;
        this.health = health;
        this.x = x;
        this.y = y;
    }

    public abstract void show();

    public abstract void attack();

}
```

不管是什么兵种必然会有攻击力、防御力、血量体力值、坐标方位等等属性，我们都定义为protected以供子类继承，除此之外还有两个抽象方法显示和攻击。接下来是人类的产品族海军陆战队士兵、变形坦克和巨型战舰，它们分别对应初、中、高级兵种。

```
public class Marine extends Unit {// 海军陆战队士兵

    public Marine(int x, int y) {
        super(6, 5, 40, x, y);
    }

    @Override
    public void show() {
        System.out.println("士兵出现在坐标：[" + x + "," + y + "]");
    }

    @Override
    public void attack() {
        System.out.println("士兵用机关枪射击，攻击力：" + attack);
    }

}
```

```
    }

    @Override
    public void show() {
        System.out.println("坦克出现在坐标：[" + x + "," + y + "]");
    }

    @Override
    public void attack() {
        System.out.println("坦克用炮轰击，攻击力：" + attack);
    }

}
```

```
public class Battleship extends Unit {// 巨型战舰

    public Battleship(int x, int y) {
        super(25, 200, 500, x, y);
    }

    @Override
    public void show() {
        System.out.println("战舰出现在坐标：[" + x + "," + y + "]");
    }

    @Override
    public void attack() {
        System.out.println("战舰用激光炮打击，攻击力：" + attack);
    }

}
```

可以看到每个兵种的属性值都不同，我们在第4行的构造方法中调用父类构造，直接赋值给继承下来的属性，兵种越高攻击防御越高（当然制造成本也更高，这里我们忽略价格），而且都重写了自己的展示和攻击方法，行为差异化，当然也许坦克会有更另类的行为比如变形什么的，我们此处忽略。然后我们定义外星生物家族的三级兵种，分别是：蟑螂、毒液、猛犸。

```
public class Roach extends Unit {// 外星蟑螂兵

    public Roach(int x, int y) {
        super(5, 2, 35, x, y);
    }

    @Override
    public void show() {
        System.out.println("蟑螂兵出现在坐标：[" + x + "," + y + "]");
    }

    @Override
    public void attack() {
        System.out.println("蟑螂兵用爪子挠，攻击力：" + attack);
    }

}
```

```
public class Spitter extends Unit {// 外星毒液口水兵

    public Spitter(int x, int y) {
        super(10, 8, 80, x, y);
    }

    @Override
    public void show() {
        System.out.println("口水兵出现在坐标：[" + x + "," + y + "]");
    }

    @Override
    public void attack() {
        System.out.println("口水兵用毒液喷射，攻击力：" + attack);
    }

}
```

```
public class Mammoth extends Unit {// 外星猛犸巨兽

    public Mammoth(int x, int y) {
        super(20, 100, 400, x, y);
    }

    @Override
```

```
12      @Override
13      public void attack() {
14          System.out.println("猛犸巨兽用獠牙顶，攻击力：" + attack);
15      }
16
17  }
```

没什么好说的，大同小异。重点来了，接下来是我们的抽象工厂，概括出三个等级的兵种制造方法，我们这里以接口来定义它。

```
1  public interface AbstractFactory {
2
3      public Unit createLowClass();// 工厂方法：制造低级兵种
4
5      public Unit createMidClass();// 工厂方法：制造中级兵种
6
7      public Unit createHighClass();// 工厂方法：制造高级兵种
8  }
```

可以看到，三个等级的接口意味着子类具体工厂必须具备初、中、高级三条生产线，它们同属一个家族、一个品牌的不同系列。理解了这一点后我们可以开始定义人类兵工厂的实现。

```
1  public class HumanFactory implements AbstractFactory{
2
3      //人族工厂坐标
4      private int x;
5      private int y;
6
7      public HumanFactory(int x, int y) {
8          this.x = x;
9          this.y = y;
10     }
11
12     @Override
13     public Unit createLowClass() {
14         Unit unit = new Marine(x, y);
15         System.out.println("制造海军陆战队员成功。");
16         return unit;
17     }
18
19     @Override
20     public Unit createMidClass() {
21         Unit unit = new Tank(x, y);
22         System.out.println("制造变形坦克成功。");
23         return unit;
24     }
25
26     @Override
27     public Unit createHighClass() {
28         Unit unit = new Battleship(x, y);
29         System.out.println("制造巨型战舰成功。");
30         return unit;
31     }
32
33 }
```

可以看到，这个兵工厂实现了人类兵种产品族的制造方法，分别对应三个等级兵种的制造方法，注意第14行意思是在工厂的坐标位置上出兵。下面是异形母巢的工厂实现。

```
1  public class AlienFactory implements AbstractFactory{
2
3      //外星虫族工厂坐标
4      private int x;
5      private int y;
6
7      public AlienFactory(int x, int y) {
8          this.x = x;
9          this.y = y;
10     }
11
12     @Override
13     public Unit createLowClass() {
14         Unit unit = new Roach(x, y);
15         System.out.println("制造蟑螂兵成功。");
16         return unit;
17     }
18
```

```
        return unit;
    }

    @Override
    public Unit createHighClass() {
        Unit unit = new Mammoth(x, y);
        System.out.println("制造猛犸巨兽成功。");
        return unit;
    }

}
```

显而易见，同样地分三级制造异形家族的产品系列，工厂准备完毕可以造兵打架了，运行游戏客户端。

```
public class Client {
    public static void main(String[] args) {
        System.out.println("游戏开始。。。");
        System.out.println("双方挖矿攒钱。。。");

        //第一位玩家选择了地球人族
        System.out.println("工人建造人族工厂。。。");
        AbstractFactory factory = new HumanFactory(10, 10);

        Unit marine = factory.createLowClass();
        marine.show();

        Unit tank = factory.createMidClass();
        tank.show();

        Unit ship = factory.createHighClass();
        ship.show();

        //另一位玩家选择了外星族
        System.out.println("工蜂建造外星虫族工厂。。。");
        factory = new AlienFactory(200, 200);

        Unit roach = factory.createLowClass();
        roach.show();

        Unit spitter = factory.createMidClass();
        spitter.show();

        Unit mammoth = factory.createHighClass();
        mammoth.show();

        System.out.println("两族开始大混战。。。");
        marine.attack();
        roach.attack();
        spitter.attack();
        tank.attack();
        mammoth.attack();
        ship.attack();

        /*
            游戏开始。。。
            双方挖矿攒钱。。。
            工人建造人族工厂。。。
            制造海军陆战队员成功。
            士兵出现在坐标：[10,10]
            制造变形坦克成功。
            坦克出现在坐标：[10,10]
            制造巨型战舰成功。
            战舰出现在坐标：[10,10]
            工蜂建造外星虫族工厂。。。
            制造蟑螂兵成功。
            蟑螂兵出现在坐标：[200,200]
            制造毒液兵成功。
            口水兵出现在坐标：[200,200]
            制造猛犸巨兽成功。
            猛犸巨兽兵出现在坐标：[200,200]
            两族开始大混战。。。
            士兵用机关枪射击，攻击力：6
            蟑螂兵用爪子挠，攻击力：5
            口水兵用毒液喷射，攻击力：10
            坦克用炮轰击，攻击力：25
            猛犸巨兽用獠牙顶，攻击力：20
            战舰用激光炮打击，攻击力：25
         */
    }
}
```

这里我们可以看到，不管玩家选择哪个种族，只要替换工厂实现就可以完成不同兵种的制造，假设玩家又需要一

至此，我们用各族工厂对种类繁多的产品进行了划分、归类，横向拆分出产品家族，纵向则拆分出产品等级。产品虽然繁多，但总得有品牌、型号之分，以各族工厂和产品线划界，分而治之。

本文作者：凸凹

设计模式(61)

分享：

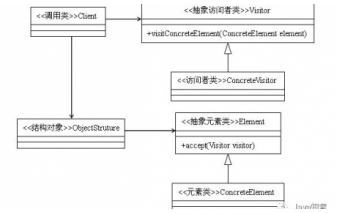
精简阅读 生成封面

赞 0

上一篇：Redis学习指南

下一篇：IntelliJ IDEA 介绍

相关文章

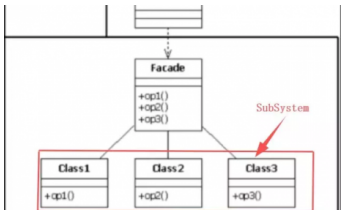


23种设计模式介绍-访问者模式

23种设计模式介绍-备忘录模式

设计模式是什么鬼（原型）

设计模式六大原则（1）：单一职责原则



23种设计模式介绍-外观模式

教练，我想学设计模式

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