**工厂代理模式**

**案例: JavaDemo4**

1. 工厂设计模式
   1. 对于接口而言，已经明确的知道，必须有子类，并且子类可以通过向上转型获得接口的实例化对象。但是在进行对象的实例化过程中也有可能出现设计问题。

范例:

|  |
| --- |
| interface IFood{  abstract void eat();  }  class Bread implements IFood{  public void eat(){  System.out.println("吃面包");  }  }  class milk implements IFood{  public void eat (){  System.out.println("喝牛奶");  }  }  public class JavaDemo4 {  public static void main(String[] args) {  IFood ifood = new Bread(); //造成耦合的元凶 new  ifood.eat();  }  } |

此时客户端(main 方法中) 必须清楚的知道子类是什么才能在Bread 和milk(吃面包和喝牛奶)之间进行转换。这样的程序耦合性太高！

因此我们必须使用工厂设计模式来降低这样的高耦合程序，工厂设计模式，让客户端不再关注子类，只需要知道工厂类和接口即可

范例: 使用工厂设计模式。JavaDemo5

|  |
| --- |
| interface IFood{  abstract void eat();  }  class Bread implements IFood{  public void eat(){  System.out.println("吃面包");  }  }  class Milk implements IFood{  public void eat (){  System.out.println("喝牛奶");  }  }  class FactorFood {  public static IFood getInstance(String ifoodname){  if ("Bread".equals(ifoodname)) {  return new Bread();  }  if("Milk".equals(ifoodname)){  return new Milk();  }  else{  return null;  }  }  }  //java JavaDemo5 Break  //java JavaDemo5 milk  public class JavaDemo5 {  public static void main(String[] args) {  IFood ifood = FactorFood.getInstance(args[0]); //将参数传递给factory  ifood.eat();  }  } |

在该程序中客户端（main）不再关注具体的子类，而是转而关注工厂类和接口，子类相当于隐藏了，当我们当增加子类或者扩展子类时客户端与子类因为没有关联所以无需更改，只需要修改工厂类即可！

1. 代理设计模式
   1. 代理设计模式，目的是为了代替子类进行重复的工作或者说是公共的工作，这会大量减少程序员的代码量。同时，代理类可以增强子类的功能。

范例:实现代理设计

|  |
| --- |
| interface IEat {  abstract void get ();  }  class EatReal implements IEat{  public void get(){  System.out.println("【真实主题】:得到一份真实的美食，开始美美的吃上一顿");  }  }  class EatProxy implements IEat{  private IEat eat;  public EatProxy(IEat eat){  this.eat = eat;  }  public void get(){  this.perpare();  this.cooking();  this.eat.get();  this.clear();  }  public void perpare(){  System.out.println("【代理主题】开始准备食材");  }  public void cooking(){  System.out.println("【代理主题】开始烹饪");  }  public void clear(){  System.out.println("【代理主题】收拾碗筷");  }  }  public class JavaDemo6{  public static void main(String[] args) {  IEat eat = new EatProxy(new EatReal()); //这里应该用工厂设计模式  eat.get();  }  } |

代理设计模式的主要特点就是：一个接口提供有两个子类，一个子类是代理业务操作类一个是真实业务操作类，没有代理业务操作类，真实业务操作类便无法进行。

把代理设计和工厂设计真正理解了才能真正的走上设计的道路上来。