Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №7
ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Виконав
студент групи ІА–13:
Лапушенко А.К.

Київ 2023

Тема: *Шаблон фасад “Facade”.*
Варіант: POS(Point-of-sales)-system.

Хід роботи

Шаблон фасад “Facade” передбачає створення єдиного уніфікованого способу доступу до підсистеми без розкриття внутрішніх деталей підсистеми. Оскільки підсистема може складатися з безлічі класів, а кількість її функцій - не більше десяти, то щоб уникнути утворення «спагеті-коду» (коли все тісно пов'язано між собою) виділяють один загальний інтерфейс доступу, здатний правильним чином звертатися до внутрішніх деталей.
В даній роботі шаблон “Facade” був використаний для відокремлення бізнес-логіки.

Клієнт

```
public class OrderController {
    private final OrderService orderService;
    private final MenuItemService menuService;
    private static final String REDIRECT = "redirect:/orders";

    @GetMapping(value = "/")
    public String getOrders() { return REDIRECT; }

    @GetMapping(value = "")
    public ModelAndView getListOrders(){
        ModelAndView result = new ModelAndView( viewName: "orders/ordersList");
        List<Order> orders = orderService.listAll();
        result.addObject( attributeName: "orders", orders);
        return result;
    }

    @GetMapping(value = "/{id}")
    public ModelAndView getOrder(@PathVariable Long id){
        ModelAndView result = new ModelAndView( viewName: "orders/order");
        Order order = orderService.getById(id);
        result.addObject( attributeName: "order", order);
        result.addObject( attributeName: "menuItems", menuService.listAll());
        return result;
    }

    @PostMapping(value = "/{id}/pay")
    public String payOrder(@PathVariable Long id,  @RequestParam(name = "strategy") String strategy){
        Order order = orderService.getById(id);
        if(strategy.equals("card")){
            System.out.println();
            orderService.pay(new PayCard(), order);
        }else {orderService.pay(new PayCash(), order);}
        return "redirect:/orders/"+id;
```

```
@PostMapping(value = "/{id}/messages")
public String message(@PathVariable Long id){
    orderService.sendMessages(id);
    return "redirect:/orders/"+id;
}

@PostMapping(value = "/{id}/bill")
public String bill(@RequestParam(name = "type") String sendBillType, @PathVariable Long id){
    orderService.sendBill(SendBillType.valueOf(sendBillType.toUpperCase(Locale.ROOT)), id);
    return "redirect:/orders/"+id;
}
```

Фасад

```java
public class OrderService {
    private final OrderRepository orderRepository;

    private final UserService userService;
    private final Map<SendBillType, BillSender> map;

    public OrderService(OrderRepository orderRepository, UserService userService, List<BillSender> senderList) {...}

    public List<Order> listAll() {
        return orderRepository.findAll();
    }

    public double getFullCostOrder(Long id) {
        Order order = getById(id);
        return order.getFullCost();
    }

    public Order getById(Long id) {
        Order order = orderRepository.findById(id).orElseThrow();
        double sum = 0;
        for (OrderMenuItems o : order.getOrderMenuItems()) {
            sum += o.getTotalPrice();
        }
        order.setBill(sum);
        return order;
    }

    public void pay(PayStrategy strategy, Order order) {
        if (strategy.pay(order.getBill())) {
            order.setPayed(true);
            orderRepository.save(order);
        }
    }
```

```java
public void sendBill(SendBillType sendBillType, Long id) {
    BillSender billSender = map.get(sendBillType);
    Set<OrderMenuItems> orderMenuItems = orderRepository.findById(id).get().getOrderMenuItems();
    String bill = BillFormer.form(orderMenuItems);
    billSender.send(bill);
}

public void sendMessages(Long id){
    Set<OrderMenuItems> orderMenuItems = orderRepository.findById(id).get().getOrderMenuItems();
    for (OrderMenuItems o: orderMenuItems) {
        MenuItemFacade.sendMessagesToProcess(o);
    }
}
```

Метод відокремлений від логіки

```
public void sendMessages(Long id){
    Set<OrderMenuItems> orderMenuItems = orderRepository.findById(id).get().getOrderMenuItems();
    for (OrderMenuItems o: orderMenuItems) {
        MenuItemFacade.sendMessagesToProcess(o);
    }
}
```

Фасад

```
public class MenuItemFacade {
    final static BarFactory barFactory = new BarFactory();

    public static void sendMessagesToProcess(OrderMenuItems item) {
        Bar bar = barFactory.getBar(item.getMenuItem());
        bar.showMessage(item);
    }
}
```

Висновок

В цій роботі я навчився користуватись та використав паттерн “Facade” у своєму застосунку, який сприяє створенню уніфікованого доступу до підсистеми без розкриття внутрішніх деталей підсистеми.

Контрольні питання

1. *Розкажіть про KISS, DRY, YAGNI, закон парето.*

KISS (keep it simple, stupid — «нехай буде просто, дурню») – процес і принцип проектування, при якому простота системи декларується як основна мета та/або цінність.
DRY (don't repeat yourself – “не повторюйся”) – принцип розробки програмного забезпечення, що направлений на уникнення дублювання інформації будь-якого вигляду.
YAGNI (you aren't going to need it – “вам це не знадобиться”) – процес і принцип проєктування, при якому основною метою та цінністю є відмова від додавання функціональності, в якій немає безпосередньої потреби.
Закон Парето – емпіричне правило, яке стверджує, що для багатьох явищ 80 відсотків наслідків спричинені 20 відсотками причин.

2. *Навіщо використовується шаблон «медіатор» ?*

Використовується для визначення взаємодії об'єктів за допомогою іншого об'єкта (замість зберігання посилань один на одного). Даний шаблон схожий на шаблон «команда», проте в даному випадку замість зберігання даних про конкретну дію, зберігаються дані про взаємодії між компонентами.

3. *Чим відрізняється шаблон «шаблонний метод» від «фабричного методу» ?*

Область його використання абсолютно інша - для покрокового визначення конкретного алгоритму; більш того, даний шаблон не обов'язково створює нові об'єкти - лише визначає послідовність дій.

4. *Яку функціональність додає шаблон «міст»?*

Дозволяє будувати платформо-незалежні програми; Приховує зайві або небезпечні деталі реалізації від клієнтського коду; Реалізує принцип відкритості / закритості.

5. *Навіщо застосовується шаблон «фасад»?*

Для створення єдиного уніфікованого способу доступу до підсистеми без розкриття внутрішніх деталей підсистеми.