**CHECK LIST CÁC CÂU HỎI CƠ BẢN VỀ JAVA**

**1. Câu hỏi cơ bản về Java:**

a. Java ra đời năm nào?

- 1995 Java 1.0 (2014 Java-8)

b. Giới thiệu đơn giản về JVM, JRE, JDK.

JVM thực thi mã nguồn Java trên máy tính.

JRE cung cấp môi trường để chạy các ứng dụng Java đã biên dịch.

JDK là bộ công cụ hoàn chỉnh cho việc phát triển ứng dụng Java, bao gồm JRE và các công cụ phát triển.

**2. Biến, mảng trong Java:**

a. Biến trong Java là gì?

Biến là một vùng lưu trữ trong bộ nhớ được sử dụng để lưu giữ giá trị

b. Mảng trong Java là gì? Cách khai báo một mảng trong Java?

Mảng là một cấu trúc dữ liệu cho phép lưu trữ một danh sách các phần tử cùng kiểu dữ liệu.

**3. Các kiểu dữ liệu trong Java:**

a. Liệt kê các kiểu dữ liệu nguyên thủy (primitive) trong Java.

Int, short, long, byte, float, double, char, boolean

b. Giải thích khái niệm kiểu Wrapper trong Java và đưa ra ví dụ.

Đóng gói primitive thành đối tượng

c. So sánh primitive với wrapper.

Primitive kiểu dữ liệu thô,ko có thuộc tính hay phương thức

Wrapper có thuộc tính và phương thức

**4. String, StringBuilder, StringBuffer:**

a. Tại sao String là bất biến trong Java?

String được gọi là bất biến (immutable) bởi vì một khi nó được tạo ra, giá trị của nó không thể thay đổi.

(Khi thay đổi giá trị của String - > Tạo ra String mới)

An toàn, tối ưu memory

b. Làm thế nào để lưu trữ một đối tượng String trong bộ nhớ?

**- Sử dụng toán tử gán (=)**

String text = "Hello, world!";

* **Sử dụng constructor của String**

String name = new String("John");

c. Sự khác nhau giữa String s = "abc"; và String s = new String("abc");?

Câu b

d. So sánh sử dụng String, StringBuilder và StringBuffer trong Java.

Sử dụng String khi bạn cần làm việc với chuỗi không thay đổi.

Sử dụng StringBuilder khi bạn cần làm việc với chuỗi biến đổi và không cần đồng bộ hóa.

Sử dụng StringBuffer khi bạn cần làm việc với chuỗi biến đổi trong môi trường đa luồng và cần đồng bộ hóa.

**5. Ép kiểu:**

a. Khái niệm ép kiểu dữ liệu trong Java là gì? Đưa ra ví dụ.

Ép kiểu có thể được thực hiện khi bạn muốn chuyển đổi giá trị từ một kiểu dữ liệu này sang một kiểu dữ liệu khác

int intValue = 42;

double doubleValue = intValue; // Ép kiểu ngầm định

b. Khái niệm ép kiểu đối tượng trong Java là gì? Đưa ra ví dụ.

Ép kiểu đối tượng trong Java là quá trình chuyển đổi giữa các kiểu đối tượng khác nhau, trong trường hợp một lớp là lớp con của lớp khác

Animal animal = new Dog() --Tạo đối tượng Dog và gắn vào biến tham chiếu Animal

Ép kiểu từ Animal sang Dog để gọi phương thức của lớp con Dog

Dog dog = (Dog) animal;

**6. Constructor:**

a. Constructor là gì trong Java?

Contructor là một phương thức đặc biệt trong một lớp được sử dụng để khởi tạo các đối tượng của lớp đó.

b. Có mấy loại constructor?

Không tham số

Có tham số truyền vào

c. Cách nào để tạo constructor nhanh trong IDE Eclipse hoặc Intellij?

- Chuột phải và chọn Generate và chọn Constructor

**7. Access modifier:**

a. Liệt kê và giải thích các phạm vi truy cập (access modifier) trong Java: private, default, protected, public.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Private | Default | Protected | Public |
| Phạm vi variable, medthod | Cùng 1 lớp | Cùng package | -Cùng package  -Lớp con của package khác | Bất cứ nơi nào trong java |

b. Khi nào chúng ta sử dụng private, default, protected và public trong việc khai báo các thành viên lớp?

Mỗi phạm vi có mục đích sử dụng và quyền truy cập khác nhau.

**8. Non-access modifier:**

a. Liệt kê và giải thích các non-access modifier: abstract, static, final, synchronized, transient, volatile (khi kết hợp với field, method hoặc class).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Abstract | Static | Final | Synchronized | Transient | Volatile |
| Trường |  | Trở thành một trường tĩnh thuộc lớp, chia sẻ giữa tất cả các đối tượng của lớp. | Trở thành một trường không thể thay đổi, giá trị không thể được gán lại sau khi khởi tạo. |  | Trường đó không tham gia quá trình serialization khi đối tượng được lưu trữ hoặc truyền qua mạng. | Trường đó chỉ được đọc và ghi từ bộ nhớ chính (main memory), đảm bảo tính nhất quán giữa các luồng. |
| Phương thức | Chỉ có khai báo và không có phần thân. | Trở thành một phương thức tĩnh của lớp, có thể gọi mà không cần tạo đối tượng. | Không thể bị ghi đè trong các lớp con. | Đảm bảo rằng chỉ có một luồng có thể thực hiện phương thức hoặc khối đó tại một thời điểm. |  |  |
| Lớp | Lớp trừu tượng và không thể được khởi tạo. |  | Trở thành một lớp không thể được kế thừa |  |  |  |
| Chú ý | Lớp trừu tượng tạo một lược đồ cho các lớp con triển khai (implement) các phương thức trừu tượng | Không thể truy cập các trường không tĩnh từ một phương thức tĩnh và ngược lại. |  | Sử dụng để đồng bộ hóa truy cập đối với tài nguyên có thể được truy cập đồng thời bởi nhiều luồng. | Thường được sử dụng cho các trường tạm thời không cần được lưu trữ theo cách thông thường. | Thường được sử dụng trong tình huống đồng thời khi nhiều luồng cùng truy cập một biến. |

**9. Exception và xử lý exception:**

a. Khái niệm về exception trong Java là gì?

Exception trong Java là một sự kiện xảy ra trong quá trình thực thi chương trình, khi có một tình huống bất thường xảy ra mà có thể ảnh hưởng đến luồng thực thi bình thường.

b. Làm thế nào để xử lý exception trong Java?

Dùng try – catch

**10. Abstract và Interface:**

a. Khái niệm abstract class và abstract method trong Java là gì?

Câu 8

b. Khái niệm interface trong Java là gì?

Nó giúp thực hiện khái niệm hướng đối tượng mục tiêu (object-oriented programming) như kế thừa đa dạng và đa kế thừa.

c. Sự khác nhau giữa abstract class và interface trong Java.

Abstract class có thể chứa cả phương thức trừu tượng và phương thức đã triển khai.

Interface chỉ chứa các phương thức trừu tượng mà các lớp triển khai (implement) phải cung cấp định nghĩa.

**11. OOP trong lập trình:**

a. Khái niệm OOP (Object-Oriented Programming) trong lập trình là gì?

- Là một phương pháp lập trình dựa trên khái niệm về đối tượng

- Trong OOP, mọi thứ trong chương trình được xem như là các đối tượng hoặc các thể hiện của các lớp

b. Tại sao OOP được coi là một phương pháp lập trình mạnh mẽ?

OOP giúp tạo ra mã nguồn có cấu trúc, dễ quản lý, dễ bảo trì và tái sử dụng

**12. Đóng gói trong Java:**

a. Java triển khai tính đóng gói như thế nào?

Phạm vi truy cập( private, protected, default, public) xem câu trên

b. Getter và Setter trong Java được sử dụng để làm gì?

Getter là một phương thức được sử dụng để truy cập giá trị của biến thành viên từ bên ngoài lớp. Thông thường, getter sẽ trả về giá trị của biến thành viên mà nó liên quan đến.

Setter Method**:** Setter là một phương thức được sử dụng để thiết lập giá trị cho biến thành viên từ bên ngoài lớp. Thông thường, setter sẽ kiểm tra và xác thực giá trị trước khi gán cho biến thành viên.

**13. Kế thừa trong Java:**

a. Khái niệm kế thừa trong Java là gì?

Nó cho phép một lớp (class) mới có thể sử dụng các thuộc tính (biến) và phương thức (hàm) của lớp đã tồn tại (lớp cha hoặc lớp cơ sở)

b. Từ khoá this và super trong Java được sử dụng để làm gì?

This là một tham chiếu (reference) đến hiện thực (instance) của lớp hiện tại (đối tượng hiện tại).

Super để truy cập đến các thành viên của lớp cha hoặc để gọi hàm khởi tạo của lớp cha

c. Liệt kê các loại kế thừa. Java có đa kế thừa hay không? Tại sao?

Đây là loại kế thừa mà một lớp kế thừa tất cả thuộc tính và phương thức từ một lớp cha duy nhất.

Đây là loại kế thừa mà một lớp con có thể triển khai nhiều giao diện. Một lớp có thể triển khai nhiều interface

Java không hỗ trợ đa kế thừa**,** Để tránh các vấn đề liên quan đến xung đột và không rõ lựa chọn khi một lớp con kế thừa từ nhiều lớp cha có cùng tên phương thức hoặc biến.

d. Sự khác nhau giữa đơn kế thừa, kế thừa nhiều cấp và kế thừa phân cấp trong Java.

Đơn kế thừa xảy ra khi một lớp con chỉ kế thừa từ một lớp cha duy nhất.

Kế thừa nhiều cấp xảy ra khi một lớp con được kế thừa từ một lớp cha và sau đó một lớp khác kế thừa từ lớp con đó.

Kế thừa phân cấp xảy ra khi nhiều lớp con kế thừa từ một lớp cha duy nhất.

**14. Đa hình trong Java:**

a. Đa hình trong compile time (overloading) là gì? Đưa ra ví dụ.

Là một tính năng trong Java cho phép bạn định nghĩa nhiều phương thức cùng tên trong một lớp.

b. Đa hình trong run time (overriding) là gì? Đưa ra ví dụ.

Là một tính năng của lập trình hướng đối tượng trong Java cho phép một lớp con định nghĩa lại phương thức đã được định nghĩa trong lớp cha. Phương thức ở lớp con có cùng tên, cùng kiểu trả về và cùng danh sách tham số với phương thức ở lớp cha.

**15. Trừu tượng trong Java:**

a. Trừu tượng trong java là gì? Thể hiện như thế nào?

Lớp trừu tượng trong java là lớp có hành vi chưa được xác định rõ

VD: đã là hình thì phải có diện tích, chu vi nhưng chưa xác định là hình gì (tam giác, tứ giác,..) nên chưa thể tính

**16. Collection:**

a. Collection trong Java được định nghĩa như thế nào?

Collection là cấu trúc dữ liệu được sử dụng để nắm giữ nhiều phần tử

- Có thể thêm ,sửa,xóa,…các phần tử

- Hợp, giao,… các tập hợp

b. Các thành phần chính của Collection là gì?

- Collection chia làm 2:

- List: là conllection mà mỗi phần tử được xuất hiện nhiều lần và truy xuất bằng chỉ số

- Set: là conllection mà mỗi phần tử được xuất hiện 1 lần và không truy xuất bằng chỉ số

c. Hệ thống cấp bậc của Collection bao gồm những gì?

**17. Iterable và Iterator:**

a. Iterable là gì trong Java? Vai trò của nó là gì?

- Là 1 interface trong java để duyệt qua các phần tử trong 1 tập hợp và trả về 1 đối tượng interator

b. Làm thế nào để sử dụng Iterator để lặp qua các phần tử trong một Collection?

- Tạo ra Interator

- dùng while hoặc for để lặp qua các phần tử

**18. Interface List và các lớp cài đặt:**

a. Interface List trong Java được sử dụng để làm gì?

List interface giúp lưu trữ và truy cập các phần tử như mảng. Nó là interface được kế thừa từ Collection interface.

b. Các lớp cài đặt của Interface List trong Java bao gồm những gì? Hãy nêu một số đặc điểm và sử dụng của mỗi lớp.

-ArrayList: Lớp ArrayList cung cấp cho chúng ta mảng động trong Java. Mặc dù nó có thể chậm hơn so với các mảng tiêu chuẩn nhưng có thể hữu ích trong các chương trình yêu cầu nhiều thao tác trên mảng.

LinkedList: triển khai danh sách bằng cách sử dụng danh sách liên kết đôi, Thích hợp cho các tình huống khi bạn thường xuyên thêm/xóa phần tử ở đầu hoặc cuối danh sách và không cần truy cập ngẫu nhiên nhiều.

-Vector: gần như ArrayList nhưng đc đồng bộ hóa, thích hợp dùng trong đa luồng

- Stack: là lớp con của Vector, thích hợp khi cần triển khai ngăn xếp

c. Tại sao cần nhiều lớp triển khai interface List? Khi nào thì nên dùng triển khai nào? Tại sao?

- Khi triển khai tùy vào hiệu suất, đơn luồng hay đa luồng mà chọn lớp thích hợp

d. Nêu chi tiết hiểu biêt về ArrayList, LinkedList. Nó khởi tạo, lưu trữ, thêm, sửa, xoá, ... các phần tử như thế nào?

ArrayList là dùng một mảng động để lưu trữ phần tử.

LinkedList sử dụng danh sách liên kết để lưu trữ phần tử. Mỗi phần thử có thể được gọi là 1 node trong danh sách.

-ArrayList thêm và xóa phần tử chậm hơn LinkedList: Vì LinkedList chỉ cần thay đổi luồng trỏ của các node trong danh sách còn ArrayList phải tăng/lùi tất cả những vị trí sau vị trí muốn thêm/xóa.

-ArrayList truy xuất phần tử nhanh hơn LinkedList: ArrayList muốn truy xuất đến phân tử thứ mấy trong danh sách thì chỉ cần gọi vị trí đó ra là được còn LinkedList thì phải duyệt qua các phần tử trước đó thì mới truy xuất được đến phần tử cần lấy

-ArrayList yêu cầu ít bộ nhớ hơn LinkedList: Vì ngoài lưu trữ giá trị thì các node trong LinkedList còn phải chứa các tham chiếu đến phần tử trước, sau của nó nữa.

**19. Queue và các lớp cài đặt:**

a. Queue trong Java có ý nghĩa gì? Đặc điểm và ứng dụng của Queue là gì?

-Queue, được hiểu là hàng đợi và là cấu trúc dữ liệu hoạt động theo nguyên tắc: vào trước ra trước (FIFO)

- Đặc điểm: thêm, xóa, check rỗng

b. Các lớp cài đặt của Queue trong Java bao gồm những gì? Hãy mô tả sự khác nhau giữa chúng.

LinkedList:

Sử dụng add để thêm phần tử vào cuối hàng đợi và remove để loại bỏ phần tử đầu tiên.

Hỗ trợ các phương thức peek, poll, và element để xem và loại bỏ phần tử đầu tiên mà không gây lỗi khi hàng đợi rỗng.

ArrayDeque:

Hỗ trợ các phương thức addFirst, addLast, removeFirst, và removeLast để thêm và loại bỏ phần tử ở cả hai đầu hàng đợi.

Hỗ trợ các phương thức peekFirst, peekLast, pollFirst, và pollLast để xem và loại bỏ phần tử ở cả hai đầu mà không gây lỗi khi hàng đợi rỗng.

PriorityQueue:

Phần tử có ưu tiên cao nhất (theo tiêu chí so sánh) sẽ được lấy ra đầu tiên.

Hỗ trợ các phương thức add, remove, và peek.

**20. Set và các lớp cài đặt:**

a. Set trong Java là gì? Đặc điểm và ứng dụng của Set là gì?

-Set là collection mà mỗi phần tử chỉ được phép xuất hiện 1 lần và không được phép truy xuất theo chỉ số

-Ứng dụng: Loại bỏ trùng lặp,…

b. Các lớp cài đặt của Set trong Java bao gồm những gì? Hãy mô tả sự khác nhau giữa chúng.

-HashSet: Hỗ trợ thao tác nhanh chóng, nhưng không đảm bảo thứ tự.

-LinkedHashSet: Bảo tồn thứ tự thêm vào, nhưng thao tác nhanh chậm hơn so với HashSet.

-TreeSet: Sắp xếp các phần tử tự nhiên (nếu phần tử thực hiện Comparable hoặc Comparator), nhưng thao tác chậm hơn các triển khai khác.

**21. Map:**

a. Map trong Java là gì? Đặc điểm và ứng dụng của Map là gì?

Map là tập hợp các entry. Mỗi entry gồm key và value. Sử dụng key để truy xuất giá trị của mỗi phần tử.

- Ứng dụng: quản lý theo dang cặp, thống kê,…

b. Các lớp triển khai Map?

- HashMap: Hỗ trợ thao tác nhanh chóng, không đảm bảo thứ tự của các cặp key-value.

- LinkedHashMap: Bảo tồn thứ tự thêm vào, thao tác nhanh hơn so với HashMap ở một số trường hợp.

- TreeMap: Sắp xếp các cặp key-value theo thứ tự tự nhiên hoặc có một Comparator được cung cấp).

- Hashtable: Tương tự như HashMap nhưng được đồng bộ hóa và không cho phép các giá trị hoặc khóa null.

c. Những hiểu biết của bạn về HashMap? Nó khởi tạo, lưu trữ, thêm, sửa, xoá, ... các phần tử như thế nào? Lưu ý nào khi dùng HashMap và nên dùng trong những trường hợp nào?

- HashMap lưu trữ các phần tử theo cặp key-value.

- Khi crud nó sẽ truy xuất theo key.

- Nên dùng khi cần ánh xạ key đến value (nhanh chóng).

**22. Collections:**

a. Collections là gì trong Java? Chức năng và sử dụng của Collections là gì?

Collection là cấu trúc dữ liệu được sử dụng để nắm giữ nhiều phần tử.

- Có thể thêm, xóa, cập nhật các phần tử.

- Hợp, giao, trừ… các tập hợp

Collection được chia làm 2 loại là List và Set

- List là collection mà mỗi phần tử được phép xuất hiện nhiều lần và truy xuất bằng chỉ số

- Set là collection mà mỗi phần tử chỉ được phép xuất hiện 1 lần và không được phép truy xuất theo chỉ số