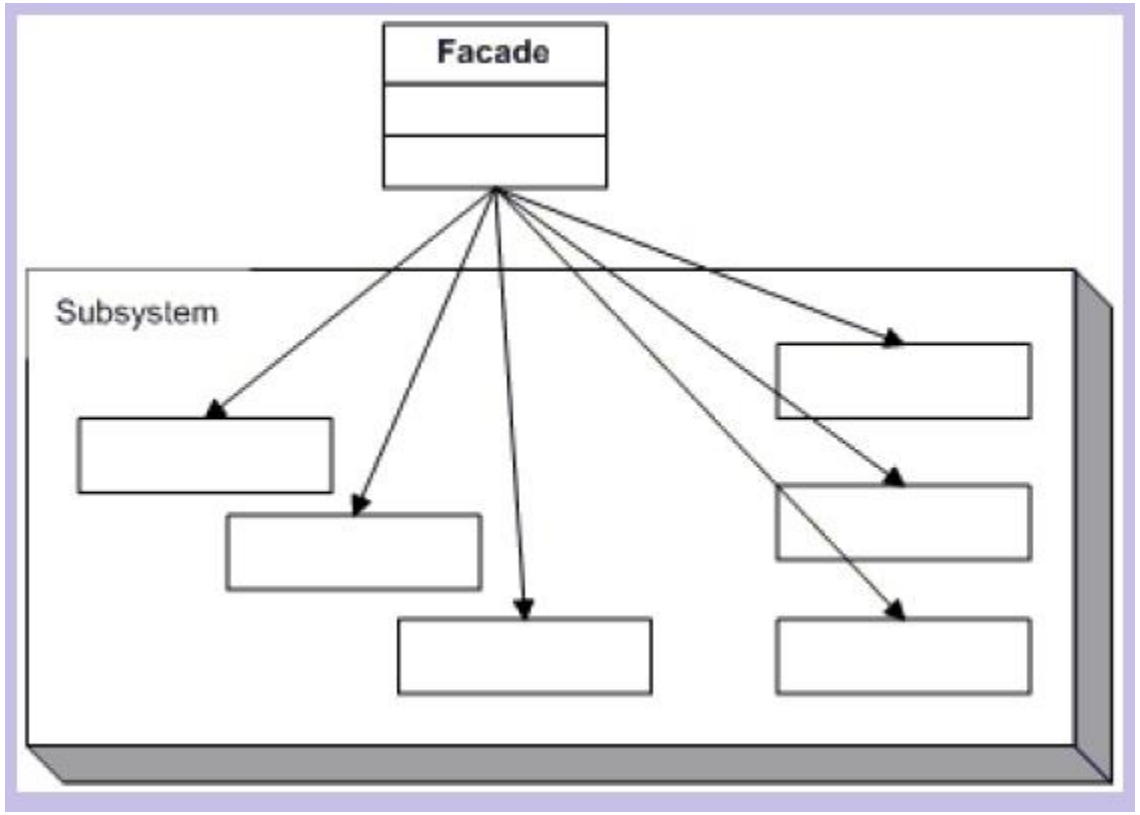
门面模式Facade

波波老师~研发总监/资深架构师

定义

- 为子系统中一组接口提供一个统一的接口
- 门面提供一个高层接口，让子系统易于使用



案例~在线购物商店

- InventoryUpdate
- AddressVerification
- Discounting
- PayCardVerification
- Shipping

代码~库存管理

```java
// 库存接口
public interface IInventory {
    public void update(int productId);
}
```

```java
// 库存实现
public class InventoryManager implements IInventory {

    @Override
    public void update(int productId) {
        String msg = "Product# " + productId +
                " is subtracted from store's inventory";
        System.out.println(msg);
    }

}
```

代码~订单校验

```java
// 订单校验接口
public interface IOrderVerify {
    public boolean verifyShippingAddress(int pincode);
}
```

```java
// 订单校验实现
public class OrderVerificationManager implements IOrderVerify {

    @Override
    public boolean verifyShippingAddress(int pincode) {
        System.out.println(
                "The product can be shipped to the pincode "
                + pincode);
        return true;
    }

}
```

代码~打折计算

```java
// 费用计算接口
public interface ICosting {
    public float applyDiscount(float price, float discountPercent);
}
```

```java
// 费用计算实现
public class CostManager implements ICosting {

    @Override
    public float applyDiscount(float price, float discountPercent) {
        String out = String.format(
                "A discount of %f%% has been applied on the product's price of %f",
                discountPercent, price);
        System.out.println(out);
        return price - ((discountPercent / 100) * price);
    }

}
```

代码~支付处理

```java
// 支付接口
public interface IPaymentGateway {
    public boolean verifyCardDetails(String cardNo);
    public boolean processPayment(String cardNo, float cost);
}
// 支付实现
public class PaymentGatewayManager implements IPaymentGateway {

    @Override
    public boolean verifyCardDetails(String cardNo) {
        String out = "Card# " + cardNo +
                " has been verified and is accepted.";
        System.out.println(out);
        return true;
    }

    @Override
    public boolean processPayment(String cardNo, float cost) {
        String out = "Card# " + cardNo +
                " is used to make a payment of " + cost + ".";
        System.out.println(out);
        return true;
    }

}
```

代码~送货

```java
// 物流接口
public interface ILogistics {
    public void shipProducts(String productName, String shippingAddress);
}


// 物流实现
public class LogisticsManager implements ILogistics {

    @Override
    public void shipProducts(String productName, String shippingAddress) {
        String out = String.format(
            "Congratulations your product %s has been shipped at the following address: %s."
            productName, shippingAddress);
        System.out.println(out);
    }

}
```

代码~订单详情类

```
public class OrderDetails {
    // region 私有成员

    public OrderDetails(String productName, String prodDescription, float price,
            float discount, String addressLine1, String addressLine2,
            int pinCode, String cardNo) {
        this.productNo = new Random(1).nextInt(100);
        this.productName = productName;
        this.productDescription = prodDescription;
        this.price = price;
        this.discountPercent = discount;
        this.addressLine1 = addressLine1;
        this.addressLine2 = addressLine2;
        this.pinCode = pinCode;
        this.cardNo = cardNo;
    }

    // region getters
}
```

客户端代码~不使用门面

```java
// Client Code without Facade.
public class NoFacadeMain {

    public static void main(String[] args) {
        // Creating the Order/Product details
        OrderDetails orderDetails = new OrderDetails("Java Design Pattern book",
                "Simplified book on design patterns in Java",
                500, 10, "Street No 1", "Educational Area", 1212,
                "8811123456");

        // Updating the inventory.
        IInventory inventory = new InventoryManager();
        inventory.update(orderDetails.getProductNo());

        // verifying various details for the order such as the shipping address.
        IOrderVerify orderVerify = new OrderVerificationManager();
        orderVerify.verifyShippingAddress(orderDetails.getPinCode());

        // Calculating the final cost after applying various discounts.
        ICosting costManager = new CostManager();
        orderDetails.setPrice(
                costManager.applyDiscount(
                        orderDetails.getPrice(),
                        orderDetails.getDiscountPercent()
                )
                );

        // Going through various steps if payment gateway like card verification,
        // charging from the card.
        IPaymentGateway paymentGateway = new PaymentGatewayManager();
        paymentGateway.verifyCardDetails(orderDetails.getCardNo());
        paymentGateway.processPayment(orderDetails.getCardNo(), orderDetails.getPrice());

        // Completing the order by providing logistics.
        ILogistics logistics = new LogisticsManager();
        String shippingAddress = String.format("%s, %s - %d",
                orderDetails.getAddressLine1(),
                orderDetails.getAddressLine2(),
                orderDetails.getPinCode());
        logistics.shipProducts(orderDetails.getProductName(), shippingAddress);
    }
}
```

代码~门面实现

```
public class OnlineShoppingFacade {
    IInventory inventory = new InventoryManager();
    IOrderVerify orderVerify = new OrderVerificationManager();
    ICosting costManager = new CostManager();
    IPaymentGateway paymentGateway = new PaymentGatewayManager();
    ILogistics logistics = new LogisticsManager();

    public void finalizeOrder(OrderDetails orderDetails) {
        inventory.update(orderDetails.getProductNo());
        orderVerify.verifyShippingAddress(orderDetails.getPinCode());
        orderDetails.setPrice(
                costManager.applyDiscount(
                        orderDetails.getPrice(),
                        orderDetails.getDiscountPercent()
                )
                );
        paymentGateway.verifyCardDetails(orderDetails.getCardNo());
        paymentGateway.processPayment(orderDetails.getCardNo(), orderDetails.getPrice());
        String shippingAddress = String.format("%s, %s - %d",
                orderDetails.getAddressLine1(),
                orderDetails.getAddressLine2(),
                orderDetails.getPinCode());
        logistics.shipProducts(orderDetails.getCardNo(), shippingAddress);
    }
}
```

客户端代码~使用门面

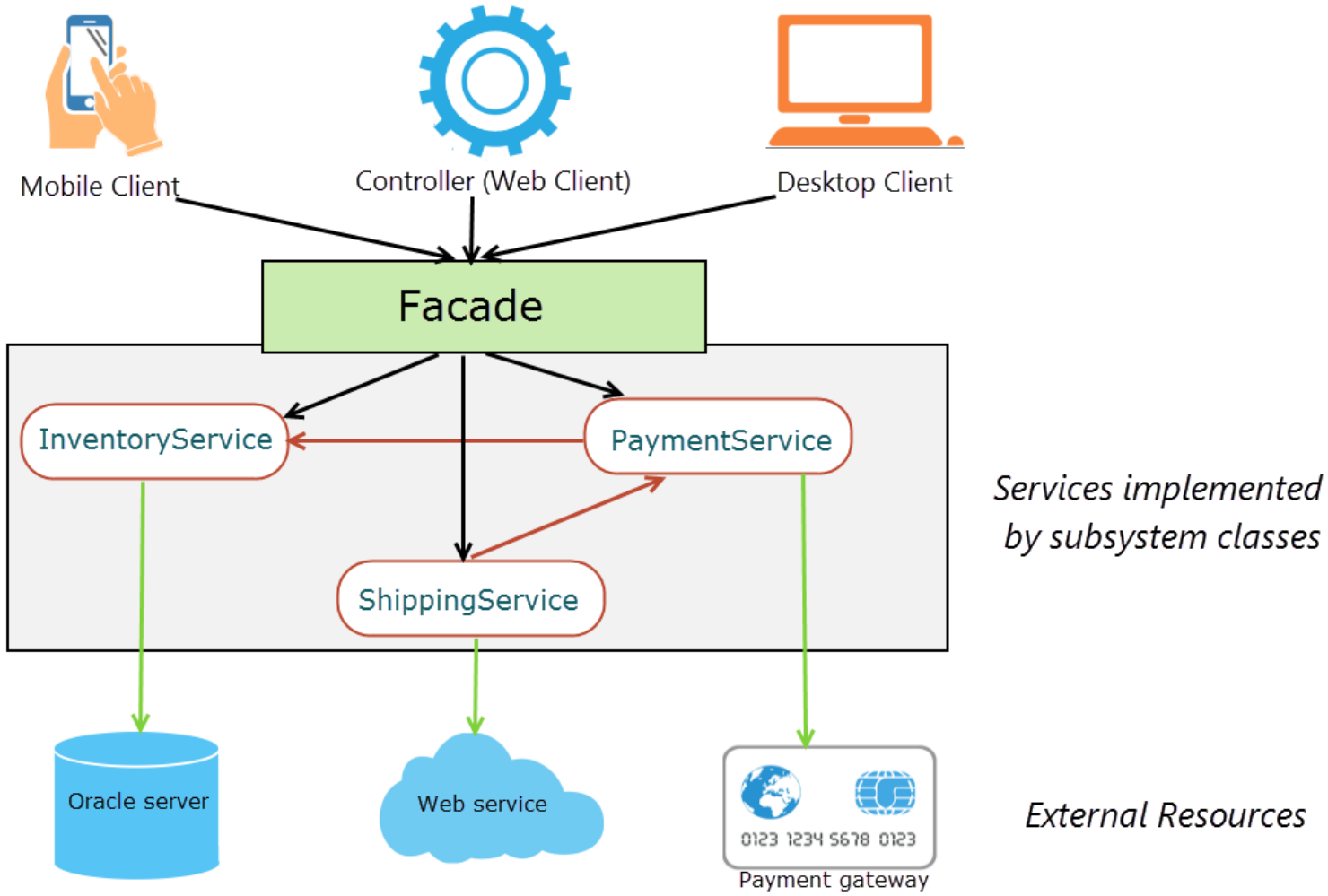
```
public class FacadeMain {

    public static void main(String[] args) {
        // Creating the Order/Product details
        OrderDetails orderDetails = new OrderDetails("Java Design Pattern book",
                "Simplified book on design patterns in Java",
                500, 10, "Street No 1", "Educational Area", 1212,
                "8811123456");

        // Using Facade
        OnlineShoppingFacade facade = new OnlineShoppingFacade();
        facade.finalizeOrder(orderDetails);

    }

}
```

好处

- 提供一个简化接口，让子系统易于使用
- 客户端不需要创建各种子系统对象，门面会负责
- 门面将客户和子系统解耦，子系统代码不污染客户端

应用

- 聚合层服务


Mobile Client
Controller (Web Client)
Desktop Client
Facade
InventoryService
PaymentService
ShippingService
Oracle server
Web service
0123 1234 5678 0123
Payment gateway
Services implemented by subsystem classes
External Resources

问题

- 门面模式和适配器模式的差异？

参考

- Façade Design Pattern(C#)
  - https://www.codeproject.com/Articles/767154/Facade-Design-Pattern-Csharp
- Understanding and Implementing Facade Pattern in C#
  - https://www.codeproject.com/Articles/481297/UnderstandingplusandplusImplementingplusFacadeplus

代码

- https://github.com/spring2go/core-spring-patterns

波波微课

spring2go.com