**1. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG LÀ GÌ:**

-lập trình hướng đối tượng là 1 kỹ thuật lập trình, cho phép lập trình viên trừu tượng hóa các đối tượng thực tế thành các đối tượng trong code.

**2. CÁC TÍNH CHẤT CỦA LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG:**

-Có 4 tính chất:

+tính đóng gói: access modifier

+tính đa hình: overload, overriding

+tính trừu tượng: abtract, interfac

+tính kế thừa: extends

**3. ARRAY LIST LÀ GÌ? KHI NÀO DÙNG ARRAY LIST**

- Arraylist là 1 mảng để lưu trữ các phần tử bên trong, có thể điều chỉnh đc kích thước tùy theo số lượng các phần tử bên trong.

- Có thể lưu được nhiều kiểu dữ liệu

-Arraylist đc sử dụng khi muốn truy xuất (nhanh) các phần tử ngẫu nhiên bên trong nhiều và ít thêm, xóa các phần tử(chậm)

**4. LINKED LIST LÀ GÌ? KHI NÀO DÙNG LINKED LIST**

-LinkedList lưu trữ các phần tử sử dụng cơ chế liên kết theo các node

-LinkedList đc sử dụng khi muốn thêm hay xóa các phần tử bên trong (nhanh) nhìêu và ít truy xuất ngẫu nhiên(chậm)

**5. STACK HOAT ĐỘNG NTN? KHI NÀO DÙNG STACK**

-Stack là 1 cấu trúc dữ liệu danh sách, trong đó việc thêm và lấy các phần tử đc thực hiện theo quy tắc FILO(first in/last out)(vào trc/ra sau)

-Stack đc sử dụng khi chúng ta muốn lưu trữ dữ liệu theo dạng ngăn xếp

**6. QUEUE HOẠT ĐỘNG NTN? KHI NÀO DÙNG QUEUE**

-Queue là 1 cấu trúc dữ liệu danh sách, trong đó việc thêm và lấy các phần tử được thực hiện theo quy tắc FIFO(first in/firstout)(vào trc/ra tr)

-Queue đc sử dụng khi chúng ta muốn lưu trữ dữ liệu kiểu hàng đợi

**7. SET HOẠT ĐỘNG NTN? KHI NÀO DÙNG SET**

-**Set** là cấu trúc dữ liệu lưu trữ các phần tử không trùng nhau

-Có thể sử dụng các lớp:

+**hashSet**: lưu trữ các phần tử của nó trong bảng băm, tuy nhiên nó không đảm bảo về thứ tự các phần tử chèn vào

+**treeSet**: lưu trữ các phần tử của nó trong một cây, sắp xếp các phần tử của nó dựa trên các giá trị của chúng, về cơ bản là chậm hơn HashSet.

+**LinkedHashSet**: được triển khai dưới dạng bảng băm với có cấu trúc dữ liệu danh sách liên kết, sắp xếp các phần tử của nó dựa trên thứ tự chúng được chèn vào tập hợp (thứ tự chèn).

**8. MAP HOẠT ĐỘNG NTN? KHI NÀO DÙNG MAP**

-Map là cấu trúc dữ liệu lưu trữ danh sách các cặp key/value.

-Map k cho phép 2 key trùng lặp

-Map đc sử dụng khi phải tìm kiếm, cập nhật hoặc xóa các phần tử dựa vào các key.

+hashMap: tối ưu cho các thao tác tìm kiếm, thêm và xóa, k hỗ trợ sắp xếp

+linkedHashMap: kế thừa hashMap, hỗ trợ sắp xếp các entry (key, value)

+treeMap: tối ưu cho thao tác duyệt qua các entry theo trật tự sắp xếp của các key

**9. SO SÁNH TOÁN TỬ "==" VÀ EQUAL()**

-Toán tử "==" dùng để so sánh địa chỉ 2 đối tượng và giá trị của các biến nguyên thủy.

-Equal() dùng để so sánh giá trị 2 đối tượng cùng tham chiếu đến

**10. SỰ KHÁC NHAU GIỮA OVERRIDE VÀ OVERLOAD**

-Override: xuất hiện khi trong class con tồn tại một method được định nghĩa với cùng tên và cùng số lượng tham số với method của class cha, có cùng kiểu trả về.

-Overload: Overloading xuất hiện khi trong một class tồn tại một method được định nghĩa với cùng tên nhưng khác số lượng tham số đầu vào với method của class cha

**11. GENERIC LÀ GÌ? VÍ DỤ MINH HỌA? TẠI SAO DÙNG GENERIC**

-Generic là cơ chế cho phép sử dụng kiểu dữ liệu như là tham số

-Vd: Có thể truyền kiểu dữ liệu String cho Arraylist

-Lý do dùng generic:

+ giúp phát hiện lỗi ngay tại thời điểm biên dịch thay vì tại thời điểm thực thi

+ generic cho phép qui định các kiểu dữ liệu đc phép sử dụng ở trong 1 lớp hoặc 1 method

+nếu kiểu dữ liệu k phù hợp đc sử dụng thì sẽ đc phát hiện

**12. SỰ KHÁC NHAU GIỮA ABTRACT CLASS VÀ INTERFACE**

Aptract class:

-Aptract class có thể có các thuộc tính và phương thức bình thường

-không thể khởi tạo đối tượng của lớp abtract

-1 lớp k phải abtract kê thừa từ 1 lớp cha abtract thì bắt buộc phải triển khai các phương thức abtract của lớp cha

-lớp/phương thức abtract thì k thể là final

Interface:

-interface chỉ chứa các hằng số và abtract method

-interface quy định các hành vi chung cho các lớp triển khai nó

**13. THUẬT TOÁN TÌM KIẾM**

Có 2 loại:

-Thuật toán tìm kiếm tuyến tính(tìm kiếm tuần tự): là hoạt động tìm kiếm liên tiếp đc diễn ra qua tất cả các phần tử (đã sắp xếp hoặc chưa)

-Thuật toán tìm kiếm nhị phân: đc thực hiện trên mảng đã đc sắp xếp.Tìm kiếm phần tử cụ thể bằng cách so sánh phần tử tại vị trí giữa nhất của tập dữ liệu

**14. THUẬT TOÁN SẮP XẾP**

Có 3 loại cơ bản:

-Sắp xếp nổi bọt: so sánh các cặp phần tử liền kề nhau và tráo đổi thứ tự của chúng nếu chúng k theo thứ tự

-Sắp xếp chọn: chọn phần tử nhỏ nhất/lớn nhất trong dãy các phần tử ban đầu,đưa phần tử này về vị trí đầu dãy,sau đó loại nó khỏi danh sách sắp xếp tiếp theo rồi cứ lặp lại cho đến hết dãy.

-Sắp xếp chèn: chèn thêm 1 phần tử vào danh sách con đã qua sắp xếp.phần tử đc chèn vào vị trí thik hợp sao cho vẫn đảm bảo danh sách con đó vẫn sắp xếp theo thứ tự

**15. CÁC KIỂU DỮ LIỆU TRONG JAVA.SỰ KHÁC NHAU CỦA CÁC KIỂU ĐÓ**

Có 2 kiểu dữ liệu:

-Kiểu dữ liệu nguyên thủy: byte, short, int, long, char, float, double, boolean

giá trị của kiểu dữ liệu nguyên thủy nằm trong bộ nhớ stack

-Kiểu dữ liệu tham chiếu: biến mảng là 1 kiểu tham chiếu, object, string

giá trị của dữ liệu tham chiếu là nó tham chiếu đến giá trị nằm trong vùng nhớ heap

**16. PHÂN BIỆT CÁC ACCESS MODIFIER**

-Public: có thể truy cập từ bất cứ đâu

-Protected: các phương thức và thuộc tính đc phép truy xuất trong cung 1 lớp và các lớp con, có thể khác package

-Defauft (k có access modifier): lớp và các thành phần của lớp đc truy xuất ở những nơi trong cùng 1 package,k thể truy xuất ở lớp con

-Private: các phương thức và thuộc tính chỉ đc truy xuất trong cùng 1 lớp

-Static: đc sử dụng để khai báo các thuộc tính và phương thức của lớp, có thể truy xuất các thành phần static bằng các sử dụng lớp hoặc đối tượng, k cần khởi tạo đối tượng vẫn có thể sử dụng các thành phần static: giảm sử dụng bộ nhớ.

**17. NGOẠI LỆ (EXCEPTION) LÀ GÌ?**

-Là các lỗi phát sinh trong quá trình thực thi

-Các kiểu lỗi thường xảy ra:

+Lỗi cú pháp

+Lỗi khi chạy chương trình

+Lỗi về tính logic cấu trúc của chương trình

**18. CHECK VÀ UNCHECK LÀ GÌ?**

-Check: lỗi đc phát sinh và đc kiểm tra trong quá trình biên dịch

-Uncheck: lỗi đc phát sinh sau khi chạy chương trình

**19. XỬ LÝ NGOẠI LỆ LÀ GÌ? CÓ BN CÁCH XỬ LÝ NGOẠI LỆ**

-Xử lý ngoại lệ là cách xử lý lỗi phát sinh để chương trình có thể tiếp tục chạy thay vì dừng chương trình khi gặp lỗi

-Các cách xử ly ngoại lệ:

+sử dụng khối try-catch: dùng để bắt ngoại lệ và xử lý chúng

+sủ dụng từ khóa throw: (new) dùng để ném ngoại lệ, đc dùng khi cần phát sinh ngoại lệ trong 1 phương thức

+sủ dụng từ khóa throws: dùng để ném ngoại lệ, đc dùng vs phương thức để gây ra bất cứ check hoặc uncheck exception nào và đồng thời nhường công việc xử lý ngoại lệ đó cho người gọi phương thức đó

**20. PHÂN BIỆT THAM TRỊ VÀ THAM CHIẾU**

-Tham trị (pass by value):giá trị của biến đc lưu trên stack, khi chương trình thực thi thì sẽ tạo ra 1 giá trị mới bằng cách copy giá trị gốc và mình chỉ thao tác vs giá trị copy này.khi chúng ta thay đổi các giá trị copy này thì không ảnh hưởng đến giá trị gốc.Thường các biến nguyên thủy là theo kiểu tham trị

-Tham chiếu (pass by reference): địa chỉ của biến đc lưu trên stack nhưng nó tham chiếu đến giá trị bên vùng nhớ heap.khi thay đổi các giá trị này thì giá trị của biến cũng thay đổi theo.Thường mảng là kiểu tham chiếu

**21. SỰ KHÁC NHAU GIỮA CLASS VÀ OBJECT**

-Class là 1 mô hình chi tiết để bạn sử dụng tạo ra các object. Class định nghĩa các thuộc tính và phương thức cần thiết của 1 object

-Object là phải thuộc class nào đó.và 1 object là 1thể hiện của class.tất cả cá object thuộc class đều có cùng các thuộc tính và phương thức

vd: có 1 class house.nhà của bạn là 1 object và nhà hàng xóm cũng là 1 obiect nhưng đều là 1 thể hiện của class house

-Như vậy class và object là 2 khái niệm không thể tách rời.Object k tự nhiên sinh ra mà phải đc tạo ra từ 1 class, ngược lại sự tồn tại của class là vô nghĩa nếu như không đc cụ thể hóa bằng object

**22. SO SÁNH GIỮA COMPARABLE VÀ COMPARATOR**

-Giống nhau: đều là interface đc sử dụng để sắp xếp các phần tử của collection

-Khác nhau:

\* Comparable:

+ bạn phải implement interface comparable cho lớp đối tượng cần so sánh

+ cung cấp phương thức compareTo() để sắp xếp các phần tử

+ sắp xếp các phần tử bởi phương thức collection.sort(list)

\*Comparator:

+ bạn không phải implement interface comparator cho lớp đối tượng cần so sánh

+ cung cấp phương thức compare() để sắp xếp các phần tử

+ sắp xếp các phần tử bởi phương thức collection.sort(list,Comparator)