My name is Nguyễn Hoàng Vinh, born in 2001. I am currently a senior student majoring in Computer Science at Ho Chi Minh City University of Technology - HCMUT. I also hold an Associate Degree in Application Development from FPT POLYTECHNIC HCM, which I completed in 2021. I possess strengths in handling high-pressure situations, a strong appetite for learning, good teamwork skills, and the ability to adapt to diverse work environments.

I acknowledge that I may have limited knowledge of Java frameworks at this stage. However, I am actively committed to daily self-improvement and continuous learning to solidify my understanding of these frameworks.

**JAVA:**

Ngôn ngữ java làm một trong những ngôn ngữ lập trình hướng đối tượng. Nó thường được dùng để phát triển phần mền, trang web, game hay ứng dụng di động.

Java được khởi đầu bới Jame Gosling và bạn đồng nghiệp Sun Microsystem vào năm 1991.

Lập trình hướng đối tượng (object oriented programing) là một phương pháp phát triển phần mềm, mà trong đó mọi thứ được coi là 1 đối tượng. Mỗi đối tượng có các thuộc tính (attributec) va phương thức (methods).

**Lợi ích của lập trình hướng đối tượng.:**

Dễ tái sử dụng, dễ bảo trì, dể mỡ rộng , và linh hoạt, và tối ưu.

**4 Tính chất** trong lập trình hướng đối tượng:

Tính đa hình, tính kế thừa, tính đống gói, tính trừu tượng.

**Phân biệt public, private, protected, no modifier.**

Public: truy cập bất kì nơi nào.

Private: chỉ truy cập bên trong class chứa nó.

Protechtic: chỉ truy cập trong package hoặc các subclass.

No modifier: chỉ truy cập bên trong cùng một package.

Hiểu về getter và setter.

* Getter dung để đọc giá trị
* Setter dung để cập nhật giá trị
* Getter và Setter giúp trong việc kiểm soát truy cập và cập nhật dữ liệu bên trong đối tượng,

Chúng rất quan trọng vì: có tính bảo mật, kiểm soát hợp lệ dữ liệu, bảo vệ tính toàn vẹn dữ liệu, phẩn ảnh ẩn thông tin, dể dàng thay đổi cài đặt nội bộ.

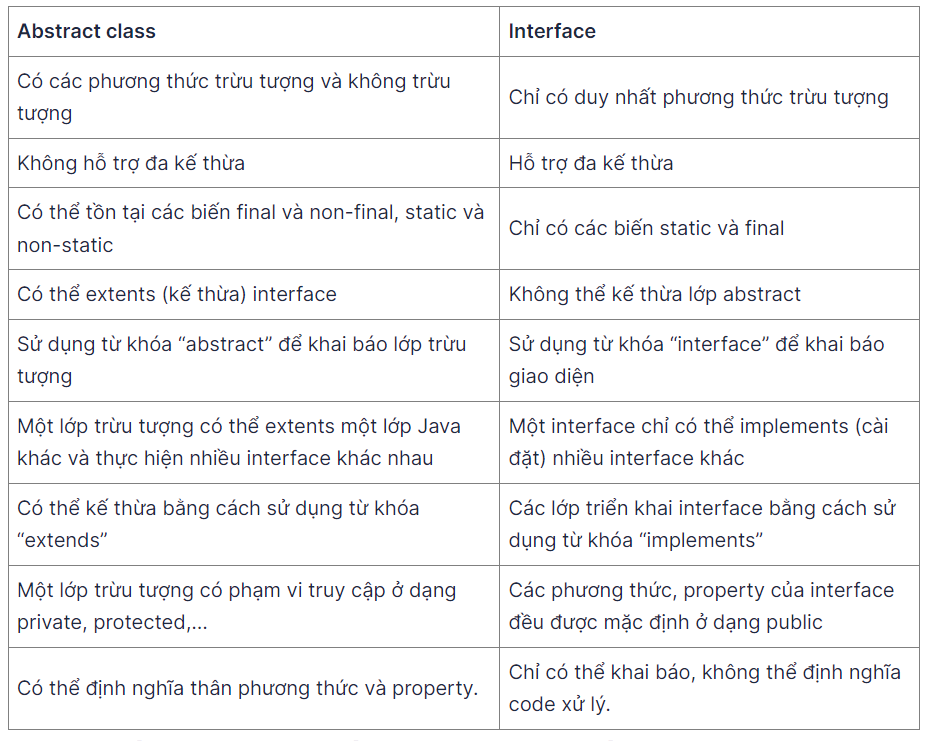
Các thuật toán sắp xếp:

1. Sắp xếp nổi bột (bubble sort)
2. Sắp xếp chèn (Incertion sort)
3. Sắp xếp chọn ( selection sort)
4. **Sắp xếp nhanh (Quick sort)**
5. **Sắp xếp trộn (Merge sort)**
6. **Sắp xếp vun đất (Heap sort)**
7. Sắp xếp đếm (Couting sort)
8. **Sắp xếp độc lập (radix sort)**
9. **Sắp xếp nổi bọt (Bubble Sort)**: Thuật toán này so sánh lần lượt các cặp phần tử kế cận và hoán đổi chúng nếu chúng không được sắp xếp. Thuật toán tiếp tục lặp lại quá trình này cho đến khi không còn phần tử nào cần hoán đổi.
10. **Sắp xếp chèn (Insertion Sort)**: Thuật toán này xem xét từng phần tử của dãy và chèn nó vào vị trí thích hợp trong dãy đã được sắp xếp. Thuật toán này hoạt động tốt với dãy nhỏ hoặc gần sắp xếp.
11. **Sắp xếp chọn (Selection Sort)**: Thuật toán này chọn phần tử nhỏ nhất trong dãy và đặt nó ở vị trí đầu tiên. Sau đó, nó chọn phần tử nhỏ thứ hai và đặt nó ở vị trí thứ hai, và tiếp tục cho đến khi dãy được sắp xếp.
12. **Sắp xếp nhanh (Quick Sort)**: Quick Sort sử dụng phân chia và chia để trị để sắp xếp dãy. Nó chọn một phần tử chốt, sau đó chia dãy thành hai phần: một phần chứa các phần tử nhỏ hơn chốt và một phần chứa các phần tử lớn hơn chốt. Quá trình này được lặp lại cho cả hai phần đó.
13. **Sắp xếp trộn (Merge Sort)**: Merge Sort chia dãy thành các phần nhỏ, sắp xếp từng phần riêng lẻ, sau đó trộn các phần đã sắp xếp lại với nhau để tạo dãy đã sắp xếp.
14. **Sắp xếp vun đất (Heap Sort)**: Heap Sort sử dụng cấu trúc dữ liệu heap (vun đất) để sắp xếp dãy. Nó chuyển dãy thành một heap và sau đó lấy phần tử tối ưu (thường là phần tử nhỏ nhất) ra khỏi heap cho đến khi dãy được sắp xếp.
15. **Sắp xếp đếm (Counting Sort)**: Counting Sort là một thuật toán sắp xếp dựa trên đếm số lần xuất hiện của từng giá trị trong dãy. Nó thích hợp cho dãy các số nguyên có giá trị trong một khoảng nhất định.
16. **Sắp xếp độc lập (Radix Sort)**: Radix Sort là một thuật toán sắp xếp dựa trên phân cơ số của các số trong dãy. Nó thường được sử dụng cho các số nguyên không âm.

**Các thuật toán Seach.**

1. Tìm kiếm tuyến tính (liner seach)
2. Tìm kiếm nhị phân (Binari seach)
3. Tìm kiếm nội suy (Interpolation seach)
4. Tìm kiếm dãy con (Substring seach)
5. Tìm kiếm đa chiều (Multi-dimensional seach)
6. Tìm kiếm đường đi trong đồ thị (Pathfinding in Graphs).

**Phân biệt giữa Abstract và Interface**



**Khi nào nên dung interface**

Muốn sử dụng nhiều loại kế thừa.

Tổng hợp các đối tượng dựa trên các hành vi phổ biến

Sử dụng đa hình vì một lớp sử dụng nhiều Interface.

**Khi nào dung astraction**

Chia sẻ code giữa các lớp liên quan với nhau

Muốn những lớp trừu tượng extend có nhiều method hoặc truy cập điều chỉnh thay vì public nó

**Phân biệt giữa overloading và overriding**

Overloading (Nạp chồng): là phương thưng cùng có cùng tên trong cùng một class hoặc interface.

Các phương thức overloading có cùng tên giống nhau, nhưng khác về tham số (khác về số lượng tham số hoặc khác về kiểu dữ liệu)

Overriding (Ghi đè): là triển khai một phương thức từ lớp cha trong lớp con (lớp kế thừa) có cùng tên, cùng kiểu dữ liệu, và cùng số lượng tham số. Overriding cho lớp con thực thi phương thức mà lớp cha đã định nghĩa.

**Phân biệt giữa Array và ArrayList**

Array (Mảng): nó sẽ cũng cấp một cấu trúc dữ liệu tĩnh. Khi khai báo một mảng thì phải xác định kích thước của mảng đó và nó sẽ không thay đổi sau khi được tạo ra. Mảng thì có thể chứa các phần tử có cùng kiểu dữ liệu và được truy cập thông quan index của mỗi phần tử.

ArrayList: là một cấu trúc dữ liệu động. nó có thể them hoặc loại bỏ các phần tử mà không cần biết trước kích thước. ArrayList nó có nhiều phương thức sẵn có thể thêm hoặc loại bỏ, tìm kiếm hay săp xếp dữ liệu.

**Phân Biệt giữa List/Set/Map**

**List:** là một giao diện và là một cấu trúc dữ liệu cho phép lưu trữ danh sách các phần tử theo thứ tự. Các phần tử trong một danh sách có thể trùng lặp. Truy cập các phần tử bằng chỉ mục index.

**Set:** cũng là một giao diện và là một cấu trúc dữ liệu lưu trữ tập hợp các phần tử không trùng lặp.Phần tử trong tập hợp là duy nhất, và không có tứ tự cụ thể trong tập hợp

**Map:** là một cấu trúc dữ liệu dung để ánh xạ các cặp khóa-giá trị khóa không thể trùng với nhau, nó là duy nhất, nhưng giá trị thì có thể trung lập với nhau.

**Comparable và comparator**

**Comparable** cho phép tự định nghĩa cách so sánh với các đối tượng cùng loại. nó có chứa một phương thức là comparaTo(), khi sử dụng nó trả về một giá trị số nguyên,

Phương thứ này trả về giá trị âm nếu đối tượng đó nhỏ hơn đối tượng so sánh.

Trả về 0 nếu 2 đối tượng bằng nhau.

Trả về giá trị dương nếu đối tượng hiện tại lớn hơn đối tượng so sánh

**Comparator** được dung để sắp xếp các đối tượng của lớp do người dung định nghĩa,có chưa một phương thức là compare() dung để so sánh đối tượng đầu tiên và đối tượng thứ 2.

**So sánh giữa String và StringBuilder và StringBuffer**

**String:** là một lớp trong java, nó có một đặt điểm quan trọng là chuỗi String là không thể thay đổi (immutable). có nghĩa là một khi đối tượng String đc tạo ra thì giá trị của nó không bị thay đổi.

Ví dụ như khi thực hiện các thao tác như cộng chuỗi hoặc thay đổi chuổi, java sẽ tạo ra một đối tượng String mới thay vì thay đổi đối tượng hiện tại.

**StringBuilder:** là một lớp trong java được thiết kế dung để thay thế String khi thực hiện nhiều thao tac sửa đổi trên chuổi. Stringbuilder là biến đổi (mutable), có nghĩa là có thể thay đổi giá trị trong chuổi àm không cần tạo ra một đối tượng mới. Stringbuilder không an toàn với đối luồng (non-thread-safe), nó không đảm bảo tính nhất quán khi sử dụng đồng thời trong các luồng khác nhau.

**StringBuffer:**  là một đối tượng giống như Stringbuilder nhưng nó an toàn trong luồn (thread-safe), nghĩa là nó có các phương thức được đồng bộ hóa để đảm bảo tính nhất quán khi sử dụng.

**Immuatable là gì?**

Immutable hiểu nôm na là không thể thay đổi còn mutable là có thể thay đổi.

2 khái niệm Immutable và mutable thường được dùng class, object (Immutable đôi khi còn dùng với Collection nhưng mình sẽ không để cập ở bài này).

**This và Super?**

**This:** là một từ khóa tham chiếu đến đối tượng hiện tại của lớp, nó được sử dụng để tránh xung đột giữa các tên biến cục bộ và biến instance khi chúng có cùng tên. This cũng có thể được sử dụng để gọi phương thức của đối tượng hiện tại

**Super:** là một từ khóa tham chiếu đển đối tượng của lớp cha, nó được sử dụng khi lớp con muốn gọi một phương thức hay truy cập một trường của lớp cha mà nó đang mở rộng.

**Static và Final?**

**Static** dung để chỉ một thành phần của lớp, biến hoặc phương phức thuộc về lớp chứ không phải đối tượng cụ thể của lớp đó. Các thành phần static được chia sẻ chung giữa tất cả các đối tượng của lớp và có thể truy cập mà không cần tạo ra một đối tượng.

Biến static được gọi là biến lớp và được chia sẻ giữa tất cả các đối tượng của lớp. Phương thức static được gọi trực tiếp từ lớp mà không cần tạo một đối tượng.

**Final** là một từ khóa dùng để chỉ rằng một biến, phương thức hoặc lớp không thể được thay đổi sau khi đã được khởi tạo hoặc định nghĩa.

Với biến, nó có thể chỉ được gán một lần.

Với phương thức, nó không thể bị ghi đè (override) ở các lớp con.

Với lớp, nó không thể được kế thừa bởi các lớp con.

SQL:

**Truy vấn con (Subquery):** là một truy vấn được nhúng bên trong một truy vấn lớn hơn để trả về một tập hợp kết quả phụ thuộc vào truy vấn chính.

**Stored procedure:** là một tập hợp các câu lệnh SQL được lưu trữ trong cơ sở dữ liệu để được gọi và thực thi theo yêu cầu. Stored procedure thường được sử dụng để thực hiện các nhiệm vụ cụ thể hoặc chức năng trong cơ sở dữ liệu mà có thể được gọi.

Stored procedure có một số **lợi ích**, bao gồm:

* Tính Tái Sử Dụng: Stored procedure có thể được sử dụng nhiều lần từ nhiều ứng dụng khác nhau.
* Tăng Hiệu Suất: Stored procedure có thể được biên dịch một lần và sử dụng nhiều lần, giảm chi phí tính toán.
* Bảo mật: Bạn có thể kiểm soát quyền truy cập vào cơ sở dữ liệu thông qua stored procedure.
* Quản lý Lỗi: Stored procedure có thể xử lý lỗi và trả về thông báo phù hợp.

**Trigger** là một thủ tục SQL được thực thi ở phía server khi có một sự kiên như Inser, Delete, Update. Trigger là một loại của strored procedure đặt biệt (không có tham số) được thực thi (execute) một các tự động khi có một sự thay đổi dữ liệu. Trigger được lưu trữ và quản lý trong Server DB, được dùng trong trường hợp ta muốn kiểm tra các ràng buộc toàn vẹn trong DB.

**Trigger được sử dụng khi nào?**

Trigger thường được sử dụng để kiểm tra rang buộc, trên nhiều quan hệ trên bảng hoặ trên nhiều dòng của bảng.

Ngăn chặn việc xóa những dữ liệu quan trọng. (có thể dùng back up các dữ liệu quan trọng sang table khác phòng khi …bị xóa ngoài ý muốn).

**Ưu điểm của Trigger**

Trigger có thể bắt lỗi business logic ở mức csdl.

Có thể dùng trigger là một cách khác để thay thế việc thực hiện những công việc hẹn giờ theo lịch.

Trigger rất hiệu quả khi được sử dụng để kiểm soát những thay đổi của dữ liệu trong bảng.

**Nhược điểm của Trigger**

Trigger chỉ là một phần mở rộng của việc kiểm tra tính hợp lệ của dữ liệu chứ không thay thế được hoàn toàn công việc này.

Trigger hoạt động ngầm ở trong csdl, không hiển thị ở tầng giao diện. Do đó, khó chỉ ra được điều gì xảy ra ở tầng csdl.

Trigger thực hiện các update lên bảng dữ liệu vì thế nó làm gia tăng lượng công việc lên csdl và làm cho hệ thống chạy chậm lại.

**UDF (User-Defined Function)** là một khái niệm trong cơ sở dữ liệu, cho phép người dùng tự định nghĩa và sử dụng các hàm tùy chỉnh trong câu lệnh truy vấn. UDF giúp tăng tính linh hoạt và tái sử dụng mã trong cơ sở dữ liệu. Có hai loại UDF chính: Scalar Functions và Table-Valued Functions

**Scalar Function** trả về một giá trị duy nhất và có thể được sử dụng trong các phần tử của câu lệnh SELECT, WHERE, ORDER BY, và nhiều ngữ cảnh khác..

**Table-Valued Function** trả về một bảng kết quả và có thể được sử dụng giống như một bảng trong các câu lệnh SELECT.

**Phân biệt giữa Stored Procedure (SP) và UDF**

**Stored Procedure (SP):**

**Trả Về Kiểu VOID hoặc INTEGER** (sự thành công/hỏng) hoặc Một Bảng: Stored Procedure thường không trả về bất kỳ giá trị nào hoặc chỉ trả về một kiểu INTEGER để chỉ ra sự thành công hoặc thất bại. Cũng có thể trả về một hoặc nhiều bảng kết quả.

Có Thể Chứa Tham Số Đầu Vào và Đầu Ra: SP có thể chứa tham số đầu vào và đầu ra, cung cấp khả năng truyền giá trị vào và nhận giá trị từ SP.

Thực Hiện Các Hành Động Cụ Thể: Thường được sử dụng để thực hiện các hành động như cập nhật, chèn, xóa dữ liệu, và có thể chứa logic phức tạp.

Không Thể Sử Dụng Trong Câu Lệnh SELECT: Stored Procedure không thể được sử dụng trực tiếp trong câu lệnh SELECT để trả về giá trị.

**User-Defined Function (UDF):**

Trả Về Giá Trị: UDF luôn phải trả về giá trị, có thể là giá trị scalar hoặc table-valued.

Chỉ Chứa Tham Số Đầu Vào: UDF chỉ chứa tham số đầu vào, không thể có tham số đầu ra. Giá trị trả về là kết quả của UDF.

Chủ Yếu Dùng Để Truy Vấn Dữ Liệu: Thường được sử dụng để thực hiện các truy vấn dữ liệu và tính toán trên dữ liệu.

Có Thể Sử Dụng Trong Câu Lệnh SELECT: UDF có thể được sử dụng trực tiếp trong câu lệnh SELECT để trả về giá trị, làm cho chúng hữu ích trong việc tích hợp logic vào truy vấn.

Chỉ Đọc Dữ Liệu: UDF nên chỉ đọc dữ liệu và không được phép thực hiện các thao tác cập nhật dữ liệu.