模板方法

```
namespace LeetCode.Properties

    public class 委托
    {
        static void Main(string[] args)
        {
            ProductFactory productFactory=new ProductFactory();
            WarpFactory warpFactory=new WarpFactory();
            Func<Product> func1 = new Func<Product>(productFactory.MakePizza);
            Func<Product> func2 = new Func<Product>(productFactory.MakeToyCar);
            Box box1=warpFactory.WarpProduct(func1);
            Box box2 = warpFactory.WarpProduct(func2);
            Console.WriteLine(box1.Product.Name);
            Console.WriteLine(box2.Product.Name);
        }
    }
    class  Product
    {
        public string Name { get; set; }
    }
    class  Box
    {
        public Product Product { get; set; }
    }
    class WarpFactory
    {
        public Box WarpProduct(Func<Product> getProduct)
        {
            Box box=new Box();
            Product product = getProduct();
            box.Product = product;
            return box;
        }
    }
    class ProductFactory
    {
        public Product MakePizza()
        {
            Product product=new Product();
            product.Name = "Pizza";
            return product;
        }
        public Product MakeToyCar()
        {
            Product product=new  Product();
            product.Name = "ToyCar";
            return product;
        }
    }
}
```

添加回调方法之后

```
namespace LeetCode.Properties
{
    public class 委托
    {

        static void Main(string[] args)
        {
            ProductFactory productFactory=new ProductFactory();
            WarpFactory warpFactory=new WarpFactory();

            Func<Product> func1 = new Func<Product>(productFactory.MakePizza);
            Func<Product> func2 = new Func<Product>(productFactory.MakeToyCar);

            Logger logger=new Logger();

            Action<Product> action1=new Action<Product>(logger.Log);

            Box box1=warpFactory.WarpProduct(func1,action1);
            Box box2 = warpFactory.WarpProduct(func2,action1);
            Console.WriteLine(box1.Product.Name);
            Console.WriteLine(box2.Product.Name);

        }
    }

    class  Product
    {
        public string Name { get; set; }
        public double Price { get; set; }
    }

    class Logger
    {
        public void Log(Product product)
        {
            Console.WriteLine("Product'{0}' created at {1} Price
is{2}",product.Name,DateTime.UtcNow,product.Price);
        }
    }
    class  Box
    {
        public Product Product { get; set; }
    }

    class WarpFactory
    {
        public Box WarpProduct(Func<Product> getProduct,Action<Product> action)
        {
            Box box=new Box();
            Product product = getProduct();
            box.Product = product;
            if (product.Price>18)
            {
                action(product);
            }
            return box;
        }
    }

    class ProductFactory
    {
        public Product MakePizza()
        {
            Product product=new Product();
            product.Name = "Pizza";
            return product;
        }
        public Product MakeToyCar()
        {
            Product product=new  Product();
            product.Price = 20;
            product.Name = "ToyCar";
            return product;
        }
    }
}
```

接口代替委托

```
namespace LeetCode.Properties
{
    public class 委托
    {

        static void Main(string[] args)
        {
            IProductFactory pizzaFactory=new PizzayFactory();
            IProductFactory toycarFactory=new ToyCarFactory();

            WarpFactory warpFactory=new WarpFactory();
            Logger logger = new Logger();

            Action<Product> action = new Action<Product>(logger.Log);
            Box box1=warpFactory.WarpProduct(pizzaFactory,action);
            Box box2 = warpFactory.WarpProduct(toycarFactory, action);

            Console.WriteLine(box1.Product.Name);
            Console.WriteLine(box2.Product.Name);

        }
    }

    class  Product
    {
        public string Name { get; set; }
        public double Price { get; set; }
    }

    interface IProductFactory
    {
        Product Make();
    }

    class PizzayFactory:IProductFactory
    {
        public Product Make()
        {
            Product product=new Product();
            product.Name = "Pizza";
            product.Price = 10;
            return product;
        }
    }

    class ToyCarFactory:IProductFactory
    {
        public Product Make()
        {
            Product product=new  Product();
            product.Price = 20;
            product.Name = "ToyCar";
            return product;
        }
    }
    class Logger
    {
        public void Log(Product product)
        {
            Console.WriteLine("Product'{0}' created at {1} Price 
is{2}",product.Name,DateTime.UtcNow,product.Price);
        }
    }
    class  Box
    {
        public Product Product { get; set; }
    }

    class WarpFactory
    {
        public Box WarpProduct(IProductFactory productFactory,Action<Product> action)
        {
            Box box=new Box();
            Product product = productFactory.Make();
            box.Product = product;
            if (product.Price>18)
            {
                action(product);
            }
            return box;
        }
    }

}
```