853769620

abstract factory模式 -> 简单工厂模式的变种 -> 最复杂的一个变种【工厂定义的是一组对象组合，多和策略模式一块使用】

场景：

（1）就是我们现在要生产的不是一个一个产品，而一个一个的产品组合【调用方，可能有的地方用产品1、2，有的地方用1、3……】

（2）比如说我们有产品ABC，现在第一种产品组合是A+B，第二种产品组合是B+C，第三种产品组合是A+Cs

（3）就是要对工厂模式进行进一步的增强

public **interface ProductA** {

   void operate();

}

public **interface ProductB** {

   void operate();

}

public class **ProductAImpl1 implements ProductA** {

   public void operate() {

       System.out.println("产品A的实现1的功能执行");

  }

}

public class **ProudctBImpl1 implements ProductB** {

   public void operate() {

       System.out.println("产品B的实现1的功能执行");

  }

}

public class **ProductAImpl2 implements ProductA** {

   public void operate() {

       System.out.println("产品A的实现2的功能执行");

  }

}

public class **ProudctBImpl2 implements ProductB** {

   public void operate() {

       System.out.println("产品B的实现2的功能执行");

  }

}

【任何组合都会需要用到产品 A、B】

public **interface AbstractFactory** {

   ProductA createProductA();

   ProudctB createProudctB();

}

public class **FactoryImpl1 implements AbstractFactory** {

private static final FactoryImpl1 instance = new FactoryImpl1();

public static FactoryImpl1 get() {

return instance;

}

   public ProductA createProductA() {

       return new ProductAImpl1();

  }

   public ProductB createProductB() {

       return new ProductBImpl1();

  }

}

public class **FactoryImpl2 implements AbstractFactory** {

private static final FactoryImpl2 instance = new FactoryImpl2();

public static FactoryImpl2 get() {

return instance;

}

   public ProductA createProductA() {

       return new ProductAImpl2();

  }

   public ProductB createProductB() {

       return new ProductBImpl2();

  }

}

这种模式就更复杂了，在实际开发场景中，**更加少见了**。因为其核心思想是，如果需要一个工厂，这个工厂可以生产出相关联的一批产品，然后不同的工厂实现，会生产出一批不同的产品组合。这个，大家只能在这里理解一下这种思想，我们在阶段一的项目实际的系统设计中，跟策略模式结合起来，电商业务，给大家找一个合适的场景来，在业务中实践一下。