# Ôn phỏng vấn

Phân biệt UI và UX

* **UI (User Interface) – Giao diện người dùng**: Đây là những gì người dùng nhìn thấy và giúp họ tương tác với hệ thống (giao diện Web, button, …).
* **UX (User Experience) – Trải nghiệm người dùng**: Đây là những gì người dùng trải nghiệm khi sử dụng sản phẩm và dịch vụ (Bao gồm cảm xúc, suy nghĩ trong quá trình sử dụng). UI chỉ là một phần của UX.

UI là một trong những công cụ xử lý mạnh nhất trong việc sắp xếp các yếu tố UX. Tại sao? Đơn giản, giao diện là phương pháp xúc tác và hiện hữu mà người dùng có thể trải nghiệm với chúng tôi.

UI là công cụ bổ trợ.

**Phân biệt web tĩnh và web động**

* **Web tĩnh** tức là website không có hệ thống quản lý nội dung, hoặc có nhưng về nội dung website thì không thay đổi được.
* **Web động** tức là website có hệ thông quản lý nội dung nên người dùng có thể thay đổi nội dung được.

DOCTYPE để làm gì

!Doctype không phải là một tag HTML, !Doctype chỉ cho trình duyệt web biết được phiên bản ngôn ngữ đánh dấu (markup language) nào được sử dụng trong trang web.

So sánh class và id, nên dùng khi nào

Về cơ bản ID và Class là giống nhau – sự khác biệt nằm ở chỗ, khi khai báo id cho một thẻ nào đó thì tên id đó không được dùng lại nữa – tức là bạn không được khai báo tên đó cho thẻ khác, trong khi đó class thì khai báo một lớp các thẻ và chúng sẽ có cùng chung thuộc tính trong CSS.

so sánh margin padding

padding là khoảng trống nằm giữa nội dung và viền.

margin là khoảng trống nằm giữa viền và phần tử tiếp theo.

Back-end:

Kiểu dữ liệu đối tượng được chia làm 2 loại:

* Framwork định nghĩa: Những đối tượng này được tạo sẵn và công việc của chúng ta chỉ việc gọi lên và sử dụng chúng, ví dụ: File, String, Scanner...
* Do người dùng định nghĩa: Chính là những class được chúng ta define từ những đối tượng trong bài toán OOP(Object-oriented programming), ví dụ: Học Sinh, Nhân Viên, Giáo Viên, Lập Trình Viên...

điểm khác biệt giữa 2 kiểu dữ liệu đối tượng và nguyên thủy đó là 1 thằng có tính tham chiếu và 1 thằng có tính tham trị và khi sử dụng chúng nó sẽ khác nhau về việc cấp phát địa chỉ ô nhớ và các refernces vào địa chỉ ô nhớ đó.

chúng ta đã được tiếp cận với từ khóa break. Ngoài việc dùng trong cấu trúc switch, từ khóa break được dùng để thoát ra khỏi vòng lặp chứa nó ngay lập tức và chuyển sang câu lệnh tiếp theo bên ngoài vòng lặp vừa kết thúc (tức là chương trình sẽ ngừng ngay mọi vòng lặp nếu bên trong vòng lặp đó có chứa từ khóa *break*). Thông thường, từ khóa break thường được dùng với một lệnh if bên trong vòng lặp để kiểm tra điều kiện dừng của vòng lặp.

Ngoài từ khóa break, Java còn hỗ trợ cho chúng ta một từ khóa đặc biệt khác cũng được dùng kết hợp với cấu trúc lặp đó là từ khóa continue. Khi gặp từ khóa continue thì lần lặp kế tiếp sẽ được thực hiện (tức là bỏ qua không thực hiện các lệnh phía bên dưới từ khóa *continue* của vòng lặp và quay lên kiểm tra trở lại biểu thức điều kiện lặp). Tương tự như break, từ khóa continue cũng thường được dùng với một lệnh if bên trong vòng lặp để kiểm tra khi nào thì cần bỏ qua những lệnh sau nó để tiếp tục thực hiện vòng lặp mới.nghĩa là kiểm tra điều kiện if nếu break thì thoát ra vòng lặp continue sẽ bỏ qua lượt lặp là lặp tiếp điều kiện.

Sự khác nhau duy nhất giữa ++i và i++ đó là:

**++i** tăng giá trị của **i**, rồi sau đó trả về giá trị **đã tăng** của **i**, ví dụ,

int i = 1;

int j = ++i;

// j = 2 và i = 2

Tăng giá trị của I lên xong gián cho cả 2 biến

int i =1;

int j = i++;

// j = 1 và i = 2

Gián gia trị ban đầu cho j, sau đó tang I lên và gián giá trị I tăng

* ++i tính luôn trên giá trị ban đầu.
* i++ tạo ra bản copy của giá trị ban đầu.

Cho nên đối với những kiểu dữ liệu phức tạp như lớp (class), ...v.v. có thể i++ sẽ lâu hơn một chút so với ++i bởi vì nó phải tạo ra bản copy của đối tượng.

Còn sử dụng ++i và i++ với vai trò là biến đếm trong vòng lặp for là như nhau cả về tốc độ thực thi và tăng dần i, bạn có thể tùy thích sử dụng trường hợp nào cũng được.

# Constant

Trường constant hay biến constant phải được gán giá trị lúc khai báo sau đó thì chúng không thể thay đổi giá trị. Mặc định constant là static nên không thể khai báo thêm từ khóa static cho biến constant.

Bạn có thể áp dụng từ khóa const vào các kiểu nguyên thủy (byte, short, int, long, char, float, double, decimal, bool), enum, một chuỗi, hoặc một kiểu tham chiếu và có thể gán với giá trị null.

Biến constant có thể được gán tất cả các access modifiers như public, private, protected, internal. Bạn sử dụng biến constant trong trường hợp chắc chắn giá trị của chúng không thay đổi.

# ReadOnly

Một biến Readonly có thể được khởi tạo vào thời điểm khai báo hoặc trong constructor của class đó. Vì vậy các biến readonly có thể được sử dụng như là các hằng số lúc thực thi (run-time constants)

Readonly có thể được áp dụng cho cả tham giá trị và kiểu tham chiếu trừ delegate và event. Sử dụng readonly khi bạn muốn tạo ra biến constant ở lúc runtime.

# Static

Từ khóa static được sử  dụng để tạo một biến hoặc một thành phần tĩnh, nghĩa là giá trị của nó sẽ được chia sẻ cho toàn bộ các đối tượng và không gắn vào một đối tượng cụ thể nào. Từ khóa static có thể được áp dụng cho cả class, fields, properties, operators, events, constructor nhưng không dùng được cho index, destructors, hay kiểu nào hơn các classs.

Kiến thức về từ khóa static

* Nếu từ khóa static được áp dụng cho 1 class tất cả các thành phần trong class cũng phải static
* Phương thức static chỉ có thể truy cập các thành phần static khác  trong class. Các static properties được sử dụng để set và get giá trị cho các giá trị của biến static trong class.

Có 4 loại thành viên tĩnh chính:

* Biến tĩnh (static variable).
* Phương thức tĩnh (static method).
* Lớp tĩnh (static class).
* Phương thức khởi tạo tĩnh (static constructor).

## Cú pháp Biến tĩnh:

**<phạm vi truy cập> static <kiểu dữ liệu> <tên biến> = <giá trị khởi tạo>;**

Bạn có thể hiểu **biến tĩnh** là:

* Một biến dùng chung cho mọi đối tượng thuộc lớp.
* Nó được khởi tạo vùng nhớ 1 lần duy nhất ngay khi chương trình được nạp vào bộ nhớ để thực thi và huỷ khi kết thúc chương trình.

Sử dụng git  
git clone (địa chỉ clone SSH) \_ down file trên git.

Git status \_ kiểm tra tình trạng file.

Git add \* or git add (tên file) \_thêm tất cả file vào local.

Git commit –m “content cmt” \_ cmt cho lượt đẩy code.

Git commit –a –m”content cmt” \_ update file.

Git push \_ đồng bộ hoá local với git.

* Để quay trờ lại,xoá lần update

Git log \_ để lấy id những lần update. (nhấn CTRL+C: dừng lệnh)

Git revert id lần cập nhât.(nhấn esc , shift+:, q, enter), để trờ về màn hình.