# **List câu hỏi phỏng vấn Java Collection**

Trong java, câu hỏi phỏng vấn collection thường được hỏi thường xuyên bởi người phỏng vấn. Dưới đây là list câu hỏi phỏng vấn Java Collection

Trước khi đi chi tiết vào các câu hỏi phỏng vấn, bạn cần phải biết **Legacy Class** trong java là gì?

Phiên bản Java đầu tiên không bao gồm Collection Framework. Nó chỉ định nghĩa một vài lớp và interface cung cấp các phương thức để lưu trữ các đối tượng. Khi Collection Framework được thêm vào trong J2SE 1.2, các lớp gốc đã được tái cấu trúc để hỗ trợ các collection interface. Các lớp này còn được gọi là lớp Legacy. Tất cả các lớp và interface Legacy được thiết kế lại bởi JDK 5 để hỗ trợ Generics.

Dưới đây là các Legacy Class được định nghĩa trong package **java.util**:

1. Dictionary
2. HashTable
3. Properties
4. Stack
5. Vector

Dưới đây là list câu hỏi phỏng vấn java collection:

1. Sự khác nhau giữa ArrayList và Vector là gì?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **Vector** |
| 1) | ArrayList là KHÔNG synchronized. | Vector là synchronized. |
| 1) | ArrayList không phải là legacy class. | Vector là legacy class. |
| 2) | ArrayList tăng kích thước của nó bằng 50% kích thước mảng. | Vector tăng kích thước của nó bằng cách nhân đôi kích thước mảng. |

2. Sự khác nhau giữa ArrayList và LinkedList là gì?

|  |  |  |
| --- | --- | --- |
| **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. |

3. Sự khác nhau giữa Iterator và ListIterator là gì?

Iterator duyệt các phần tử chỉ theo một chiều hướng là chuyển tiếp, trong khi ListIterator duyệt các phần tử theo hai hướng là chuyển tiếp và ngược lại.

Iterator có thể được sử dụng trong List, Set và Queue.

ListIterator chỉ có thể được sử dụng trong List.

4. Sự khác biệt giữa Iterator và Enumeration là gì?

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **Enumeration** |
| 1) | Iterator duyệt các phần tử legacy và non-legacy. | Enumeration chỉ có thể duyệt các phần tử legacy. |
| 2) | Iterator là chậm hơn Enumeration. | Enumeration là nhanh hơn Iterator. |

5. Sự khác nhau giữa List và Set là gì?

List có thể chứa các phần tử **trùng lặp (dublicate)**, trong khi Set chỉ chứa các phần tử duy nhất.

6. Sự khác nhau giữa HashSet và TreeSet là gì?

HashSet **không** duy trì **thứ tự nào**, trong khi TreeSet duy trì **thứ tự tăng dần**.

7. Sự khác nhau giữa Set và Map là gì?

Set chỉ chứa giá trị, trong khi Map chứa cặp key và value.

8. Sự khác biệt giữa HashSet và HashMap là gì?

HashSet chỉ chứa giá trị, trong khi HashMap chứa cặp key và value.

9. Sự khác nhau giữa HashMap và TreeMap là gì?

HashMap duy trì **không có thứ tự**, trong khi TreeMap duy trì **thứ tự tăng dần**.

10. Sự khác nhau giữa HashMap và Hashtable là gì?

|  |  |  |
| --- | --- | --- |
| **No.** | **HashMap** | **Hashtable** |
| 1) | HashMap là KHÔNG synchronized. | Hashtable là synchronized. |
| 2) | HashMap có thể chứa một khóa null và nhiều giá trị null. | Hashtable không thể chứa bất kỳ khóa null hoặc giá trị null. |

11. Sự khác nhau giữa Collection và Collections là gì?

Collection là một interface, trong khi Collections là một lớp. Collecion interface cung cấp các chức năng về cấu trức dữ liệu cho List, Set, Queue. Nhưng lớp Collections là để sắp xếp và đồng bộ các phần tử Collection.

12. Sự khác nhau giữa Comparable và Comparator là gì?

|  |  |  |
| --- | --- | --- |
| **No.** | **Comparable** | **Comparator** |
| 2) | Nó cung cấp phương thức compareTo(). | Nó cung cấp phương thức compare(). |
| 3) | Nó được đặt trong java.lang package. | Nó được đặt trong java.util package. |
| 4) | Nếu chúng ta một lớp được implement Comparable interface, thì lớp đó phải được sửa đổi. | Lớp không bị sửa đổi. |

13. Lợi thế của Properties file là gì?

Nếu bạn thay đổi giá trị trong tệp thuộc tính, bạn không cần phải biên dịch lại lớp java. Vì vậy, nó làm cho ứng dụng dễ quản lý.

14. Phương thức hashCode() là gì?

Phương thức hashCode() trả về một giá trị mã băm (một số nguyên).

Phương thức hashCode() trả về cùng số nguyên, nếu hai keys (bằng phương thức equals()) giống nhau.

Tuy nhiên, có thể hai mã băm có thể có các keys khác nhau hoặc giống nhau.

15. Tại sao chúng ta phải nghi đè phương thức equals()?

Phương thức equals() được sử dụng để kiểm tra xem hai đối tượng có giống nhau hay không. Nó cần phải được ghi đè nếu chúng ta muốn kiểm tra các đối tượng dựa trên thuộc tính của chúng.

Ví dụ, Nhanvien là một lớp có 3 thành viên dữ liệu: id, ten và luong. Nhưng, chúng ta muốn kiểm tra sự giống nhau của đối tượng nhân viên trên cơ sở tiền lương. Khi đó, chúng ta cần ghi đè bằng phương thức equals().

16. Làm thế nào để đồng bộ List, Set và Map?

Lớp Collection cung cấp phương thức để làm cho các phần tử List, Set và Map là đồng bộ:

|  |
| --- |
| public static List synchronizedList(List l){} |
| public static Set synchronizedSet(Set s){} |
| public static SortedSet synchronizedSortedSet(SortedSet s){} |
| public static Map synchronizedMap(Map m){} |
| public static SortedMap synchronizedSortedMap(SortedMap m){} |

17. Lợi ích của generic collection là gì?

Nếu chúng ta sử dụng lớp generic, chúng ta không cần typecasting. Nó là typesafe và kiểm tra tại thời gian biên dịch.

18. Hash-collision trong Hashtable là gì? Và nó được xử lý như thế nào?

Hai keys khác nhau có cùng giá trị băm được gọi là sự va chạm băm (hash-collision). Hai mục khác nhau sẽ được giữ trong một chậu băm đơn để tránh va chạm.

19. Lớp Dictionary là gì?

Lớp Dictionary cung cấp khả năng lưu trữ các cặp key-value.

20. Sự khác nhau giữa Array và ArrayList là gì?

**Sự khác nhau giữa Array với ArrayList** được tóm tắt lại như trong bảng sau:

|  |  |
| --- | --- |
| **Array** | **ArrayList** |
| 1) Kích thước **cố định**. | Kích thước có thể **thay đổi được**. |
| 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) Tốc độ lưu trữ và thao tác **nhanh hơn**. | Tốc độ lưu trữ vào thao tác **chậm hơn**. |
| 4) Chỉ có thuộc tính **length**. | Có nhiều phương thức để thao tác với dữ liệu. |