Assignment 09

Programming

* 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. **NỘI DUNG CHI TIẾT**

#### 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 ?

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

* Nếu như sau khi xem xong và bạn đảm bảo hiểu hết những gì viết 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 hiêu 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,..
* Vây bài học hôm nay sẽ giúp các bạn giải quyết từng vấn đề và bạn có thể áp dụng vào trong Project của mình

#### 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 phầm 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 annotatio 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/>

#### Thực hành tạo Javadoc với Eclipse

1. **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

1. **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 shortcurt. 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

Graphical user interface, text, application, email

Description automatically generated

* 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:

Graphical user interface, text, application, website

Description automatically generated

* 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***

Graphical user interface

Description automatically generated with low confidence

* ***get method***

Graphical user interface, text, application

Description automatically generated

* ***post method***

Text

Description automatically generated

* ***allowMethods***

Graphical user interface, text, application

Description automatically generated

1. **Thực hành thêm comment cho các phương thức**
   * Để làm rõ hơn các câu lện trong từng method thì chúng ta có thể thêm comment (//) vào trong các method như sau

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

#### Làm quen với refactoring

Diagram

Description automatically generated

1. **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 ra dễ quản lý mã nguồn và đưa được product của mình ra thị trường một cách nhanh chóng

1. **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

1. **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

#### Thực hành refactoring code với class API

1. **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:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

* 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.

1. **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()***

Graphical user interface, text, application

Description automatically generated

* + ***ReadResponse()***

Graphical user interface, text

Description automatically generated

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

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

## 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**