Phát hiện vi phạm nguyên lý SRP và OCP trong codebase

I. SRP

- **PaymentController:** phương thức làm trống giỏ hàng nên được tách ra vào class liên quan tới xử lý giỏ hàng để class này tập trung vào chức năng thanh toán.

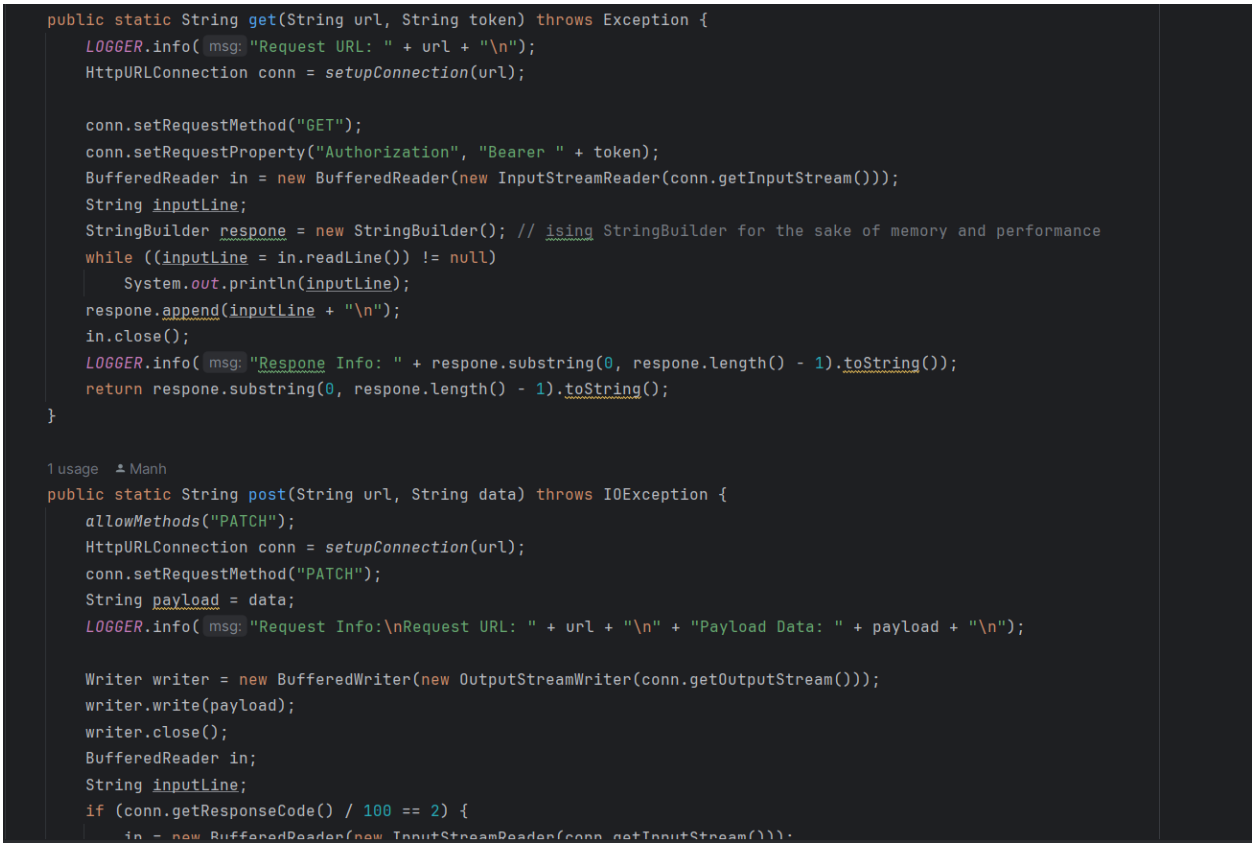
```
public Map<String, String> payOrder(int amount, String contents, String cardNumber, String cardHolderName,
        String expirationDate, String securityCode) {
    Map<String, String> result = new Hashtable<~>();
    result.put("RESULT", "PAYMENT FAILED!");
    try {
        this.card = new CreditCard(
                cardNumber,
                cardHolderName,
                getExpirationDate(expirationDate),
                Integer.parseInt(securityCode));

        this.interbank = new InterbankSubsystem();
        PaymentTransaction transaction = interbank.payOrder(card, amount, contents);

        result.put("RESULT", "PAYMENT SUCCESSFUL!");
        result.put("MESSAGE", "You have successfully paid the order!");
    } catch (PaymentException | UnrecognizedException ex) {
        result.put("MESSAGE", ex.getMessage());
    }
    return result;
}

1 usage  Manh
public void emptyCart() { SessionInformation.cartInstance.emptyCart(); }
```

- **PlaceOrderControler:** Class đang thực hiện nhiều trách nhiệm khác nhau: xử lý đặt hàng, xử lý thông tin giao hàng, tạo hoá đơn, xác thực dữ liệu. Cần tách ra thành các class riêng

```java
public DeliveryInfo processDeliveryInfo(HashMap info) throws InterruptedException, IOException, InvalidDeliveryInfoException {
    LOGGER.info( msg: "Process Delivery Info");
    LOGGER.info(info.toString());
    validateDeliveryInfo(info);
    DeliveryInfo deliveryInfo = new DeliveryInfo(
            String.valueOf(info.get("name")),
            String.valueOf(info.get("phone")),
            String.valueOf(info.get("province")),
            String.valueOf(info.get("address")),
            String.valueOf(info.get("instructions")),
            new DistanceCalculator());
    System.out.println(deliveryInfo.getProvince());
    return deliveryInfo;
}

/**
* The method validates the info
* @param info
* @throws InterruptedException
* @throws IOException
*/
1 usage  Manh
public void validateDeliveryInfo(HashMap<String, String> info) throws InterruptedException, IOException, InvalidDeliveryInfoExcep
    if (validatePhoneNumber(info.get("phone"))
    || validateName(info.get("name"))
    || validateAddress(info.get("address"))) return;
    else throw new InvalidDeliveryInfoException();
}
```

```java
1 usage  Manh
public boolean validatePhoneNumber(String phoneNumber) {
    if (phoneNumber.length() != 10) return false;
    if (!phoneNumber.startsWith("0")) return false;
    try {
        Integer.parseInt(phoneNumber);
    } catch (NumberFormatException e) {
        return false;
    }
    return true;
}

1 usage  Manh
public boolean validateName(String name) {
    if (Objects.isNull(name)) return false;
    String patternString = "^[a-zA-Z\\s]*$";
    Pattern pattern = Pattern.compile(patternString);
    Matcher matcher = pattern.matcher(name);
    return matcher.matches();
}

1 usage  Manh
public boolean validateAddress(String address) {
    if (Objects.isNull(address)) return false;
    String patternString = "^[a-zA-Z\\s]*$";
    Pattern pattern = Pattern.compile(patternString);
    Matcher matcher = pattern.matcher(address);
    return matcher.matches();
}
```

– **ApplicationProgrammingInterface:** Class đang thực hiện nhiều nhiệm vụ khác nhau như thực hiện các yêu cầu HTTP Get và Patch, thiết lập kết nối HTTP, cho phép các phương thức HTTP được chỉ định. Cần tách ra vào các class khác nhau.



```
public static String get(String url, String token) throws Exception {
    LOGGER.info( msg: "Request URL: " + url + "\n");
    HttpURLConnection conn = setupConnection(url);

    conn.setRequestMethod("GET");
    conn.setRequestProperty("Authorization", "Bearer " + token);
    BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
    String inputLine;
    StringBuilder respone = new StringBuilder(); // ising StringBuilder for the sake of memory and performance
    while ((inputLine = in.readLine()) != null)
        System.out.println(inputLine);
    respone.append(inputLine + "\n");
    in.close();
    LOGGER.info( msg: "Respone Info: " + respone.substring(0, respone.length() - 1).toString());
    return respone.substring(0, respone.length() - 1).toString();
}

1 usage  Manh
public static String post(String url, String data) throws IOException {
    allowMethods("PATCH");
    HttpURLConnection conn = setupConnection(url);
    conn.setRequestMethod("PATCH");
    String payload = data;
    LOGGER.info( msg: "Request Info:\nRequest URL: " + url + "\n" + "Payload Data: " + payload + "\n");

    Writer writer = new BufferedWriter(new OutputStreamWriter(conn.getOutputStream()));
    writer.write(payload);
    writer.close();
    BufferedReader in;
    String inputLine;
    if (conn.getResponseCode() / 100 == 2) {
        in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
```

– **MyMap:** Class đang thực hiện các chức năng với các mục đích khác nhau như chuyển đổi Object sang Map, chuyển đổi JSON String sang Map, chuyển đổi MyMap sang JSON. Có thể tách thành các class riêng biệt.

```
2 usages  Manh
public String toJSON() {
    int max = size() - 1;
    if (max == -1)
        return "{}";

    StringBuilder sb = new StringBuilder();
    Iterator<Map.Entry<String, Object>> it = entrySet().iterator();

    sb.append('{');
    for (int i = 0;; i++) {
        Map.Entry<String, Object> e = it.next();
        String key = e.getKey();
        Object value = e.getValue();
        sb.append('"' + key.toString() + '"');
        sb.append(':');
        sb.append(value instanceof MyMap ? ((MyMap) value).toJSON() : ('"' + value.toString() + '"'));

//      if (value instanceof MyMap) {
//          sb.append(((MyMap) value).toJSON());
//      } else {
//          sb.append('"' + value.toString() + '"');
//      }
        if (i == max)
            return sb.append('}').toString();
        sb.append(",");
    }
}
```

```
2 usages  Manh
public static Map<String, Object> toMyMap(Object obj) throws IllegalArgumentException, IllegalAccessException {
    Map<String, Object> map = new MyMap();
    List<Field> fields = new ArrayList<>();
    fields.addAll(Arrays.asList(obj.getClass().getDeclaredFields()));
    fields.addAll(Arrays.asList(obj.getClass().getSuperclass().getDeclaredFields()));

    for (Field field : fields) {
        field.setAccessible(true);
        Object value = field.get(obj);
        try {
            if (!value.getClass().getPackage().getName().startsWith("java")) {
                value = MyMap.toMyMap(value).toString();
            }
        } catch (Exception ex) {
            ;
        }
        map.put(field.getName(), value);
        field.setAccessible(false);
    }
    return map;
}
```

```java
2 usages  Manh
public static MyMap toMyMap(String str, int idx) throws IllegalArgumentException {
    if (str == null || str.length() < 2 || str.charAt(idx) != '{') {
        throw new IllegalArgumentException("Cannot resolve the input.");
    } else if (idx >= str.length()) {
        return null;
    }

    MyMap root = new MyMap();
    StringBuilder sb = new StringBuilder();
    int i = idx;
    sb.append(str.charAt(i));

    i++;
    try {
        while (true) {
            // open quote
            if (str.charAt(i) != '"') {
                throw new IllegalArgumentException("Cannot resolve the input.");
            }
            // get key
            String key;
            try {
                key = getNextTerm(str, i);
            } catch (Exception ex) {
                throw new IllegalArgumentException("Cannot resolve the input.");
            }
            if (key == null) {
                throw new IllegalArgumentException("Cannot resolve the input.");
```

II. OCP

- **PaymentController:** Khi có thêm phương thức thanh toán mới sẽ cần phải sửa đổi class do mới chỉ đang xử lý với CreditCard

```java
public Map<String, String> payOrder(int amount, String contents, String cardNumber, String cardHolderName,
        String expirationDate, String securityCode) {
    Map<String, String> result = new Hashtable<~>();
    result.put("RESULT", "PAYMENT FAILED!");
    try {
        this.card = new CreditCard(
                cardNumber,
                cardHolderName,
                getExpirationDate(expirationDate),
                Integer.parseInt(securityCode));

        this.interbank = new InterbankSubsystem();
        PaymentTransaction transaction = interbank.payOrder(card, amount, contents);

        result.put("RESULT", "PAYMENT SUCCESSFUL!");
        result.put("MESSAGE", "You have successfully paid the order!");
    } catch (PaymentException | UnrecognizedException ex) {
        result.put("MESSAGE", ex.getMessage());
    }
    return result;
}
```

– **PlaceOrderController:** Khi sử dụng thư viện mới thay cho DistanceCalculator sẽ phải sửa code

```
1 usage  Manh
public DeliveryInfo processDeliveryInfo(HashMap info) throws InterruptedException, IOException, InvalidDeliveryInfoException {
    LOGGER.info( msg: "Process Delivery Info");
    LOGGER.info(info.toString());
    validateDeliveryInfo(info);
    DeliveryInfo deliveryInfo = new DeliveryInfo(
            String.valueOf(info.get("name")),
            String.valueOf(info.get("phone")),
            String.valueOf(info.get("province")),
            String.valueOf(info.get("address")),
            String.valueOf(info.get("instructions")),
            new DistanceCalculator());
    System.out.println(deliveryInfo.getProvince());
    return deliveryInfo;
}
```

– **PaymentTransaction:** khi có phương thức thanh toán mới (ví dụ Domestic Card) sẽ phải sửa code do đang xử lý cố định với CreditCard

```
1 usage  Manh
public PaymentTransaction(String errorCode, CreditCard card, String transactionId, String transactionContent,
                          int amount, String createdAt) {
    super();
    this.errorCode = errorCode;
    this.card = card;
    this.transactionId = transactionId;
    this.transactionContent = transactionContent;
    this.amount = amount;
    this.createdAt = createdAt;
}
```

– **DeliveryInfo:** Khi sử dụng thư viện tính toán khoảng cách khác thay vì DistanceCalculator sẽ phải sửa code

```
public DeliveryInfo(String name, String phone, String province, String address, String shippingInstructions, DistanceCalculator dista
    this.name = name;
    this.phone = phone;
    this.province = province;
    this.address = address;
    this.shippingInstructions = shippingInstructions;
    this.distanceCalculator = distanceCalculator;
}

1 usage  Manh
public int calculateShippingFee(Order order) {
    int distance = distanceCalculator.calculateDistance(address, province);
    return (int) (distance * 1.2);
}
```

– **InterbankPayloadConverter:** Khi có phương thức thanh toán mới sẽ phải sửa code do đang xử lý cố định với CreditCard

```java
String convertToRequestPayload(CreditCard card, int amount, String contents) {
    Map<String, Object> transaction = new MyMap();

    try {
        transaction.putAll(MyMap.toMyMap(card));
    } catch (IllegalArgumentException | IllegalAccessException e) {
        // TODO Auto-generated catch block
        throw new InvalidCardException();
    }
    transaction.put("command", InterbankConfigs.PAY_COMMAND);
    transaction.put("transactionContent", contents);
    transaction.put("amount", amount);
    transaction.put("createdAt", getToday());

    Map<String, Object> requestMap = new MyMap();
    requestMap.put("version", InterbankConfigs.VERSION);
    requestMap.put("transaction", transaction);

    return ((MyMap) requestMap).toJSON();
}
```

- **InterbankSubsystemController:** Khi có phương thức thanh toán mới sẽ phải sửa code do đang xử lý cố định với CreditCard

```java
1 usage  Manh
public PaymentTransaction refund(CreditCard card, int amount, String contents) { return null; }

1 usage  Manh  1 related problem
public PaymentTransaction payOrder(CreditCard card, int amount, String contents) {
    String requestPayload = interbankPayloadConverter.convertToRequestPayload(card, amount, contents);
    String responseText = interbankBoundary.query(InterbankConfigs.PROCESS_TRANSACTION_URL, requestPayload);
    return interbankPayloadConverter.extractPaymentTransaction(responseText);
}
```

- **InterbankInterface:** Khi có phương thức thanh toán mới sẽ phải sửa code do đang xử lý cố định với CreditCard

```
2 usages  1 implementation  Manh
public abstract PaymentTransaction payOrder(CreditCard card, int amount, String contents)
        throws PaymentException, UnrecognizedException;

/**
 * Refund, and then return the payment transaction
 *
 * @param card     - the credit card which would be refunded to
 * @param amount   - the amount to refund
 * @param contents - the transaction contents
 * @return {@link PaymentTransaction PaymentTransaction} - if the
 *         payment is successful
 * @throws PaymentException      if responded with a pre-defined error code
 * @throws UnrecognizedException if responded with an unknown error code or
 *                               something goes wrong
 */
1 usage  1 implementation  Manh
public abstract PaymentTransaction refund(CreditCard card, int amount, String contents)
        throws PaymentException, UnrecognizedException;
```

– **InterbankSubsystem:** Khi có phương thức thanh toán mới sẽ phải sửa code do đang xử lý cố định với CreditCard

```
2 usages  Manh  1 related problem
public PaymentTransaction payOrder(CreditCard card, int amount, String contents) {
    PaymentTransaction transaction = ctrl.payOrder(card, amount, contents);
    return transaction;
}

/**
 * @see InterbankInterface#refund(CreditCard, int,
 *      String)
 */
1 usage  Manh
public PaymentTransaction refund(CreditCard card, int amount, String contents) {
    PaymentTransaction transaction = ctrl.refund(card, amount, contents);
    return transaction;
}
```

– **CartScreenHandler, HomeScreenHandler, LoginScreenHandler, IntroScreenHandler, InvoiceScreenHandler, PaymentScreenHandler, ResultScreenHandler, ShippingScreenHandler:**

Khi muốn thay đổi cách hiển thị thông báo lỗi và nội dung của thông báo lỗi sẽ phải sửa code do hiện đang xử lý hiển thị thông báo lỗi trên màn hình Popup và nội dung thông báo là cố định nếu xảy ra lỗi IOException

```
super(stage, screenPath);
try {
    setupData(invoice);
    setupFunctionality();
} catch (IOException ex) {
    LOGGER.info(ex.getMessage());
    PopupScreen.error("Error when loading resources.");
} catch (Exception ex) {
    LOGGER.info(ex.getMessage());
    PopupScreen.error(ex.getMessage());
}
```