Analisi del Pattern Facade nel Progetto Habibi Shawarma

Panoramica dell'Implementazione

La tua implementazione utilizza il pattern Facade per fornire un'interfaccia semplificata ai sottosistemi degli use case. Ogni use case ha una propria Facade che nasconde la complessità interna.

Architettura del Sistema

```
flowchart TB
    subgraph "Presentation Layer (Views)"
        GUI1[CreaOrdineGUIController]
        CLI1[CreaOrdineCLIController]
        GUI2[StoricoOrdiniGUIController]
        CLI2[StoricoOrdiniCLIController]
        GUI3[VisualizzaOrdiniGUIController]
        CLI3[VisualizzaOrdiniCLIController]
    end

    subgraph "Facade Layer"
        F1[CreaOrdineFacade]
        F2[StoricoOrdiniFacade]
        F3[VisualizzaOrdiniFacade]
        F4[UsaVoucherFacade]
    end

    subgraph "Controller Layer (Business Logic)"
        C1[CreaOrdineController]
        C2[StoricoOrdiniController]
        C3[VisualizzaOrdiniController]
        C4[UsaVoucherController]
    end

    subgraph "Model/Entity Layer"
        M1[Ordine]
        M2[Food / Decorator]
        M3[Voucher]
        M4[User / Cliente]
    end

    subgraph "Persistence Layer"
        DAO[DAO Factory + DAOs]
    end

    GUI1 --> F1
    CLI1 --> F1
    GUI2 --> F2
```

CLI2 --> **F2**
GUI3 --> **F3**
CLI3 --> **F3**

F1 --> **C1**
F1 --> **C4**
F2 --> **C2**
F3 --> **C3**
F4 --> **C4**

C1 --> M1
C1 --> M2
C1 --> M3
C4 --> M3
C2 --> M1
C3 --> M1

C1 --> DAO
C2 --> DAO
C3 --> DAO
C4 --> DAO

Le 4 Facade del Progetto

Facade	Sottosistema	Controller Interno	Responsabilità
CreaOrdineFacade	Creazione ordini	CreaOrdineController + UsaVoucherController	Gestione completa creazione ordine
StoricoOrdiniFacade	Storico ordini	StoricoOrdiniController	Recupero ordini passati del cliente
VisualizzaOrdiniFacade	Gestione ordini (admin)	VisualizzaOrdiniController	Visualizzazione e gestione stati ordini
UsaVoucherFacade	Voucher	UsaVoucherController	Applicazione/rimozione voucher

---

Conformità con il Pattern GoF

Elementi del Pattern GoF Facade

Elemento GoF	Tua Implementazione	Presente?
Facade (classe che semplifica l'accesso)	*Facade classes	Sì
Sottosistema (classi complesse nascoste)	Controller + DAO + Factory	Sì
Client (usa solo la Facade)	GUI/CLI Controllers	Sì

Valutazione Dettagliata

Aspetti Conformi al GoF

1. Interfaccia Semplificata

Le Facade espongono metodi ad alto livello che nascondono la complessità:

```
// Il client (GUI) chiama un solo metodo
boolean success = facade.aggiungiProdottoAOrdine(richiesta);

// La Facade delega al Controller che gestisce:
// - Creazione prodotto base via Factory Method
// - Applicazione Decorators per gli add-on
// - Aggiunta all'ordine
// - Conversione Entity → Bean
```

2. Accesso Unificato al Sottosistema

I GUI/CLI Controllers usano SOLO la Facade, mai direttamente i componenti interni:

```
// In CreaOrdineGUIController.java
private CreaOrdineFacade facade;

facade = new CreaOrdineFacade(tokenKey);
OrdineBean ordine = facade.inizializzaNuovoOrdine();
List<FoodBean> prodotti = facade.getProdottiBaseDisponibili();
```

3. Delega ai Componenti Interni

La Facade non implementa la logica, delega al Controller:

```
// CreaOrdineFacade.java
public boolean aggiungiProdottoAOrdine(FoodBean foodBean) throws DAOException {
    return controller.aggiungiProdottoAOrdine(foodBean);  // Delega
}
```

4. Gestione delle Dipendenze Interne

I client non conoscono l'esistenza di: DAO, Factory, Session Manager, Entity classes.

```
// La Facade gestisce internamente SessionManager
public CreaOrdineFacade(String tokenKey) throws MissingAuthorizationException {
    this.sessionUser = SessionManager.getInstance().getSessionUserByTokenKey(tokenKey);
    // Validazione autorizzazione
    this.controller = new CreaOrdineController();
}
```

Buone Pratiche Aggiuntive

1. Gestione Centralizzata della Sessione

CreaOrdineFacade integra la validazione della sessione utente, liberando il GUI da questa responsabilità:

```
if (sessionUser == null || sessionUser.getRole() == null ||
    sessionUser.getRole().getClienteRole() == null) {
    throw new MissingAuthorizationException("Accesso negato");
}
```

2. Conversione Entity → Bean nella Facade

La StoricoOrdiniFacade converte direttamente Ordine in OrdineBean, mantenendo isolato il livello entity:

```
public List<OrdineBean> getStoricoOrdini() throws ... {
    List<Ordine> ordini = controller.getOrdiniByCliente(user);
    List<OrdineBean> beans = new ArrayList<>();
    for (Ordine o : ordini) {
        OrdineBean bean = new OrdineBean();
        bean.setNumeroOrdine(o.getNumeroOrdine());
        // ... altri mapping
        beans.add(bean);
    }
    return beans;
}
```

3. Composizione di Use Case

CreaOrdineFacade compone internamente UsaVoucherController, permettendo di applicare voucher senza esporre la complessità:

```
// CreaOrdineController.java
private final UsaVoucherController voucherController;

public VoucherBean applicaVoucher(String codiceVoucher) {
    return voucherController.applicaVoucherAOrdine(ordineCorrente, codiceVoucher);
}
```

Struttura delle Classi per Use Case

CreaOrdine (Use Case Principale)

```
classDiagram
  class CreaOrdineFacade {
    -CreaOrdineController controller
    -User sessionUser
    +inizializzaNuovoOrdine() OrdineBean
    +getProdottiBaseDisponibili() List~FoodBean~
    +getAddOnDisponibili() List~FoodBean~
    +aggiungiProdottoAOrdine(FoodBean) boolean
    +rimuoviProdottoDaOrdine(int) boolean
    +applicaVoucher(String) VoucherBean
    +rimuoviVoucher() void
    +getRiepilogoOrdine() RiepilogoOrdineBean
    +confermaOrdine() boolean
    +annullaOrdine() void
  }

  class CreaOrdineController {
    -Ordine ordineCorrente
    -UsaVoucherController voucherController
    +inizializzaNuovoOrdine(String) OrdineBean
    +aggiungiProdottoAOrdine(FoodBean) boolean
    -creaProdottoBase(String) Food
    -applicaDecorator(Food, String) Food
  }

  class UsaVoucherController {
    +applicaVoucherAOrdine(Ordine, String) VoucherBean
    +rimuoviVoucherDaOrdine(Ordine) void
    +calcolaSconto(Ordine) double
  }

  CreaOrdineFacade --> CreaOrdineController : delega
  CreaOrdineController --> UsaVoucherController : usa
```

Analisi Conformità GoF

Checklist Pattern Facade

Principio GoF	Rispettato?	Evidenza
Fornisce interfaccia semplificata		Metodi come inizializzaNuovoOrdine() nascondono Factory, DAO
Nasconde complessità sottosistema		GUI non conosce Controller, DAO, Entity
Non impedisce accesso diretto		Controller pubblici, ma Views usano solo Facade
Delega ai componenti interni		facade.aggiungiProdotto() → controller.aggiugniProdotto()
Riduce accoppiamento client-sottosistema		GUI dipende solo da Facade e Bean

---

Conclusione

> [!TIP]
> **La tua implementazione del pattern Facade è CORRETTA e BEN PROGETTATA.**

Punti di Forza

1. **Separazione Netta**: GUI/CLI → Facade → Controller → DAO
2. **Isolamento del Domain Model**: I client vedono solo Bean, non Entity
3. **Gestione Centralizzata**: Sessione e autorizzazioni gestite dalla Facade
4. **Composizione**: CreaOrdineFacade compone UsaVoucherController senza esporlo
5. **Consistenza**: Ogni use case segue lo stesso pattern architetturale

Aspetti da Considerare (Non Errori)

> [!NOTE]
> **Osservazione**: StoricoOrdiniFacade e VisualizzaOrdiniFacade fanno anche la conversione Entity→Bean, mentre CreaOrdineFacade delega questo al Controller. Entrambi gli approcci sono validi, ma potresti uniformare.

File Analizzati

File

[CreaOrdineFacade.java](file:///Users/danieledimeo/Desktop/uni/ISPW/PROGETTO/ISPW3.0%20copia/src/m

[StoricoOrdiniFacade.java]
(file:///Users/danieledimeo/Desktop/uni/ISPW/PROGETTO/ISPW3.0%20copia/src/main/java/org/example/use
[UsaVoucherFacade.java]
(file:///Users/danieledimeo/Desktop/uni/ISPW/PROGETTO/ISPW3.0%20copia/src/main/java/org/example/use
[VisualizzaOrdiniFacade.java]
(file:///Users/danieledimeo/Desktop/uni/ISPW/PROGETTO/ISPW3.0%20copia/src/main/java/org/example/use
[CreaOrdineController.java]
(file:///Users/danieledimeo/Desktop/uni/ISPW/PROGETTO/ISPW3.0%20copia/src/main/java/org/example/use
[CreaOrdineGUIController.java]
(file:///Users/danieledimeo/Desktop/uni/ISPW/PROGETTO/ISPW3.0%20copia/src/main/java/org/example/use