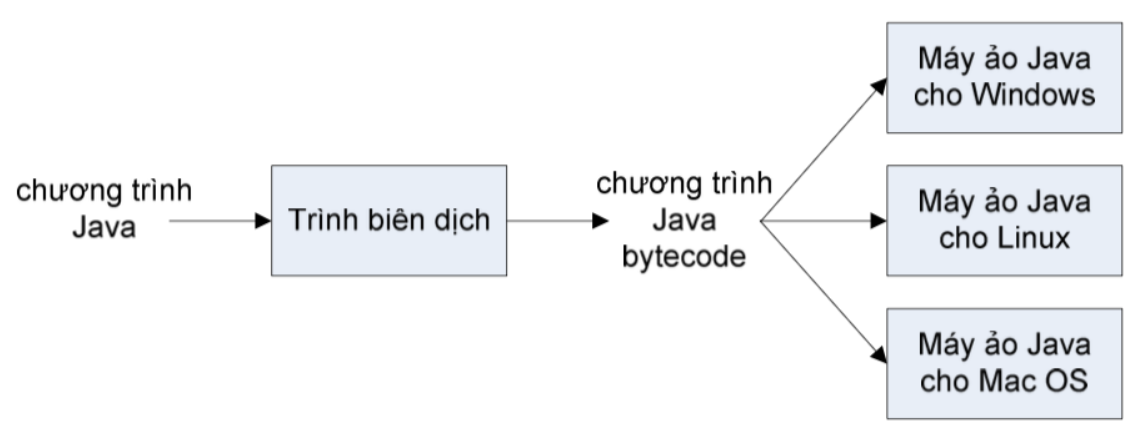
1. **Java là gì?**
   * Là một ngôn ngữ lập trình hướng đối tượng
   * Có khả năng thực thi ở nhiều loại thiết bị
   * Được sử dụng rộng rãi
   * Có tính độc lập nền tảng
   * Java có thể chạy trên các nền tảng khác nhau mà không cần phải biên dịch lại
   * Viết 1 lần chạy mọi nơi
2. **JDK , JRE , JVM**

JDK (Java Development Kit – Bộ công cụ phát triển Java) : Giúp phát triển và biên dịch mã java thành chương trình java, gồm có trình biên dịch, trình thông dịch, trình giúp sửa lỗi

JRE (Java Runtime Environment) [ môi trường chạy java ]: Giúp thực thi các chương trình java trong JVM , JRE cũng được kèm theo trong JDK

JVM (Java Virtual Machine ): Các chương trình viết bằng Java được biên dịch thành mã máy, nhưng đây là loại ngôn ngữ máy dành cho loại máy tính không tồn tại – loại máy "ảo" này được gọi là Máy ảo Java .  Ngôn ngữ máy dành cho máy ảo Java được gọi là Java bytecode,



|  |
| --- |
| 1. **Thông dịch, biên dịch khác nhau như thế nào ?**   Trình biên dịch là trình dịch chuyển đổi ngôn ngữ nguồn (ngôn ngữ cấp cao) thành ngôn ngữ đối tượng (ngôn ngữ máy). Ngược lại với trình biên dịch, trình thông dịch là một chương trình bắt chước việc thực thi các chương trình được viết bằng ngôn ngữ nguồn. Một điểm khác biệt giữa Trình biên dịch và trình thông dịch là Trình biên dịch chuyển đổi toàn bộ chương trình trong một lần, mặt khác Trình thông dịch chuyển đổi chương trình bằng cách thực hiện một dòng tại một thời điểm.  Thông dịch đòi hỏi ít bộ nhớ hơn vì k tạo mã đối tượng trung gian. |
| 1. **JAVA là ngôn ngữ thông dịch hay biên dịch ?**   Vừa thông vừa biên |
| 1. **Các kiểu dữ liệu trong java, và giá trị mặc định khi khai báo.**   Java có 2 loại kiểu dữ liệu :  + Kiểu dữ liệu nguyên thủy : boolean(false) , char('\u0000') , byte(0) , short 0 , int 0, long 0L, float 0.0f, double 0.0d  + Kiểu dữ liệu tham chiếu : String, array -null |
| 1. **Có bao nhiêu lại mệnh đề if**   If, if else, else if, toán tử 3 ngôi(if else rút gọn):  variable = (condition) ? expressionTrue : expressionFalse; |
| 1. **So sánh if và switch-case**   If : Kiểm tra 100 điều kiện để thực hiện 100 trường hợp khác nhau  - Switch-case : kiểm tra 1 biến để thực hiện 100 trường hợp  -> Nên sử dụng if ở trường hợp ít điều kiện còn switch case ngược lại |
| 1. **Khi nào dùng for, while, do-while ?**   For : khi biết trước số lần lặp  - While : chưa biết trước số lần lặp  - Do while : thực hiện 1 lần rồi mới lặp |
| 1. **Phân biệt for-i và for-each**   For-index : sử dụng khi biết trước số vòng lặp, ở vị trí theo điều kiện  - For-each : duyệt tất cả các phần tử có trong mảng, từ đầu tới cuối(khuyến khích khi duyệt mảng) |
| 1. **Khác nhau giữa while, do-while. Cho ví dụ khi nào dùng ?**   While : lặp với số lần chưa biết trước.Các lần lặp sẽ không xảy ra nếu điều kiện sai  - Do-while : xảy ra ít nhất 1 lần nếu điều kiện sai ở lần đầu tiên  VD : |
| 1. **Break, Continue có tác dụng gì trong mệnh đề lặp ?** |
| Break : thoát khỏi vòng lặp hiện tại chứa break  - Continue : bỏ qa các câu lệnh thực thi để qua điều kiện khác |
| 1. **Trình bày các cách khởi tạo một mảng trong JAVA**   Khai báo mảng mảng có 100 ptu và chưa gán : int[] arr = new int[100]  -Khai báo mảng với ptu cụ thể : String[] array = {"A","B"}; |
|  |
| 1. **Phần tử của mảng có thể dùng kiểu dữ liệu nào, và có giá trị mặc định là gì ?**   Giá trị mặc định của mảng khi k khai báo là null;  - Phần tử của mảng có thể dùng kiểu dữ liệu nguyên thủy hoặc object |
| 1. **OOP là gì ?**   Object Oriented Programming : lập trình hướng đối tượng  Trừu tượng (abstraction)  Bao Đóng (Encapsulation)  Kế thừa (Inherritance)  Đa hình (Polimophysm) |
| 1. **Phân biệt class và object**   Class : Lớp là sự trừu tượng hóa của đối tượng. Những đối tượng có những đặc tính tương tự nhau sẽ được tập hợp thành một lớp. Lớp cũng sẽ bao gồm 2 thông tin là thuộc tính và phương thức.  - Object : là một thể hiện của class. Lớp là một mẫu hoặc thiết kế từ đó các đối tượng được tạo ra |
| 1. **Constructor là gì**   Constructor - hàm khởi tạo:  là phương thức đặc biết để khởi tạo đối tượng.  - |
| 1. **Cách khai báo constructor và đặc điểm constructor trong JAVA**   TenClass(int i, String n) {  id = i;  name = n;  }  Đặc điểm : có tên hàm trùng với tên class , không có kiểu dữ liệu trả về , thuộc tính accessmodifier là public  + Mỗi class sẽ có 1 constructor mặc định là constructor không tham số, nhưng nếu mình có khai báo constructor  thì constructor không tham số kia sẽ bị xóa đi.  + Có thể tạo nhiều constructor trong một class  Constructor có thể gọi cons khác nhưng lệnh gọi phải đứng đầu. this(tham số). |
| 1. **Phân biệt constructor và method**   Constructor : không có giá trị trả về , constructor luôn là public , có thể có nhiều constructor trùng tên  Method : có giá trị trả về , method có thể là public , private , protected or default , method cũng có thể có nhiều method trùng tên nhưng phải khác tham số ( hàm overload ) |
| 1. **Tính bao đóng là gì ? Làm sao để thu được tính bao đóng trong java ?** |
| Encapsulation  - Làm sao để thu được tính bao đóng : Sử dụng phương thức getter setter để thiếp lập và lấy dữ liệu trong nó.  - Dữ liệu trong lớp private  - Tính bao đóng : nhằm mục đích che dấu sự tác động trực tiếp mà phải thông qa một phương thức một cách nào  đó của mình ví dụ như là getter và setter |
| 1. **Tham trị, tham chiếu**   -Tham trị : tham số khai báo kiểu nguyên thủy gồm byte short int long float double boolean char, 2 biến có tên khác nhau sẽ có 2 bộ nhớ khác nhau, nó sẽ copy giá trị từ địa chỉ bộ nhớ này sang địa chỉ bộ nhớ khác chứ không truyền bản thân địa chỉ của biến trong bộ nhớ.  -Tham chiếu : nó lưu trữ bên trong 1 địa chỉ ( không phải là giá trị ) mà địa chỉ đó sẽ dùng để truy cập bộ nhớ khi lưu / lấy dữ liệu (giá trị) của biến ( chính là các đối tượng )  -Tham trị : truyền giá trị vào 1 hàm, khi đi qua hàm đó thì giá trị sẽ không thay đổi  - Tham chiếu : truyền giá trị vào 1 hàm, khi đi qua hàm đó thì giá trị sẽ bị thay đổi theo như trong hàm |
| 1. **Từ khóa static dùng để làm gì ?**   -Là 1 từ khóa khai báo thuộc tính, phương thức, khối lệnh, class(class nằm trong class) và chịu sự qly của class  -Truy xuất các thành phần qua tên class hoặc đối tượng class.  Mục đích: định nghĩa thuộc tính, phương thức chung của các đối tượng, tạo các class tiện ích.(chỉ chứa pthuc và thuộc tính static). |
| 1. **Ràng buộc khi sử dụng static**   - Trong phương thức static thì chỉ được phép gọi biến static hoặc là phương thức static , còn những pthuc khác muốn gọi thì phải thông qa đối tượng ( khởi tạo đối tượng và new đối tượng đó ra mới có thể gọi được )  Chỉ có thể khởi tạo giá trị cho thuộc tính static qua khối lệnh hoặc gán tr tiếp khi khai báo. |
| 1. **Các loại biến trong JAVA**   - Biến cục bộ : được khai báo trong phương thức , hàm constructor hoặc trong các block. Không được sd access modifier khi khai báo biến cục bộ. Bạn phải khai báo giá trị mặc định cho biến cục bộ trước khi sử dụng  - Biến toàn cục : được khai báo trong 1 class , bên ngoài các phương thức , constructor và các block.  - Biến static : Khai báo class với từ khóa "static" phía bên ngoài các phương thức , constructor và block. |
| 1. **Trình bày các loại access modifier, và phạm vi truy cập.**   private -> default -> protected -> public. Trong lớp-trong package-lớp con-công cộng |
| 1. **Kế thừa trong JAVA là gì ?**   - (Inheritance) Cho phép 1 lớp con có thể sử dụng lại thuộc tính và phương thức đã được định nghĩa ở trong lớp cha  - Nếu lớp cha không muốn lớp con kế thừa thì sd thuộc tính final  - Mối quan hệ is-a. |
| 1. **Lớp con kế thừa được những tài sản nào(thuộc tính, phương thức) của lớp cha ?**   - Kế thừa tất cả những gì mà lớp cha có  - Trừ khi thuộc tính hay phương thức có từ khóa final, private, và constructor. |
| 1. **Lớp Object là gì**   Là lớp gốc tất cả các class |
| 1. **Khái niệm đa hình**   Polymorphism  - là một phương thức có thể thực hiện bằng nhiều cách khác nhau  - Là khả năng một đối tượng có thể hiện/hành vi theo nhiều cách khác nhau tùy thuộc vào ngữ cảnh.  - Trong java tính đa hình thể hiện qua ghi đè(override, mối quan hệ is -a, lớp con đè phương thức lớp cha) và nạp chồng phương thức(over load, lớp con định nghĩa đc nhiều phương thức cùng yên nhưng khác tham số tr vào).  - Ví dụ: tất cả mọi người đều đi chơi, người thì chơi game, người thì chơi đá bóng,.. |
| 1. **Phân biệt Overloading và Overriding**   - Overloading (nap chong) : cùng lớp , cùng tên , khác tham số. Kiểu dữ liệu trả về khác nhau được  - Override (ghi de ): khác lớp , cùng tên , cùng tham số, cùng kiểu dl trả về.  Kiểu dữ liệu trả về có thể khác nhau nếu kdl trả về của lớp con là con của kdl của lớp cha  Class cha kiểu đối tượng trả về là A , class con sẽ override class cha nhưng nó trả về là kdl là B thì trong trường hợp nào trả về B là đc , trong trường hợp B là con của A thì mới trả về được , nếu không thì sẽ không được. Nếu cha trả về kiểu long thì con có thể trả về kiểu int cx được. |
| 1. **ép kiểu là gì ? các loại ép kiểu**   - Có 2 loại ép kiểu : ngầm định và tường minh  - Ngầm định : hệ thống sẽ tự ép kiểu cho chúng ta  int a = 5;  long b = a;  - Tường minh : chúng ta phải tự đi ép kiểu cho nó nếu cùng kiểu số: float->int  long a = 6;  int b = (int) a; |
| 1. **Tính trừu tượng là gì ?**   - Abstraction  - tính chất khả năng ẩn di các chi tiết không cần thiết và chỉ hiển thị và khai báo các chi tiết cần thiết cho người dùng.  - Chỉ quan tâm đến kết quả đạt được mà không quan tâm đến quá trình thực hiện(khái niệm của abstract method)  - Trừu tượng nó lên , không biết cách nào để diễn đạt nó  - Là một method không biết nó làm gì , chỉ biết tên method làm gì thôi.  - Thể hiện qua abstract method có trong abstract class và interface.  - Nó sẽ bỏ qa những thứ không cần thiết |
| 1. **Phân biệt abstract class và abstract interface**   -Abstract class: class trừu tượng tới mức k thể tạo đtg. Class chứa abstract method chắc chắn là abstract class, ko dùng final. Có thể extends từ abstract class khác ko cần override.  -Interface: là 1 bản thiết kế quy định hành vi lớp triển khai(sd implements)(has-a, can-do). Không thể tạo đối tượng, thuộc tính mặc định là hằng số, k có constructor.  So sánh: Interface: 1 class triển khai đc nhiều interface, toàn bộ là abstract method(ngoại trừ khai báo static & dèfault), toàn bộ thuộc tính là hằng số.  Abstract class: 1 class chỉ kế thừa 1 abstract class, có method thường và abstract. Thuộc tính bthg |
| 1. **So sánh Array và ArrayList**   Array: kích thước cố định, lưu trữ đc dl ng thủy & đối tg, chỉ có thuộc tính length  Arrlist:có thể thay đổi kthc, chỉ lưu trữ dl đối tg(sd ng thủy dùng wrapper class cơ chế auto boxing), có nhiều pthuc. |
| 1. **So sánh ArrayList và LinkedList**   ArrList: sd mảng lưu trữ ptu, truy xuất & sửa ngẫu nhiên nhanh hơn  LinkedList: sd dsach lket, thêm xóa nhanh hơn |
| 1. **Set là gì, các lớp triển khai của Set**   Set là interface  - một tập hợp các đối tượng không bị trùng nhau  - các lớp triển khai : hashset , treeset |
| 1. **Generic là gì**   Là cơ chế cho phép sd kiểu dl như là tham số, cho phép tạo class, method, interface hd trên nhiều kiểu dl. |
| 1. **Ưu điểm và hạn chế khi dùng generic ?**   Ưu: phát hiện lỗi ngay tại khi compile; không cần ép kiểu; xây dựng thuật toán tổng quát, tái sd mã nguồn  Hạn chế: không thể gọi generic ở kiểu nguyên thủy |
| 1. **Stack là gì, các phương thức của stack ?**   - là một cấu trúc dữ liệu để lưu trữ danh sách , nó sẽ thực hiện first in last out (FILO)  - có những phương thức : add(): thêm ptu, trả vè true , peek() : trả về phần tử trên cùng nhưng không xóa , pop() ; xóa ptu trên cùng  push() : thêm ptu vào trên cùng, trả về đối tượng vừa thêm , search() : tìm kiếm |
| 1. **Queue là gì, các class triển khai của queue ?**   - Queue được sử dụng để lưu trữ các phần tử theo một thứ tự, nó cũng tương tự như List nhưng cơ chế hoạt động của Queue là First In First Out |
| 1. **Phương thức cơ bản của queue ?**   peek(ấy phần tử ở đầu hàng đợi, mà không xóa phần tử này.) isFull() isEmpty() enqueue(thêm vào cuối) dequeue(lấy ở đầu) |
| 1. **So sánh Comparable và Comparator, khi nào dùng cái nào ?**   - Comparerable : định nghĩa là compareTo(); định nghĩa 1 object có khả năng so sánh đối tượng của nó. VD compare lớp Student thì Student có thể so sánh với nhau được, khi phương thức so sánh mình làm nhiều lần  - Comparerator : định nghĩa là compare(); định nghĩa một phương thức so sánh ( không phải trong lớp ) , so sánh phương thức khác thì dùng comparerator để nó không bị ảnh hưởng |
| 1. **Map là gì, các class triển khai**   - Map là cấu trúc dữ liệu lưu trữ danh sách các cặp key/value  - Map cho phép thực hiện các hành động truy xuất, xoá và cập nhật các cặp key/value rất hiệu quả thông qua việc sử dụng key  - Map không cho phép 2 key trùng lặp  - Mỗi key tương ứng với một value  - Một cặp key-value được gọi là một Entry  - Các class : HashMap , TreeMap , LinkedHashMap |
| 1. **Cây nhị phân là gì**   Binary Tree (Cây nhị phân) là một cấu trúc phân tầng  Bao gồm:  Một nút gốc (root)  Một cây con trái (left subtree)  Một cây con phải (right subtree)  Một nút có thể có 0, 1 hoặc 2 cây con |
| 1. **Trình bày các cách duyệt cây nhị phân.**   - Duyệt theo Node - Left - Right (NLR) : Duyệt node gốc, duyệt node trái, duyệt node phải.  - Duyệt theo Left - Node - Right (LNR) : Xuất ra giá trị tăng dần, duyệt Right - Node - Left (RNL) sẽ cho giá trị giảm dần.  - Duyệt theo Left - Right - Node (LRN) : |
| 1. **Ngoại lệ (Exception là gì) ?**   - là một vấn đề phát sinh trong quá trình thực thi phá vỡ luồng thực thi bình thường, đôi khi gây chết ctrinh.  (thực thi là runtimeException) |
| 1. **Phân loại Exception**   Có 2 loại : unchecked exception : Xảy ra trong qtrinh runtime, lỗi không chắc chắn xảu ra, không bắt buộc xử lý và checked exception xảy ra khi compile, bắt buộc phải xử lý. |
| 1. **Phân biệt Error và Exception**   - Error : lỗi người code không thể can thiệp và không xử lý được  - Exception : mình có thể xử lý được và có thể lường trước được và xử lý |
| 1. **Có bao nhiêu cách để xử lý ngoại lệ (handle)**   - 2 cách : try/cacth : bắt và xử lý ngoại lệ ngay tại chỗ  - throw : dùng để ném ngoại lệ ra dùng để dánh dấu lỗi bắt lỗi  - throws: throws ủy quyền cho class cao hơn xử lý |
| 1. **Một số lưu ý khi dùng try-catch**   -Tại 1 thời điểm có duy nhất 1 khối catch đc thực thi  -1 try/catch có thể có nhiều khối catch nhưng phải sxep từ cụ thể nhất tới chung chung nhất(từ exception con đến cha)  -try bắt buộc có còn catch và finally có 1 trong 2 là được. |

Một khối finally là nơi ta đặt các đoạn mã phải được thực thi bất kể ngoại lệ có xảy ra hay không