

Assignment 10

Programming

1.1. CHUẨN BỊ

- Người học cần cài đặt sẵn Eclipse và môi trường Java11 trên máy cá nhân
- Clone project từ: <https://github.com/leminhnguyen/AIMS-Student>

1.2. NỘI DUNG CHI TIẾT

1.2.1.1. Bắt đầu với code sample

- Trước khi đến với bài học các bạn hãy xem qua một class có tên **API.java** khoảng 15-20p và hãy suy ngẫm các câu hỏi:
 - Class này mục đích là gì ?
 - Class này do ai viết ?
 - Sử dụng class này như thế nào ?

```
MyClass.java App.java PlaceOrderController.java AIMSDB.java API.java ✖
1 package utils;
2
3 import java.io.BufferedReader;
4 import java.io.BufferedWriter;
5 import java.io.IOException;
6 import java.io.InputStreamReader;
7 import java.io.OutputStreamWriter;
8 import java.io.Writer;
9 import java.lang.reflect.Field;
10 import java.lang.reflect.Modifier;
11 import java.net.HttpURLConnection;
12 import java.net.URL;
13 import java.text.DateFormat;
14 import java.text.SimpleDateFormat;
15 import java.util.Arrays;
16 import java.util.LinkedHashSet;
17 import java.util.Set;
18 import java.util.logging.Logger;
19
20 import entity.payment.CreditCard;
21 import entity.payment.PaymentTransaction;
22
23 public class API {
24
25     public static DateFormat DATE_FORMATER = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
26     private static Logger LOGGER = Utils.getLogger(Utils.class.getName());
27
28     public static String get(String url, String token) throws Exception {
29         LOGGER.info("Request URL: " + url + "\n");
30         URL line_api_url = new URL(url);
31         HttpURLConnection conn = (HttpURLConnection) line_api_url.openConnection();
32         conn.setDoInput(true);
33         conn.setDoOutput(true);
34         conn.setRequestMethod("GET");
35         conn.setRequestProperty("Content-Type", "application/json");
36         conn.setRequestProperty("Authorization", "Bearer " + token);
37         BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
38         String inputLine;
39         StringBuilder response = new StringBuilder(); // using StringBuilder for the sake of memory and performance
40         while ((inputLine = in.readLine()) != null)
41             System.out.println(inputLine);
42         response.append(inputLine + "\n");
43         in.close();
44         LOGGER.info("Response Info: " + response.substring(0, response.length() - 1).toString());
45         return response.substring(0, response.length() - 1).toString();
46     }
47 }
```

```

47
48
49 public static String post(String url, String data) throws IOException {
50     allowMethods("PATCH");
51     URL line_api_url = new URL(url);
52     String payload = data;
53     LOGGER.info("Request Info:\nRequest URL: " + url + "\n" + "Payload Data: " + payload + "\n");
54     HttpURLConnection conn = (HttpURLConnection) line_api_url.openConnection();
55     conn.setDoInput(true);
56     conn.setDoOutput(true);
57     conn.setRequestMethod("PATCH");
58     conn.setRequestProperty("Content-Type", "application/json");
59     Writer writer = new BufferedWriter(new OutputStreamWriter(conn.getOutputStream()));
60     writer.write(payload);
61     writer.close();
62     BufferedReader in;
63     String inputLine;
64     if (conn.getResponseCode() / 100 == 2) {
65         in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
66     } else {
67         in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
68     }
69     StringBuilder response = new StringBuilder();
70     while ((inputLine = in.readLine()) != null)
71         response.append(inputLine);
72     in.close();
73     LOGGER.info("Response Info: " + response.toString());
74     return response.toString();
75 }
76
77 private static void allowMethods(String... methods) {
78     try {
79         Field methodsField = HttpURLConnection.class.getDeclaredField("methods");
80         methodsField.setAccessible(true);
81
82         Field modifiersField = Field.class.getDeclaredField("modifiers");
83         modifiersField.setAccessible(true);
84         modifiersField.setInt(methodsField, methodsField.getModifiers() & ~Modifier.FINAL);
85
86         String[] oldMethods = (String[]) methodsField.get(null);
87         Set<String> methodsSet = new LinkedHashSet<>(Arrays.asList(oldMethods));
88         methodsSet.addAll(Arrays.asList(methods));
89         String[] newMethods = methodsSet.toArray(new String[0]);
90
91         methodsField.set(null/* static field */, newMethods);
92     } catch (NoSuchFieldException | IllegalAccessException e) {
93         throw new IllegalStateException(e);
94     }
95 }
96
97 }

```

- Nếu như sau khi xem xong và bạn đảm bảo hiểu hết những gì viết trong class thì chắc hẳn bạn phải là một lập trình viên Java nhiều kinh nghiệm
- Tuy nhiên đa phần các bạn sẽ bị choáng ngợp và cảm thấy khó hiểu ở class khoảng gần 100 dòng này. Bạn sẽ không hiểu class này mục đích là gì, dùng như thế nào và ai viết
- Không những class trên gây khó hiểu mà nó đang còn tồn đọng nhiều vấn đề như tái cấu trúc, tối ưu code,..
- Vậy bài học hôm nay sẽ giúp các bạn giải quyết từng vấn đề và bạn có thể áp dụng vào trong Project của mình

1.2.1.2. Làm quen với Javadoc

- Chắc hẳn trong chúng ta thì ai cũng đã nghe tới khái niệm như thêm comment hoặc documentation cho các class hoặc method mình viết

- Documentation là gì? Documentation đơn giản là các đoạn text được thêm vào trong mã nguồn của dự án phần mềm với mục đích giải thích những cái bạn đang làm, thực hiện như thế nào và làm thế nào để sử dụng nó.
- Việc tạo thói quen thêm comment và documentation vào code thì có vai trò rất quan trọng, nó sẽ giúp cho người khác hiểu code của bạn, và cũng có thể là chính bạn sau này khi đọc lại code của mình. Hơn thế nữa việc thêm documentation vào code sẽ chứng tỏ bạn là một lập trình viên chuyên nghiệp.
- JAVADOC là documentation cho ngôn ngữ Java, mục đích chính của nó cũng là giúp cho bạn giải thích những đoạn code được viết. Việc thêm JAVADOC có thể thực hiện thủ công bằng cách gõ từng ký tự nhưng cũng có rất nhiều công cụ IDE hỗ trợ bạn tạo doc tự động như: Eclipse, IntelliJ, VSCode,...
- Cú pháp của JAVADOC: các documentation của Java được đặt ở trong cặp `/** */` và có thể thêm nhiều dòng trong giữa cặp dấu.
- JAVADOC thường được mô tả bởi các annotation (bắt đầu bởi `@`), một vài loại annotation phổ biến trong JAVADOC
 - **@author**: chỉ tên tác giả của đoạn code hoặc có thể là người đóng góp nhiều nhất. Thường được áp dụng cho các level: class hoặc package
 - **@param**: Mô tả tham số truyền vào một phương thức hoặc constructor
 - **@return**: mô tả giá trị trả về của một class hoặc phương thức
 - **@since**: phiên bản mà thuộc tính được thêm vào
 - **@throws**: loại exception mà phương thức có thể tung ra
 - **@deprecated**: chỉ cho người khác biết là phương thức hoặc class này không còn được sử dụng nữa
 - **{@link}**: tạo liên kết tới những phương thức hoặc phần nội dung khác
- Người học có thể tham khảo chi tiết thêm những loại annotation Java cung cấp ở link sau: <https://idratherbewriting.com/java-javadoc-tags/>

1.2.1.3. Thực hành tạo Javadoc với Eclipse

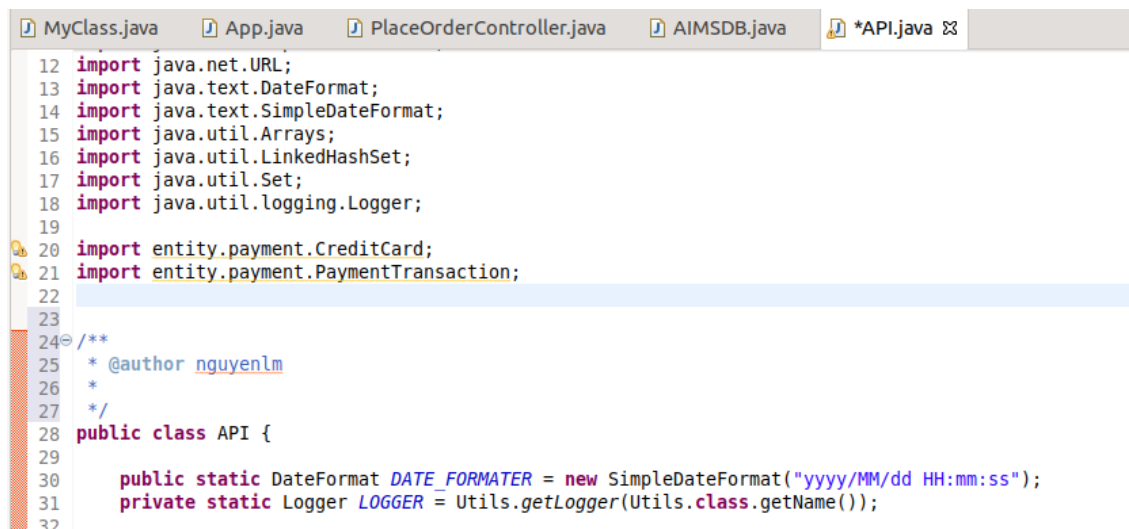
a) Import project vào Eclipse

- Mở Eclipse và import Project đã clone về máy, hoặc nếu bạn đã clone thì sử dụng git pull để lấy code mới nhất về nhánh master
- Mở class `src/Utils/API.java`
- Tiếp theo chúng ta sẽ tiến hành thêm javadoc cho class và method ở trong class `API.java` này

b) Thực hành thêm doc cho API class

- Để thêm doc vào trong Java code thông qua Eclipse thì chúng ta có 3 cách:
 - + Gõ thủ công

- + Dùng thanh công cụ
- + Dùng các shortcuts
- Để thêm javadoc cho class, method hoặc attribute thông qua thanh công cụ thì ta sẽ click con trỏ chuột lên ngay bên trên class, method hoặc attribute đó. Sau đó click chuột phải -> Source -> Generate Element Comment
- Cách nhanh nhất để thêm java doc chính là thông qua shortcut. Để thêm javadoc thì ta click chuột lên phía trên của class, method hoặc attribute sau đó gõ `/**` và ấn Enter, sau đó Eclipse sẽ tạo Javadoc tự động cho chúng ta
- Ví dụ với class API

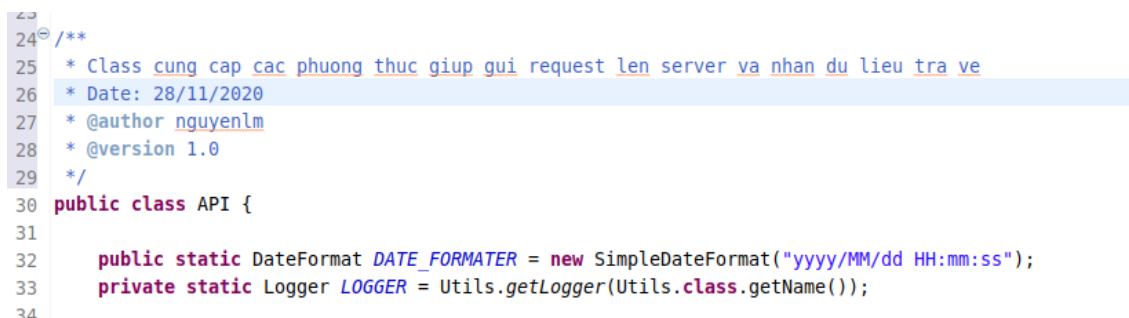


```

12 import java.net.URL;
13 import java.text.DateFormat;
14 import java.text.SimpleDateFormat;
15 import java.util.Arrays;
16 import java.util.LinkedHashSet;
17 import java.util.Set;
18 import java.util.logging.Logger;
19
20 import entity.payment.CreditCard;
21 import entity.payment.PaymentTransaction;
22
23
24 /**
25  * @author nguyenlm
26  *
27  */
28 public class API {
29
30     public static DateFormat DATE_FORMATER = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
31     private static Logger LOGGER = Utils.getLogger(Utils.class.getName());
32

```

- Việc thêm javadoc ở trên mới cho class API chỉ là bước đầu và khá sơ khai, vậy nên chúng ta cần bổ sung thêm các thông tin khác như: mô tả, @version, ngày viết chương trình
- Việc thêm mô tả cho một class, method hoặc attribute nên để lên đầu của documentation. Sau khi thêm các thông tin ta sẽ có thông tin documentation như sau:



```

24 /**
25  * Class cung cap cac phuong thuc giup gui request len server va nhan du lieu tra ve
26  * Date: 28/11/2020
27  * @author nguyenlm
28  * @version 1.0
29  */
30 public class API {
31
32     public static DateFormat DATE_FORMATER = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
33     private static Logger LOGGER = Utils.getLogger(Utils.class.getName());
34

```

- Sau khi thêm documentation cho class API thì chúng ta đã có một cái nhìn tổng quan về mục đích của class này. Tiếp theo chúng ta sẽ tiếp tục thêm documentation cho các attribute và method

+ **Attributes**

```

30 public class API {
31
32     /**
33      * Thuoc tinh giúp format ngày tháng theo định dạng
34      */
35     public static DateFormat DATE_FORMATER = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
36
37     /**
38      * Thuoc tinh giúp log ra thông tin ra console
39      */
40     private static Logger LOGGER = Utils.getLogger(Utils.class.getName());

```

+ **get method**

```

42     /**
43      * Phương thức giúp gọi các api dạng GET
44      * @param url: đường dẫn tới server cần request
45      * @param token: đoạn mã bí mật cần cung cấp để xác thực người dùng
46      * @return response: phản hồi từ server (dạng string)
47      * @throws Exception
48      */
49     public static String get(String url, String token) throws Exception {
50         LOGGER.info("Request URL: " + url + "\n");
51         URL line_api_url = new URL(url);
52         HttpURLConnection conn = (HttpURLConnection) line_api_url.openConnection();
53         conn.setDoInput(true);
54         conn.setDoOutput(true);
55         conn.setRequestMethod("GET");
56         conn.setRequestProperty("Content-Type", "application/json");
57         conn.setRequestProperty("Authorization", "Bearer " + token);
58         BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
59         String inputLine;
60         StringBuilder response = new StringBuilder(); // using StringBuilder for the sake of memory and performance
61         while ((inputLine = in.readLine()) != null)
62             System.out.println(inputLine);
63         response.append(inputLine + "\n");
64         in.close();
65         LOGGER.info("Response Info: " + response.substring(0, response.length() - 1).toString());
66         return response.substring(0, response.length() - 1).toString();
67     }
68

```

+ **post method**

```

70  /**
71   * Phương thức giúp gọi các API dạng POST (thanh toán,..)
72   * @param url: đường dẫn tới server cần request
73   * @param data: dữ liệu đưa lên server để xử lý (dạng JSON)
74   * @return response: phản hồi từ server (dạng string)
75   * @throws IOException
76   */
77  public static String post(String url, String data) throws IOException {
78      allowMethods("PATCH");
79      URL line_api_url = new URL(url);
80      String payload = data;
81      LOGGER.info("Request Info:\nRequest URL: " + url + "\n" + "Payload Data: " + payload + "\n");
82      HttpURLConnection conn = (HttpURLConnection) line_api_url.openConnection();
83      conn.setDoInput(true);
84      conn.setDoOutput(true);
85      conn.setRequestMethod("PATCH");
86      conn.setRequestProperty("Content-Type", "application/json");
87      Writer writer = new BufferedWriter(new OutputStreamWriter(conn.getOutputStream()));
88      writer.write(payload);
89      writer.close();
90      BufferedReader in;
91      String inputLine;
92      if (conn.getResponseCode() / 100 == 2) {
93          in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
94      } else {
95          in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
96      }
97      StringBuilder response = new StringBuilder();
98      while ((inputLine = in.readLine()) != null)
99          response.append(inputLine);
100     in.close();
101     LOGGER.info("Response Info: " + response.toString());
102     return response.toString();
103 }

```

+ *allowMethods*

```

105  /**
106   * Phương thức cho phép gọi các loại giao thức API khác nhau như PATCH, PUT,.. (chỉ hoạt động với Java 11)
107   * @deprecated chỉ hoạt động với Java <= 11
108   * @param methods: giao thức cần cho phép [PATCH, PUT,..]
109   */
110  private static void allowMethods(String... methods) {
111      try {
112          Field methodsField = HttpURLConnection.class.getDeclaredField("methods");
113          methodsField.setAccessible(true);
114
115          Field modifiersField = Field.class.getDeclaredField("modifiers");
116          modifiersField.setAccessible(true);
117          modifiersField.setInt(modifiersField, methodsField.getModifiers() & ~Modifier.FINAL);
118
119          String[] oldMethods = (String[]) methodsField.get(null);
120          Set<String> methodsSet = new LinkedHashSet<>(Arrays.asList(oldMethods));
121          methodsSet.addAll(Arrays.asList(methods));
122          String[] newMethods = methodsSet.toArray(new String[0]);
123
124          methodsField.set(null/* static field */, newMethods);
125      } catch (NoSuchFieldException | IllegalAccessException e) {
126          throw new IllegalStateException(e);
127      }
128  }

```

c) Thực hành thêm comment cho các phương thức

- Để làm rõ hơn các câu lệnh trong từng method thì chúng ta có thể thêm comment (//) vào trong các method như sau

```

42  /**
43   * Phương thức giúp gọi các api dạng GET (lấy số dư tài khoản,..)
44   * @param url: đường dẫn tới server cần request
45   * @param token: đoạn mã bí mật cần cung cấp để xác thực người dùng
46   * @return response: phản hồi từ server (dạng string)
47   * @throws Exception
48   */
49  public static String get(String url, String token) throws Exception {
50
51      // phần 1: setup
52      LOGGER.info("Request URL: " + url + "\n");
53      URL line_api_url = new URL(url);
54      HttpURLConnection conn = (HttpURLConnection) line_api_url.openConnection();
55      conn.setDoInput(true);
56      conn.setDoOutput(true);
57      conn.setRequestMethod("GET");
58      conn.setRequestProperty("Content-Type", "application/json");
59      conn.setRequestProperty("Authorization", "Bearer " + token);
60
61      // phần 2: đọc dữ liệu trả về từ server
62      BufferedReader in;
63      String inputLine;
64      if (conn.getResponseCode() / 100 == 2) {
65          in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
66      } else {
67          in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
68      }
69      StringBuilder response = new StringBuilder(); // sử dụng String Builder cho việc tối ưu về mặt bộ nhớ
70      while ((inputLine = in.readLine()) != null)
71          System.out.println(inputLine);
72      response.append(inputLine + "\n");
73      in.close();
74      LOGGER.info("Response Info: " + response.substring(0, response.length() - 1).toString());
75      return response.substring(0, response.length() - 1).toString();
76  }
77

```

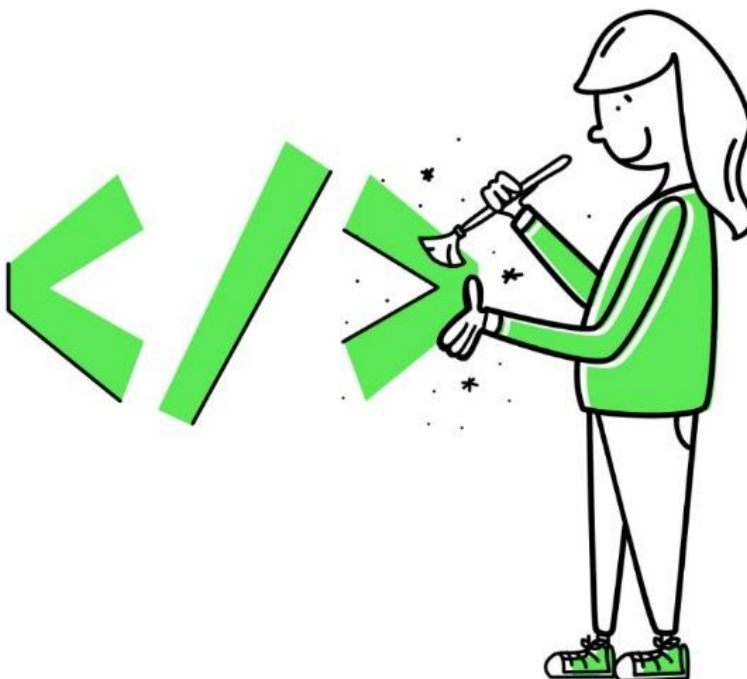


```

79  /**
80   * Phuong thuc giup goi cac api dang POST (thanh toan,...)
81   * @param url: duong dan toi server can request
82   * @param data: du lieu dua len server de xu ly (dang JSON)
83   * @return response: phan hoi tu server (dang string)
84   * @throws IOException
85   */
86  public static String post(String url, String data) throws IOException {
87      allowMethods("PATCH");
88
89      // phan 1: setup
90      URL line_api_url = new URL(url);
91      String payload = data;
92      LOGGER.info("Request Info:\nRequest URL: " + url + "\n" + "Payload Data: " + payload + "\n");
93      HttpURLConnection conn = (HttpURLConnection) line_api_url.openConnection();
94      conn.setDoInput(true);
95      conn.setDoOutput(true);
96      conn.setRequestMethod("PATCH");
97      conn.setRequestProperty("Content-Type", "application/json");
98
99      // phan 2: gui du lieu
100     Writer writer = new BufferedWriter(new OutputStreamWriter(conn.getOutputStream()));
101     writer.write(payload);
102     writer.close();
103
104     // phan 3: doc du lieu gui ve tu server
105     BufferedReader in;
106     String inputLine;
107     if (conn.getResponseCode() / 100 == 2) {
108         in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
109     } else {
110         in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
111     }
112     StringBuilder response = new StringBuilder(); // su dung String Builder cho viec toi uu ve mat bo nho
113     while ((inputLine = in.readLine()) != null)
114         response.append(inputLine);
115     in.close();
116     LOGGER.info("Response Info: " + response.toString());
117     return response.toString();
118 }

```

1.2.1.4. Làm quen với refactoring



a) Mục đích của refactoring

- Refactoring là việc thay đổi hoặc tái cấu trúc lại mã nguồn nhằm mục đích chuyển code rối rắm (mess code) về dạng code dễ hiểu hơn (clean code).
- Refactoring giúp cho việc bảo trì một cách dễ dàng, giảm bớt đi sự phức tạp rườm rà từ đó giúp chúng ta dễ quản lý mã nguồn và đưa được product của mình ra thị trường một cách nhanh chóng

b) Khi nào cần refactoring code

- Khi code trong project của mình bị lặp đi lặp lại nhiều lần
- Khi code quá rườm rà, dài dòng, khó hiểu và khó thêm thuộc tính mới
- Trong quá trình fix bug, khi chúng ta liên tục gặp phải bugs thì việc refactoring cho code cleaner hơn sẽ giúp chúng ta nhận diện ra những vấn đề trong code
- Trong quá trình review code trước khi đưa ra sản phẩm, đây có thể là cơ hội cuối cùng để tinh chỉnh code trước khi đưa ra sản phẩm

c) Một vài loại refactoring phổ biến

- **Extract method:** phương pháp này có nghĩa là nếu như bạn thấy đoạn code nào đó lặp đi lặp lại nhiều lần trong các phương thức khác thì hãy tách đoạn code đó ra thành một phương thức riêng
- **Extract class:** đây là cách làm tương tự như extract method nhưng ở mức độ class, những phương thức nào có liên quan tới nhau và hay được sử dụng thì chúng ta có thể tách ra một class khác và tái sử dụng bằng cách kế thừa hoặc kết tập

1.2.1.5. Thực hành refactoring code với class API

a) Nhận diện vấn đề

- Đây là thời điểm chúng ta sẽ quay lại với class API phía trên, nếu chúng ta để ý kỹ chúng ta sẽ thấy 2 phương thức get và post có nhiều phần chung được phân tách như sau:

```

42  /**
43   * Phương thức giúp gọi các api dạng GET (lay so du tai khoan,..)
44   * @param url: đường dẫn tới server cần request
45   * @param token: đoạn mã bí mật cần cung cấp để xác thực người dùng
46   * @return response: phản hồi từ server (dạng string)
47   * @throws Exception
48   */
49  public static String get(String url, String token) throws Exception {
50
51      // phần 1: setup
52      LOGGER.info("Request URL: " + url + "\n");
53      URL line_api_url = new URL(url);
54      HttpURLConnection conn = (HttpURLConnection) line_api_url.openConnection();
55      conn.setDoInput(true);
56      conn.setDoOutput(true);
57      conn.setRequestMethod("GET");
58      conn.setRequestProperty("Content-Type", "application/json");
59      conn.setRequestProperty("Authorization", "Bearer " + token);
60
61      // phần 2: đọc dữ liệu trả về từ server
62      BufferedReader in;
63      String inputLine;
64      if (conn.getResponseCode() / 100 == 2) {
65          in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
66      } else {
67          in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
68      }
69      StringBuilder response = new StringBuilder(); // sử dụng String Builder cho việc tối ưu về mặt bộ nhớ
70      while ((inputLine = in.readLine()) != null)
71          System.out.println(inputLine);
72      response.append(inputLine + "\n");
73      in.close();
74      LOGGER.info("Response Info: " + response.substring(0, response.length() - 1).toString());
75      return response.substring(0, response.length() - 1).toString();
76  }
77

```

```

79  /**
80   * Phương thức giúp gọi các api dạng POST (thanh toán,..)
81   * @param url: đường dẫn tới server cần request
82   * @param data: dữ liệu đưa lên server để xử lý (dạng JSON)
83   * @return response: phản hồi từ server (dạng string)
84   * @throws IOException
85   */
86  public static String post(String url, String data) throws IOException {
87      allowMethods("PATCH");
88
89      // phần 1: setup
90      URL line_api_url = new URL(url);
91      String payload = data;
92      LOGGER.info("Request Info:\nRequest URL: " + url + "\n" + "Payload Data: " + payload + "\n");
93      HttpURLConnection conn = (HttpURLConnection) line_api_url.openConnection();
94      conn.setDoInput(true);
95      conn.setDoOutput(true);
96      conn.setRequestMethod("PATCH");
97      conn.setRequestProperty("Content-Type", "application/json");
98
99      // phần 2: gửi dữ liệu
100     Writer writer = new BufferedWriter(new OutputStreamWriter(conn.getOutputStream()));
101     writer.write(payload);
102     writer.close();
103
104     // phần 3: đọc dữ liệu gửi về từ server
105     BufferedReader in;
106     String inputLine;
107     if (conn.getResponseCode() / 100 == 2) {
108         in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
109     } else {
110         in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
111     }
112     StringBuilder response = new StringBuilder(); // sử dụng String Builder cho việc tối ưu về mặt bộ nhớ
113     while ((inputLine = in.readLine()) != null)
114         response.append(inputLine);
115     in.close();
116     LOGGER.info("Response Info: " + response.toString());
117     return response.toString();
118 }

```

- Sau khi nhìn lại 2 methods và có sự phân cách giữa các dòng chúng ta sẽ thấy giữa 2 phương thức post và get này có các đoạn code chung như setup connection và đọc dữ liệu trả về từ server,
- Ta có thể tiến hành refactor đoạn mã nguồn này bằng cách chọn đoạn mã nguồn cần extract, ấn chuột phải, chọn Refactor → Extract method.

b) Thực hành refactoring

- Chúng ta sẽ tiến hành trích xuất 2 phương thức đặt tên là setupConnection và readResponse
- Code chúng ta sau khi refactoring sẽ như sau
 - **setupConnection()**

```

44  /**
45   * Thiet lap connection toi server
46   * @param url: duong dan toi server can request
47   * @param method: giao thuc api
48   * @param token: doan ma bam can cung cap de xac thuc nguoi dung
49   * @return connection
50   * @throws IOException
51   */
52  private static HttpURLConnection setupConnection(String url, String method, String token) throws IOException {
53      // phan 1: setup
54      LOGGER.info("Request URL: " + url + "\n");
55      URL line_api_url = new URL(url);
56      HttpURLConnection conn = (HttpURLConnection) line_api_url.openConnection();
57      conn.setDoInput(true);
58      conn.setDoOutput(true);
59      conn.setRequestMethod(method);
60      conn.setRequestProperty("Content-Type", "application/json");
61      conn.setRequestProperty("Authorization", "Bearer " + token);
62      return conn;
63  }
64

```

○ ReadResponse()

```

65  /**
66   * Phuong thuc doc du lieu tra ve tu server
67   * @param conn: connection to server
68   * @return response: phan hoi tra ve tu server
69   * @throws IOException
70   */
71  private static String readResponse(HttpURLConnection conn) throws IOException {
72      // phan 2: doc du lieu tra ve tu server
73      BufferedReader in;
74      String inputLine;
75      if (conn.getResponseCode() / 100 == 2) {
76          in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
77      } else {
78          in = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
79      }
80      StringBuilder response = new StringBuilder(); // su dung String Builder cho viec toi uu ve mat bo nho
81      while ((inputLine = in.readLine()) != null)
82          System.out.println(inputLine);
83      response.append(inputLine + "\n");
84      in.close();
85      LOGGER.info("Response Info: " + response.substring(0, response.length() - 1).toString());
86      return response.substring(0, response.length() - 1).toString();
87  }
88

```

- Khi đó 2 phương thức get và post sẽ còn lại như sau

```

89  /**
90   * Phuong thuc giup gọi các api dạng GET (lay so du tai khoan,..)
91   * @param url: duong dan toi server can request
92   * @param token: doan ma bam can cung cap de xac thuc nguoi dung
93   * @return response: phan hoi tu server (dang string)
94   * @throws Exception
95   */
96  public static String get(String url, String token) throws Exception {
97
98      // phan 1: setup
99      HttpURLConnection conn = setupConnection(url, "GET", token);
100
101      // phan 2: doc du lieu tra ve tu server
102      String response = readResponse(conn);
103
104      return response;
105  }
106

```

```
108  /**
109   * Phuong thuc giup goi cac api dang POST (thanh toan,..)
110   * @param url: duong dan toi server can request
111   * @param token: doan ma bam can cung cap de xac thuc nguoi dung
112   * @param data: du lieu dua len server de xu ly (dang JSON)
113   * @return response: phan hoi tu server (dang string)
114   * @throws IOException
115   */
116  public static String post(String url, String data, String token) throws IOException {
117      // cho phep PATCH protocol
118      allowMethods("PATCH");
119
120      // phan 1: setup
121      HttpURLConnection conn = setupConnection(url, "GET", token);
122
123      // phan 2: gui du lieu
124      Writer writer = new BufferedWriter(new OutputStreamWriter(conn.getOutputStream()));
125      writer.write(data);
126      writer.close();
127
128      // phan 3: doc du lieu gui ve tu server
129      String response = readResponse(conn);
130
131      return response;
132  }
```

1.3. BÀI TẬP

- Thực hành thêm javadoc cho các class, method và attribute cho UC Place Rush Order. Sau đó tiến hành review code và refactoring lại những điểm code chưa hợp lý (lưu ý cần chỉ rõ cần refactoring ở điểm nào và tại sao)

HẾT