**Ôn tập phỏng vấn**

# **Java Core**

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

Stack là một vùng nhớ được sử dụng để lưu trữ các tham số và các biến local của phương thức mỗi khi một phương thức được gọi ra.

Heap là một vùng nhớ trong bộ nhớ được sử dụng để lưu trữ các đối tượng khi từ khóa new được gọi ra, các biến static và các biến toàn cục (biến instance).

1. **Giá trị mặc định của các biến local là gì?**

Các biến local không được khởi tạo với bất kỳ giá trị mặc định nào, bất kể là nguyên thủy hay tham chiếu đối tượng.

1. **Khác nhau Biến tham trị và Biến tham chiếu?**

Biến tham trị là một biến mà giá trị của nó được truyền vào phương thức và khi trong phương thức thay đổi giá trị của nó thì không làm thay đổi giá trị ban đầu của nó.

Biến tham chiếu là biến mà giá trị của nó được nằm bên bộ nhớ heap và khi thay đổi giá trị của nó trong phương thức thì thay đổi giá trị ban đầu của nó.

1. **Class là gì?**

Class là một nhóm các đối tượng có chung thuộc tính và hành vi

1. **Biến local là gì?**

Là biến được tạo bên trong phương thức

1. **Biến instance là gì?**

Là biến được tạo bên ngoài phương thức và bên trong class

1. **Giá trị khởi tạo của biến tham chiếu đối tượng được định nghĩa là biến instance là gì?**

Null

1. **Constructor là gì?**

Là phương thức đặc biệt được dùng để tạo đối tượng thuộc một lớp. Constructor trả về giá trị là Object.

1. **Có thể tạo constructor final không?**

Không, có thể overload

|  |
| --- |
| 1. **Biến static là gì?**   Các biến được khai báo với từ khóa static tham chiếu đến thuộc tính chung cho tất cả các đối tượng của lớp. |
| 1. **Phương thức static là gì?**   Một phương thức static thuộc lớp chứ không phải đối tượng của lớp.  Một phương thức static gọi mà không cần tạo một instance của một lớp.  Phương thức static có thể truy cập biến static và có thể thay đổi giá trị của nó. |

1. **Khối static là gì?**

Được sử dụng để tạo phương thức và biến static. Khối static được chạy trước phương thức main.

1. **Sự khác nhau giữa phương thức static và phương thức instance?**

* Phương thức non-static thì gọi được các biến static và non-static
* Phương thức static thì không gọi được biến non-static

1. **This là gì, super là gì?**

* This là từ khóa để trỏ đến đối tượng thuộc lớp hiện tại
* Super là từ khóa để trỏ đến đối tượng thuộc lớp cha gần nhất

1. **Các tính chất của OOP là gì?**

* Kế thừa là một khái niệm trong OOP cho phép một class thừa hưởng tất cả các thuộc tính và phương thức của một class khác.
* Bao gói là nguyên tắc không cho phép truy cập trực tiếp vào thuộc tính của một lớp mà cần truy cập thông qua các phương thức liên quan.
* Đa hình là tính chất cho phép một đối tượng có những thể hiện khác nhau dựa trên những đối tượng khác nhau. VD: lớp dog và cat kế thừa lớp animal thì phương thức speak sẽ được overide theo 2 cách khác nhau
* Trừu tượng (abstraction) là một khái niệm quan trọng trong lập trình hướng đối tượng (OOP), nó cho phép ta tập trung vào các tính năng quan trọng của một đối tượng và bỏ qua các chi tiết không quan trọng.

1. **Access modifier và non - access modifier là gì?**

* Access modifier qui định phạm vi truy cập của các thành phần thuộc một lớp. Gồm 4 từ khóa: public, private, protected và default
* Non-access modifier qui định tính chất của các thành phần thuộc một lớp. Gồm

các từ khóa như: abstract, final, static

1. **Overloading và Overiding là gì?**

* Overloading là nạp chồng phương thức, có nghĩa là các phương thức có tên giống nhau nhưng khác nhau về tham số hoặc kiểu dữ liệu trả về.
* Overiding:

+ là ghi đè phương thức, là cơ chế cho phép lớp con triển khai lại phương thức đã được định nghĩa thuộc lớp cha.

+ Phương thức overide có cùng tên, kiểu dữ liệu trả về và tham số với phương thức thuộc lớp cha

+ phương thức thuộc lớp con phải có access modifier cao hơn lớp cha

1. **Có thể ghi đè phương thức static không?**

Không. Vì phương thức static thuộc về lớp chứ không thuộc về đối tượng.

1. **Từ khóa final?**

Biến final được dùng để định nghĩa hằng.

Phương thức final khổng thể ghi đè.

Lớp final không thể kế thừa.

1. **Có thể sử dụng cả abstract và final cho một phương thức không ?**

Không, vì phương thức trừu tượng (abstract) cần phải được ghi đè, trong khi đó không thể ghi đè được phương thức final.

1. **Interface là gì?**

Interface là một tập hợp các phương thức trừu tượng (abstract methods) và/hoặc các hằng số (constants) mà một lớp có thể triển khai (implement).

Các tính chất quan trọng của interface:

• Interface không phải là một lớp

• Trong interface chỉ được khai báo hằng không khai báo biến

• Các phương thức trong interface chỉ được gọi tên chứ không được định nghĩa

• Không thể khởi tạo đối tượng từ interface

• Các interface có thể kế thừa lẫn nhau

• Một class có thể implements nhiều interface

1. **Xử lý ngoại lệ (handling exception) là gì?**

Là một cơ chế mạnh mẽ để xử lý các lỗi runtime để có thể duy trì luồng bình thường của ứng dụng.

1. **Sự khác biệt giữa checked exception và unchecked exception là gì?**

Checked exception:

Các lớp được extends lớp Throwable ngoại trừ RuntimeException và Error được gọi là các checked exception. Ví dụ: IOException, SQLException, ... Checked exception được kiểm tra tại thời gian biên dịch.

Unchecked exception:

Các lớp được extends lớp RuntimeException được gọi là unchecked exception. Ví dụ như ArithmeticException, NullPointerException, ... Unchecked exception không được kiểm tra tại thời gian biên dịch.

1. **Từ khóa finally**

* Khối lệnh finally trong java luôn được thực thi cho dù có ngoại lệ xảy ra hay không.
* Khối finally phải theo sau khối try hoặc catch.

1. **Phân biệt throw và throws?**

|  |  |  |
| --- | --- | --- |
| No | Throw | Throws |
| 1) | Từ khóa throw trong java được sử dụng để ném ra một ngoại lệ rõ ràng. | Từ khóa throws trong java được sử dụng để khai báo một ngoại lệ. |
| 2) | Ngoại lệ checked không được truyền ra nếu chỉ sử dụng từ khóa throw. | Ngoại lệ checked được truyền ra ngay cả khi chỉ sử dụng từ khóa throws. |
| 3) | Sau throw là một instance. | Sau throws là một hoặc nhiều class. |
| 4) | Throw được sử dụng trong phương thức. | Throws được khai báo ngay sau dấu đóng ngoặc đơn của phương thức. |
| 5) | Bạn không thể throw nhiều exceptions. | Bạn có thể khai báo nhiều exceptions, Ví dụ: public void method()throws IOException,SQLException. |

1. **Ý nghĩa của immutable (bất biến) trong String là gì?**

của immutable là không thể sửa đổi hoặc không thể thay đổi được. Một khi đối tượng String đã được tạo ra, giá trị của nó không thể thay đổi

1. **Tại sao các đối tượng String trong java là immutable?**

Bởi vì java sử dụng khái niệm về string literal. Giả sử có 5 biến tham chiếu, tất cả các tham chiếu đến một đối tượng "hello". Nếu một biến tham chiếu thay đổi giá trị của đối tượng, nó sẽ bị ảnh hưởng đến tất cả các biến tham chiếu khác. Đó là lý do tại sao đối tượng String trong java immutable (bất biến).

|  |
| --- |
| 1. **Có bao nhiêu cách để tạo ra một đối tượng String trong java?**   Có 2 cách để tạo đối tượng String:  Sử dụng string literal  Sử dụng từ khóa new |
| 1. **Tại sao java sử dụng khái niệm string literal?**   Vì điều này khiến Java sử dụng bộ nhớ hiệu quả hơn (vì không có đối tượng String mới được tạo ra trong string constant pool). |
| 1. **Sự khác nhau giữa String và StringBuffer là gì?**   string là một đối tượng immutable (bất biến). StringBuffer là môt đối tượng mutable (có thể biến đổi). |
| 1. **Sự khác nhau giữa StringBuffer và StringBuilder là gì?**   StringBuffer là đồng bộ còn StringBuilder là KHÔNG đồng bộ. |
| 1. **Làm thế nào để tạo lớp immutable trong java?**   Chúng ta có thể tạo ra lớp immutable như lớp String bằng cách định lớp với từ khóa final |
| 1. **Mục đích của phương thức toString() trong java là gì?**   Phương thức toString() trả về biểu diễn chuỗi của bất kỳ đối tượng nào. |

1. **Sự khác nhau giữa Array và ArrayList?**

|  |  |  |
| --- | --- | --- |
|  | **Array** | **ArrayList** |
| 1 | 1) Kích thước **cố định**. | Kích thước có thể **thay đổi được**. |
| 2 | 2) Có thể lưu trữ dữ liệu kiểu **nguyên thủy** và **đối tượng**. | Chỉ có thể lưu trữ dữ liệu kiểu **đối tượng**. Kể từ Java 5, kiểu nguyên thủy được tự động chuyển đổi trong các đối tượng được gọi là **auto-boxing**. |
| 3 | 3) Tốc độ lưu trữ và thao tác **nhanh hơn**. | Tốc độ lưu trữ vào thao tác **chậm hơn**. |
| 4 | 4) Chỉ có thuộc tính **length**. | Có nhiều phương thức để thao tác với dữ liệu. |

1. **Hoạt động xóa trong LinkedList là nhanh hơn trong ArrayList, tại sao?**

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

1. **Sự khác nhau giữa ArrayList và LinkedList?**

|  |  |  |
| --- | --- | --- |
| No. | ArrayList | LinkedList |
| 1) | ArrayList sử dụng một mảng động. | LinkedList sử dụng danh sách liên kết doubly. |
| 2) | ArrayList không hiệu quả với thao tác vì cần nhiều chuyển đổi. | LinkedList là hiệu quả cho thao tác. |
| 3) | ArrayList là tốt hơn để lưu trữ và lấy dữ liệu. | LinkedList là tốt hơn để thao tác dữ liệu. |