public abstract class KuaiCanFactory    //快餐工厂  
    {

        //两个抽象方法，您吃鸡腿呢，还是汉堡呀?因为是抽象出来的，所以不分商家哦！  
        public abstract JiTui eatJiTui();  
        public abstract HanBao eatHanBao();  
    }

    public class KDJFactory:KuaiCanFactory      {  
        public override JiTui eatJiTui()  
        {  
            return new KDJJiTui();  
        }

        public override HanBao eatHanBao()  
        {  
            return new KDJHanBao();  
        }  
    }

下面就到食物了。。

    public class KDJHanBao:HanBao   //具体的肯德基汉堡  
    {  
        public override void getHanBao()  
        {  
            Console.WriteLine( "肯德基汉堡");  
        }  
    }

    public abstract class JiTui    //鸡腿抽象类  
    {  
        public abstract void getJiTui();  
    }

    public class KDJJiTui:JiTui  //具体的肯德基鸡腿  
    {  
        public override void getJiTui()  
        {  
            Console.WriteLine( "肯德基鸡腿");  
        }  
    }

   这样就大功告成，客户去快餐店吃鸡腿，并不关心他们怎么拿出来的，只需要大吼一声“我吃鸡腿”，就OK了。

        static void Main(string[] args)  
        {  
            //这里或许使用简单工厂来处理选择商店更好  
            KuaiCanFactory KC = new KDJFactory(); //想吃KFC的鸡腿，那么就去KFC。  
            JiTui jt = KC.eatJiTui(); //想吃鸡腿了，那么就告诉KC（快餐店）  
            jt.getJiTui(); //鸡腿拿来咯~~……

            HanBao hb = KC.eatHanBao(); //想吃汉堡  
            hb.getHanBao();

            KC = new MDLFactory();  
            jt = KC.eatJiTui();  
            jt.getJiTui();

            hb = KC.eatHanBao();  
            hb.getHanBao();  
        }