**1 JAVA là gì?**

Là ngôn ngữ lập trình hướng đối tượng, có slogan là write once, run anywhere.

**2 JDK, JRE, JVM?**

- JDK( Java Development Kit): là tool – bộ công cụ hổ trợ lập trình viên phát triển ứng dụng trong Java.

- JRE( Java Runtime Enivoment): môi trường thực thi ứng dụng.

- JVM( Java Virtual Machine): máy ảo Java – thực thi mã java byte code (.class) ra ngôn ngữ máy.

**3 Thông dịch, biên dịch khác nhau như thế nào ?**

- Biên dich: dịch toàn bộ file một lần, sau đó sử dụng kết quả biên dịch mà không cần dịch lại

- Thông dịch: dịch từng câu lệnh.

+ Từ file.java biên dịch ra file.class

+ Từ file.class được JVM thông dịch ra ngôn ngữ máy

**4 JAVA là ngôn ngữ thông dịch hay biên dịch ?**

**Cả hai**

**5 Java Platform là gì?**

Nền tảng Java - là 1 tập hợp các chương trình giúp phát triển và chạy các ctrinh đc viết bằng ngôn ngữ java. Java platform bao gồm: execution engine (công cụ thực thi), trình biên dịch (Compiler) và 1 bộ thư viện Java

**6 Các kiểu dữ liệu trong java, và giá trị mặc định khi khai báo.**

- Có 2 kiểu dữ liệu..

* Kiểu nguyên thủy.
* byte,shot,int,long,float,double,boolean,char.
* Kiểu đối tượng:
* String ,Arrays.

**7 Sự khác nhau giữa bộ nhớ heap và stack trong java?**

* Bộ nhớ Stack có dung lượng nhỏ hơn dùng để lưu

các biến cục bộ các lời gọi hàm hay các biến tham chiếu;

* Bộ nhớ Heap dùng để lưu các đối tượng dùng các biến

tham chiếu để trỏ đến các đối tượng để sử dụng

**8 Trình bày các cách khởi tạo một mảng trong JAVA ?**

* Có 2 cách:
* C1:Dùng từ khóa new;
  + VD:elementType[] arrayName = new element[];
* C2:Khởi tạo nhanh
  + element[]arrauName = {};
  + element arrayName[] = {};

**9 Phần tử của mảng có thể dùng kiểu dữ liệu nào, và có giá trị mặc định là gì ?**

* Kiểu nguyên thủy , kiểu đối tượng

**10 OOP là gì ?**

* Lập trình hướng đối tượng ánh sạ các sự vật hiện tượng đời sống vào bên trong lập trình

**11 Phân biệt class và object**

* Class là nơi định nghĩa các thuộc tính và hành vi của đối tượng,dùng từ khóa class để khai báo, class chỉ khai báo một lần không trùng trong phạm vi cùng 1 package
* Object là nơi thể hiện của class, dùng từ khóa new để khởi tạo, đối tượng có thể khởi tạo nhiều lần.

**12 Constructor là gì**

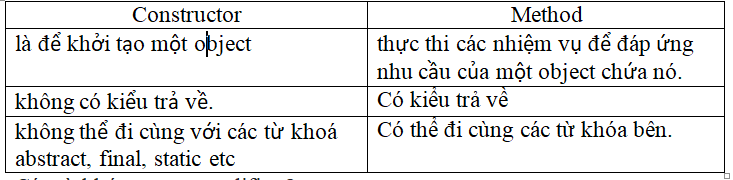
* Constructor là một phương thức đặc biệt, nó được dùng để khởi tạo và trả về đối tượng của lớp mà nó được định nghĩa.

**13 Cách khai báo constructor và đặc điểm constructor trong JAVA**

* Tên constructor phải trùng tên vs class.
* Không có kiểu trả về.
* Trong 1 class có thể có nhiều constructor.
* Nếu không khai báo constructor thì Java sẽ cung cấp 1 constructor mặc định không có tham số - body rỗng.
* Có thể gọi constructor khác thông qua từ khóa this(…)

**LƯU Ý: this(…) phải được gọi đầu tiên trong nội dung của phương thức khởi tạo.**

**14 Phân biệt constructor và method**



**15 "Tính bao đóng là gì ? Làm sao để thu được tính bao đóng trong java ?"**

* Tính bao đóng là khả năng che giấu đi những thông tin quan trọng của đối tượng chỉ hiển thị những thông tin cần thiết
* Mục đích : Bảo vệ trạng thái bên trong của đối tượng ,giảm độ phức tạp của chương trình
* Để đạt được tính bao đóng ta dùng
* Getter,setter
* Access modifiers;

**16 Tham trị, tham chiếu**

* Tham trị: là việc tham số sao chép giá trị của đối số mà nếu tham số thay đổi sẽ không ảnh hưởng tới đối số
* Tham chiếu:là việc tham số trỏ đến vùng nhớ của đối số mà nếu tham số thay đổi thì sẽ ảnh hưởng tới đối số;

**17 Từ khóa static dùng để làm gì ?**

* Từ khóa static dùng để khai báo thuộc tính phương thức, class.
* Khi đó thuộc tính và phương thức được khai báo bằng static sẽ chịu sự quản lí của class-Không phải của đối tượng
* Đặc điểm: Định nghĩa thuộc tính ,phương thức dùng chung cho toàn bộ đối tượng,tạo ra các lớp tiện ích.

**18 Ràng buộc khi sử dụng static**

* Phương thức static chỉ có thể gọi được phương thức static khác
* Phương thức static chỉ có thể truy cập được biến static.

**19 Các loại biến trong JAVA**

Có 3 loại biến

* Biến local(Cục bộ):biến được khai báo bên trong method constructer,hoặc các block.
* Biến local được tạo ra khi mà method,constructer, block được gọi
* Block được gọi và bị hủy khi mà method,constructer được thực hiện xong,
* Không dùng được access modifier và cần khởi tạo giá trị mặc định khi sử dụng
* Biến instance(toàn cục):biến được khai báo bên trong block lớp (bên ngoài constructor,method,block..)
* Biến toàn cục được khởi tạo khi mà đối tượng được khởi tạo thông qua từ khóa "New" và bị hủy khi đối tượng bị hủy.
* Biến static trong java.
* Khai báo bằng từ khóa "static".
* Biết static được tạo khi chương trình bắt đầu chạy và kết thúc khi chương trình dừng.

**20 Trình bày các loại access modifier, và phạm vi truy cập.**

* Private:bên trong class.
* Public:toàn bộ project
* Default:Trong class trong packege
* Protected:trong class,trong packege và ngoài class có mối quan hệ kế thừ

**21 Kế thừa trong JAVA là gì ?**

* Kế thừa là cơ chế cho phép một lớp con kế thừ lại các đặc điểm hành vi đã được định nghĩa ở lớp cha
* Mục đích tái sử dụng mã nguồn.
* Có mối quan hệ is a.
* Lớp con không thể truy cập tài sản private của lớp cha
* Không hỗ trợ đa kế thừa.

**22 Lớp con kế thừa được những tài sản nào(thuộc tính, phương thức) của lớp cha ?**

Lớp con chỉ được phép được sỡ hữu tài sãn (thuộc tín, method) public, default (nếu cha và con cùng package) protected

**23 Lớp Object là gì**

Là lớp gốc của tất cả các lớp trong Java

Các phương thức chính bao gồm:

* toString() => Trả về chuỗi biểu diễn thông tin thuộc tính của đối tượng
* equals() => Để so sánh 2 đối tượng bằng nhau theo tiêu chí nào thuộc tính nào?
* hashCode() => Trả về mã băm

**24 Khái niệm đa hình**

Là khả năng 1 đối tượng có thể hiện / hành vi theo nhiều cách # nhau, tùy thuộc vào ngữ cảnh.

**25 Phân biệt Overloading và Overriding**

* Overriding(): là cơ chế cho phép lớp con định nghĩa lại nội dung các phương thức đã có trước ở lớp cha
* Phương thức overriding phải cùng tên ,cùng danh sách tham số,cùng kiểu trả về
* Access modifier ở phương thức lớp con phải cao hơn hoặc bằng phương thức ở lớp cha
* Xảy ra trong class có mối quan hệ kế thừa
* Đa hình tại rumtime
* Overloading(nạp chồng):Cơ chế cho phép 1 lớp định nghĩa ra nhiều phương thức cùng tên nhưng khác nhau tham số truyền vào

Đa hình tại compile

**26 Ép kiểu là gì ? Các loại ép kiểu**

* **Ép kiểu trong java** là việc gán giá trị của một biến có kiểu dữ liệu này tới biến khác có kiểu dữ liệu khác.
* Ép kiểu tường minh: coder phải chỉ rõ kiểu dữ liệu cần ép

Thu hẹp khả năng lưu trữ có thể gây mất thông tin

* Ép kiểu ngầm định: trình biên dịch (compiler) sẽ tự **động chuyển đổi**từ một kiểu dữ0 liệu này sang kiểu dữ liệu khác **(kiểu dữ liệu cơ sở)**.

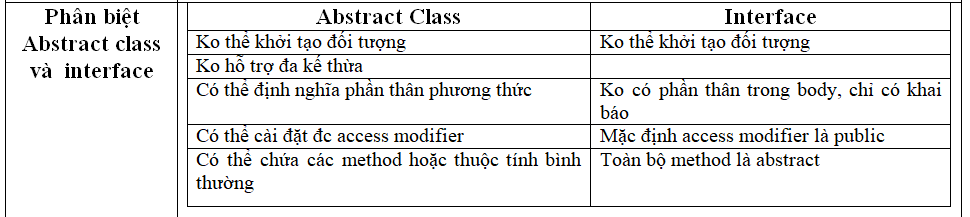
Mở rộng khả năng lưu trữ đảm bảo không làm mất thông tin

**27 Tính trừu tượng là gì ?**

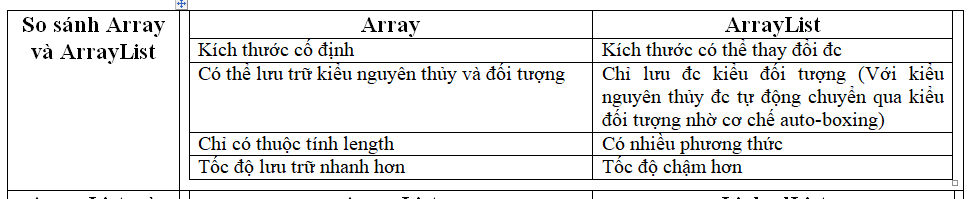
Tính trừu tượng là khả năng ẩn đi các chi tiết của trình triển khai, chỉ hiễn thị tính năng người dùng, chỉ quan tâm cách thực hiện

Trong java trình trừu tượng được thể hiện qua abstract class và interface

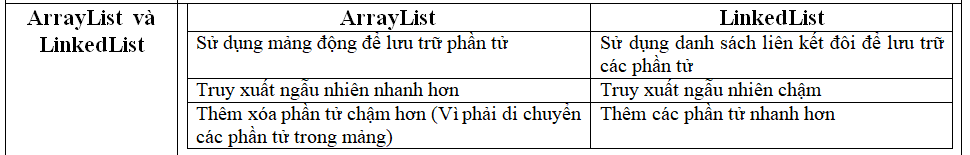
**28 Phân biệt abstract class và interface**



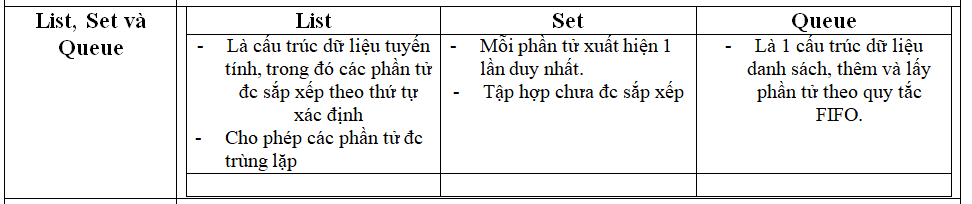
**29 So sánh Array và ArrayList**



**30 So sánh ArrayList và LinkedList**



**31 So sánh List, Set và Queue ?**



**32 Set là gì, các lớp triển khai của Set**

**Set** là một interface kế thừa Collection interface trong java. Set trong java là một Collection không thể chứa các phần tử trùng lặp.

Set được triển khai bởi Hashset, LinkedHashset, Treeset

**HashSet** lưu trữ các phần tử của nó trong bảng băm, là cách thực hiện tốt nhất, tuy nhiên nó không đảm bảo về thứ tự các phần tử được 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).

**33 Generic là gì**

Thuật ngữ “**generics**” được hiểu là tham số hóa kiểu dữ liệu. Cho phép chúng ta tạo ra các class, method, interface hoạt động với nhiều kiểu dữ liệu khác nhau.

**34 Ưu điểm và hạn chế khi dùng generic ?**

**ƯU ĐIỂM:**  
- Kiểu dữ liệu an toàn: Chúng ta chỉ có thể giữ được một loại đối tượng trong Generics. Nó không cho phép lưu trữ các loại đối tượng khác.  
- Kiểm tra dữ liệu chặt chẽ ở Compile-time mà không phải là Runtime-error. Nên chúng ta sẽ dễ dàng kiểm soát lỗi hơn.  
- Hạn chế việc ép kiểu (cast) thủ công mà không an toàn.  
- Giúp chúng ta viết các thuật toán được sử dụng nhiều (reusable), dễ dàng thay đổi, an toàn dữ liệu và dễ đọc hơn.  
**NHƯỢC ĐIỂM:  
-** Không thể gọi Generics bằng kiểu dữ liệu nguyên thủy (vd: int,long,double,v..v..) thay vào đó sử dụng các kiểu dữ liệu Object.  
- Không thể tạo các instances của kiểu dữ liệu Generics.  
- Không thể sử dụng static cho Generics.

**35 Stack là gì, các phương thức của stack ?**

**Stack là** một cấu trúc dữ liệu với nguyên lý cơ bản **là** vào sau ra trước (Last **in** First **out**), ngoài ra **Stack** class cũng cung cấp các method để thao tác với **Stack** như pop(), push(), isEmpty() etc.

**36 Queue là gì, các class triển khai của queue ?**

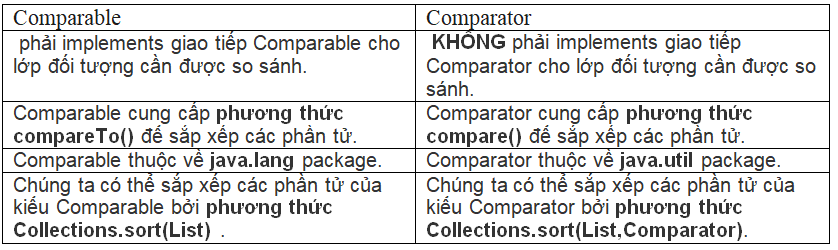
**Queue** được biết đến **là** một dạng cấu trúc dữ liệu hàng đợi, dùng lưu trữ các phần tử theo thứ tự.Cơ chế hoạt động của Queue là FIFO (First in First out) có nghĩa là phần tử nào vào đầu tiên thì sẽ được lấy ra đầu tiên.  
**Các class triển khai Queue:  
-** Linkedlist  
- PriorityQueue  
-ArrayDeque

**37 Phương thức cơ bản của queue ?**

Các phương thức cơ bản:

* Add() : thêm mới thành công trả về true, thất bại thì ném ra ngoại lệ.
* Offer() : : thêm mới thành công trả về true, thất bại thì trả về false.
* Element() : lấy phần tử ở đầu hang đợi, ném ra ngoại lệ nếu hàng đợi rỗng.
* Peek() : lấy phần tử ở đầu hang đợi, trả về null nếu hàng đợi rỗng.
* Remove(): xoá phần tử ở đầu hàng đợi, ném ra ngoại lệ nếu hàng đợi rỗng.
* Poll(): lấy phần tử đầu hàng đợi và xoá nó đi, trả về null nếu hàng đợi rỗng.

**38 So sánh Comparable và Comparator, khi nào dùng cái nào ?**



* Comparable:

Interface Comparable trong Java được sử dụng để sắp xếp các đối tượng của lớp do người dùng định nghĩa (user-defined). Giao diện này thuộc về gói java.lang. Để sử dụng được Comparable, ta phải implements interface Comparable cho lớp đối tượng cần được so sánh.

* Comparator:

Giao diện Comparator trong Java được sử dụng để sắp xếp các đối tượng của lớp do người dùng định nghĩa (user-defined). Để sử dụng Comparator ta không cần phải implements Comparator cho lớp đối tượng cần được so sánh.

**39 Map là gì, các class triển khai**

Map : sử dụng để lưu trữ và truy xuất theo cặp khoá (key) và giá trị (value). Mỗi cặp key value gọi là Entry, Map không cho phép 2 key trùng nhau, mỗi key tương ứng với 1 value.

* HashMap.
* LinkedHashMap.
* TreeMap.

**40 Phương thức equals() và hashcode() trong Java là gì?**

Phương thức equal(): so sánh hai chuỗi đưa ra dựa trên nội dung của chuỗi. Nếu hai chuỗi khác nhau nó trả về false. Nếu hai chuỗi bằng nhau nó trả về true.

Phương thức hashCode(): trả về mã băm của đối tượng

**41 Cây nhị phân là gì**

Cây nhị phân là một cấu trúc dữ liệu được sử dụng cho mục đích lưu trữ dữ liệu. Một cây nhị phân bao gồm các Node và mỗi Node bao gồm 3 thành phần:

* **Data**: Giá trị của một phần tử
* **Left pointer**: Con trỏ trỏ đến cây nhị phân bên trái Node.
* **Right pointer**: Con trỏ trỏ đến cây nhị phân bên phải Node.

Các thành phần cơ bản của cây nhị phân bao gồm:

* **Root**: Được gọi là Node gốc của cây (là một Node cha), một cây chỉ có một Node gốc duy nhất và nó không có Node cha nào.
* **Parent Node**: Là Node cha của một Node cụ thể nào đó.
* **Child Node**: Là Node con của một Node cụ thể nào đó.
* **Sub-tree**: Là cây con biểu diển các con của một Node.
* **LeafNode**: Là Node không có Node con.
* **Siblings**: Các Node có cùng một cha.
* **Internal Node**: Node có ít nhất một Node con.
* **External Node**: Node không có Node con nào.

**42 Trình bày các cách duyệt cây nhị phân.**

Có 3 cách duyệt cây nhị phân:

* Duyệt tiền tự (NLR): duyệt nút gốc, duyệt tiền tự cây con trái, duyệt tiền tự cây con phải.
* Duyệt trung tự (LNR): duyệt trung tự cây con trái, duyệt nút gốc, duyệt trung tự cây con phải.
* Duyệt hậu tự (LRN): duyệt hậu tự cây con trái, duyệt hậu tự cây con phải, duyệt nút gốc.

**43 Ngoại lệ (Exception là gì) ?**

* Exception: là một sự kiện bất thường xảy ra trong quá trình thực thi của 1 chương trình thậm chí gây chết chương trình

**44 Phân loại Exception**

* Check Exception: là loại exception xảy ra trong lúc compile time bắt buộc lập trình viên phải handle
* Uncheck Exception: loại exception xảy ra tại thời điểm thực thi chương trình lỗi này không chắc chắn xãy ra có thể bỏ qua trong quá trình compile lập trình viên không cần bắt buộc phải xử lý

**45 Phân biệt Error và Exception**

* Exception: là một sự kiện bất thường xảy ra trong quá trình thực thi của 1 chương trình thậm chí gây chết chương trình
* Error: là những lỗi nghiêm trọng liên quan đến môi trường thực thi (JVM) của ứng dụng hệ thống mà lập trình viên không thể xử lý nó có thể gây chết cả chương trình



**46 Có bao nhiêu cách để xử lý ngoại lệ (handle)**

* Try/catch – finally: xử lý ngoại lệ ngay tại đoạn mã bị lỗi
* Dùng throw/throws dùng để ném ngoại lệ cho method/logic khác xử

**47 Một số lưu ý khi dùng try-catch**

Lưu ý:

* Khối try có thể không có khối catch hoặc có nhiều khối catch nhưng chỉ có 1 khối finally
* Tại mỗi thời điểm thì xãy ra 1 ngoại lệ và tại thời điểm chỉ có môt khối catch được thực thi
* Khối catch phải được sắp xếp từ cụ thể đến chung nhất ( từ exception con đến exception cha)
* Khối finally luôn được thực thi dù chương trình có xãy ra ngoại lệ hạy không

**48 Serialization là gì ? lưu ý khí sử dụng Serialization**

Là cơ chế tuần tự hóa cho phép chuyển trạng thái của 1 đối tượng (object) vào 1 byte stream. Ngược lại chuyển đổi từ byte stream về object được gọi Desrilization

Lưu ý:

* Class cha implement interface seralizable thì các lớp con không cần implement nữa
* Thuộc tính áp dụng từ khóa stactic, transient thì sẽ không được serialization
* Thuộc tính của đối tượng phải implement serializable

**49 SOLID là gì? Phát biểu của từng nguyên lý**

SOLID là những nguyên tắc ( chỉ dẫn) để giúp ta xây dựng các ứng dụng OOP hiệu quả. Giúp cho chương trình viết ra những đoạn code dễ đọc dễ đọc dễ hiểu và dễ maintain

* Single responsibility: Mỗi lớp chỉ đảm bảo 1 nhiệm vụ duy nhất

+ dễ quản lý mã nguông

+ Mỗi lớp chỉ tập trung vào 1 nhiệm vụ

* Open closed principle: có thể mở rộng 1 class nhưng không được sữa chữa bên trong class đó

+ dễ thay đổi

+ dễ mở rộng

* Liskov substitiution: trong 1 chương trình interface lớn ta nên tách nhiều interface nhỏ với nhiều mục đích cụ thể

+ tránh dư thừa mã nguồn không cần thiết

* Dependency inversion principle: các module cấp cao không nên phụ thuộc vào các module các thấp cả 2 nên phụ thuộc vào abstraction, Interface(abstraction) không nên phụ thuộc vào chi tiết và ngược lại

**50 Design pattern là gì? Chia làm bao nhiêu loại. Tổng số là bao nhiêu?**

* DP ( Mẫu thiết kế) là một giải pháp tổng thể cho các vấn đề chung trong thiết kế phần mềm
* DP không phải là ngôn ngữ lập trình cụ thể nào, được sử dụng phổ biến trong lập trình OOP

Lợi ích:

* Tăng tốc độ phát triển phần mềm
* Hạn chế rũi ro tiềm ẩn
* Tái sử dụng soure code
* Giúp đọc code dễ hơn
* Lập trình viên làm nhóm cũng giao tiếp thuận lợi hơn vì có được tiếng nói chung

DP chia làm 3 loại:

* Nhóm Creational (khởi tạo): sử dụng để giải quyết các đề thường gặp đối với việc khởi tạo đối tượng
* Nhóm structural (cấu trúc): sử dụng để thiết kế các thành phần của lớp, và đối tượng giúp để nhận ra các mối quan hệ giữa các thực thể
* Nhóm behavioral (hành vi tương tác): sử dụng để giải quyết các vấn đề giao tiếp giưã các đối tượng