Java:

- OOP là gì? : là 1 phương pháp để thiết kế một chương trình sử dụng các lớp và đối tượng

- các tính chất của OOP?:

* Kế thừa : một lớp có thể mở rộng sang lớp khác, vì vậy các mã có thể sử dụng lại từ lớp này sang

lớp khác. Lớp hiện có được gọi là lớp cha , lớp dẫn xuất gọi là lớp con

VD: khai báo 3 class Xe,XeDap,XeMay, Class Xe gọi là class cha nó chứa thuộc tính và phương thức của 1 chiếc xe, XeDap-XeMay extends Xe lúc này XeDap-XeMay cũng có đầy đủ thuộc tính và phương thức của Class Xe

* Đóng gói : Tính đóng gói là kỹ thuật ẩn giấu thông tin không liên quan và hiện thị ra thông liên quan. Mục đích chính của đóng gói trong java là giảm thiểu mức độ phức tạp phát triển phần mềm

VD:+ Khai báo các biến của Class là private

+ Cung cấp phương thức setter và getter là public để có thể sửa đổi và xem các giá trị biến

* Đa hình: cùng 1 hành động ở ngữ cảnh khác nhau thì cho ra kết quả khác nhau (

Overriding: được sử dụng dùng để cung cấp cài đặt cụ thể cho phương thức được khai báo ở lớp cha: đa hình lúc run time

Overloadding: nạp chồng phương thức được sử dụng giúp code chương trình dễ đọc hơn )

VD: Khi cùng thực hiện 1 hành động thanh toán đơn hàng. Ngữ cảnh khác nhau (người dùng có thể chọn thanh toán tiền mặt hay thanh toán qua ngân hàng) thì sẽ cho ra kết quả khác nhau

* Trừu tượng : Ẩn đi cách thực thi cụ thể

VD: như tạo ra tính năng gửi tin nhắn, ta chỉ cần hiểu là người dùng viết tin rồi nhấn gửi đi. Còn quy trình xử lý tin nhắn gửi như thế nào thì ta chưa đề cập đến.

- Để tạo lớp trừu tượng ta dùng từ khóa abstract trước từ khóa class

- Acces Modifile ?

+ xác định phạm vi có thể truy cập của biến, phương thức, constructor hoặc lớp

+ có 4 phạm vi truy cập của acces modifile

1 private: truy cập trong phạm vi của lớp (một lớp không thể là private hay protected ngoại trừ lớp lồng nhau)

2 default: khi không khai báo modifile nào, được phép truy cập trong cùng package

3 protected: có thể truy cập được bên trong và bên ngoại package nhưng phải kế thừa, có thể áp dụng cho biến, phương thức, constructor,

4 public: truy cập ở mọi nơi

- So sánh giữa String, String buffer và String builder? (hỏi inmutable và mutable)

+ String : không thể thay đổi (immutable) , không có phép có class con.

+ Stringbuffer:có thể thay đổi (mutabel)

+ Stringbulider : có thể thay đổi (mutabel)

String buffer và Stringbulider là giống nhau, nó chỉ khác biệt tình huống sử dụng có liên quan tới đa luồng. (mulyit thread). Về tốc độ sử lý thì stringbulider có tốc độ nhanh nhất sau đó đến stringbuffer,cuối cùng là string

* String có tính chất immutable nên khi khi thực hiện các hành động như nối chuỗi, cắt chuỗi thì nó sẽ khởi tạo một object string mới khi chúng ta xử lý quá nhiều hành động trên String thì nó sẽ sinh ra rất nhiều giá trị rác ở heap. vậy nên ta sử dụng StringBuffer trong trường hợp chúng ta cần thao tác nhiều trên chuỗi

Trong trường hợp nối chuổi nên sử dụng buffer

- Làm thể nào để biến objech sang mutable?

+ khởi tạo hàm setter cho object để có thể sửa các trường.

- Làm thế nào để không dùng 1 String thứ 3 mà vẫn đổi được

- Conletion có mấy loại

4 loại conletion: set,list,queue,map:

Set: là kiểu dữ liệu mà bên trong mỗi phần tử chỉ xuất hiện 1 lần duy nhất

List: là cấu trúc dữ liệu tuyến tính trong đó các phần tử được sắp xếp theo một thứ tự xác định

Queue: được dử dụng khi chúng ta muốn lưu trữ và truy cập các phần tử theo cái FIFO

Map: cho phép các phần tử lưu trữ trong các cặp key/value

**ArrayList**

* ArrayList sử dụng một mảng động
* Truy xuất phần tử trong ArrayList nhanh hơn LinkedList. Bởi vì các phần tử trong ArrayList được lưu trữ dựa trên chỉ mục (index).
* ArrayList có thể truy xuất ngẫu nhiên phần tử.
* Thao tác thêm và xóa phần tử với ArrayList là chậm bởi vì nó sử dụng nội bộ mảng. Bởi vì sau khi thêm hoặc xóa các phần tử cần sắp xếp lại
* ArrayList yêu cầu ít bộ nhớ hơn so với LinkedList. Bởi vì ArrayList chỉ lưu trữ dữ liệu (data) của nó và chỉ mục (index).
* ArrayList tốt hơn trong việc truy suất dữ liệu**.**
* **Động**
* **Tốc độ xử lý nhanh**
* **Truy xuất ngẫu nhiên**
* **Crud chậm**
* **Thích hợp truy xuất**
* **Sử dụng ít bộ nhớ**
* **LinkedList:**
* LinkedList sử dụng danh sách liên kết doubly.
* Thao tác truy xuất phần tử trong LinkedList chậm hơn nhiều so với ArrayList. Bởi vì, nó phải duyệt qua lần lượt các phần tử từ đầu tiên cho đến cuối cùng
* Thao tác thêm và xóa phần tử với LinkedList nhanh hơn ArrayList. Bởi vì nó không cần sắp xếp lại các phần tử sau khi thêm hoặc xóa. Nó chỉ cần cập nhật lại tham chiếu tới phần tử phía trước và sau nó
* LinkedList không thể truy xuất ngẫu nhiên. Nó phải duyệt qua tất cả các phần tử từ đầu tiên đến phần tử cuối cùng để tìm phần tử
* LinkedList yêu cầu nhiều bộ nhớ hơn so với ArrayList. Bởi vì LinkedList lưu giữ thông tin của nó và tham chiếu tới phần tử trước và sau nó.
* LinkedList tốt hơn việc thao tác dữ liệu thêm xóa
* **Liên kết đôi**
* **Tốc độ chậm**
* **Crud nhanh**
* **Ko thể truy xuất ngẫu nhiên**
* **Tốn nhiều bộ nhớ**
* **Thích hợp crud**

thêm vào nhanh chậm ntn? : list,set,queue,map

set nhanh nhat

- so sánh Static và Final ? ( 1 phương thức Static có đc overri ko?)

**Static**: được sử dụng để quản lý bộ nhớ tốt hơn, có thể truy cập trực tiếp không cần thông qua lớp khởi tạot

* Có thể sử dụng static với các biến, phương thức,các khối,các lớp lồng nhau
* Khi khai báo 1 biến là static: thì biến đó được gọi là biến tĩnh
* Một class có thể được đặt là static khi nó là 1 lớp lồng nhau

Khối static :

+ được dùng để khởi tạo hoặc thay đổi giá trị của các biến static

+ được thực thi trước khi phương thức main tại thời gian tải lớp

+ một class có thể có nhiều khối static

**Final**:

* khi 1 biến được khai báo là final, nó chỉ chứa 1 giá trị duy nhất trong toàn bộ chương trình
* khi 1 phương thức khai báo là final các class con sẽ không thể ghi đè phương thức này
* khi từ kháo final sử dụng cho 1 lớp , lớp này sẽ không được kế thừa
* Một biến final mà không được khởi tạo tại thời điểm khai báo được gọi là final trống

Gia tri duy nhat

Khong the ghi de

Class k the ke thua

Tai thoi diem khoi tao khong gan gia tri thi la final trong

* Không thể khi đè phương thức static trong java : phương thức ghi đè dựa trên quá trình (binding) động khi chương trình đang chạy và những phương thức static được gán tĩnh trong thời gian biên dịch. Phương thức tĩnh không rang buộc với thực thể của lớp nên phương tích không thể ghi đè

K the ghi de phuong thuc static vi ghi de dua tren qua trinh binding khi chuong trinh dang chay và phương thức static được gán tĩnh trong thời gian biên dịch, pthuc tĩnh k ràng buộc vói lớp nên k thể gđè.

- String POOl là gì?

* String pool là 1 vùng nhớ đặc biệt nằm trong vùng nhớ heap dùng để lưu trữ các biến được khai báo kiểu string
* String pool giúp tối ưu hóa việc lưu trữ và sử dụng vùng nhớ khi khai báo biến string,giúp hạn chế tình trạng tràn bộ nhớ heap space

Khác nhau giữa Has a và Is a ?

Has a : 1 class tham chiếu đến thực thể khác (ví dụ class student tham chiếu đến address)

Is a :1 class kế thừa từ lớp khác