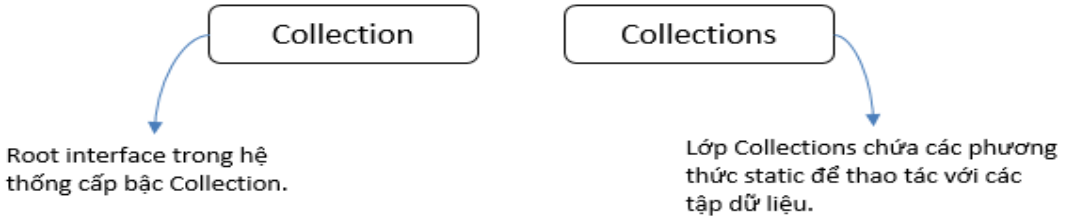
Hướng dẫn sử dụng Java Collection Framework

1. Collection vs Collections

"Collection" và "Collections" trong java là hai khái niệm khác nhau.

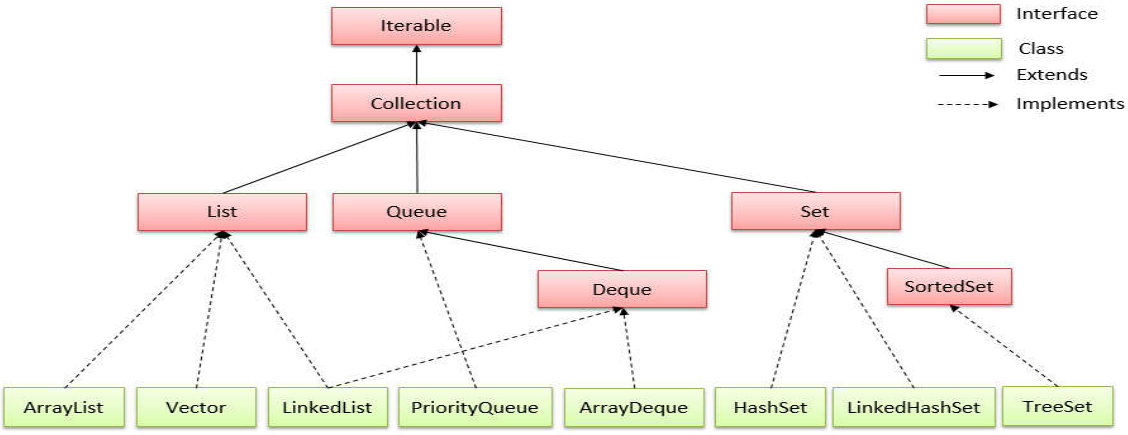
Collections trong java là một khuôn khổ cung cấp một kiến trúc để lưu trữ và thao tác tới nhóm các đối tượng. Tất cả các hoạt động mà bạn thực hiện trên một dữ liệu như tìm kiếm, phân loại, chèn, xóa,… có thể được thực hiện bởi Java Collections.

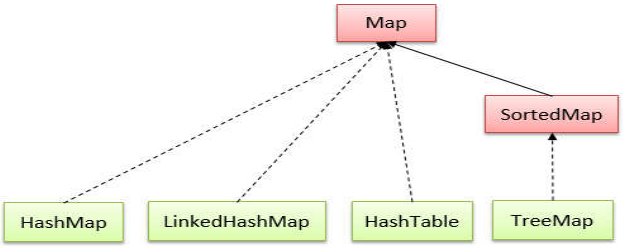
Collection trong java là một root interface trong hệ thống cấp bậc Collection. Java Collection cung cấp nhiều interface (Set, List, Queue, Deque vv) và các lớp (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet vv).



2. Hệ thống cấp bậc Collection trong java

Gói **java.util** chứa tất cả các lớp và interface của Collection.





3. Dưới đây là mô tả những interface chính của Collection:

- **Set:** là một collection không thể chứa 2 giá trị trùng lặp. Set được sử dụng để biểu diễn các bộ, chẳng hạn như bộ tú lu khơ, thời khóa biểu của học sinh, các tiến trình đang chạy trên máy tính…
- **List:** là một collection có thứ tự (đôi khi còn được gọi là một chuỗi). List có thể chứa các phần tử trùng lặp. Thường có quyền kiểm soát chính xác vị trí các phần tử được chèn vào và có thể truy cập chúng bằng chỉ số (vị trí của chúng).
- **Queue (hàng đợi):** là một collection được sử dụng để chứa nhiều phần tử trước khi xử lý. Bên cạnh các thao tác cơ bản của collection, Queue cung cấp các thao tác bổ sung như chèn, lấy ra và kiểm tra. Queue có thể được sử dụng như là FIFO (first-in, first-out – vào trước, ra sau)
- **Deque:** là một collection được sử dụng để chứa nhiều phần tử trước khi xử lý. Ngoài các thao tác cơ bản của collection, một Deque cung cấp các thao tác bổ sung như chèn, lấy ra và kiểm tra. Deques có thể được sử dụng như là FIFO (first-in, first-out – vào trước, ra sau) và LIFO (last-in, first-out – vào sau, ra trước). Trong một Deque, tất cả các phần tử mới có thể được chèn vào, lấy ra và lấy ra ở cả hai đầu.
- **Map:** là một đối tượng ánh xạ mỗi key tương úng với một giá trị. Map không thể chứa giá trị trùng lặp. Mỗi key có thể ánh xạ đến nhiều nhất một giá trị.

4. Dưới đây là mô tả 2 interface được sắp xếp của Set mà Map:

- **SortedSet:** là một Set chứa các phần tử theo thứ tự tăng dần.
- **SortedMap:** là một Map chứa các phần tử được sắp xếp theo thứ tự tăng dần của key của chúng. Các SortedMap được sử dụng cho các collection theo thứ tự tự nhiên của cặp key/value, chẳng hạn như từ điển và danh bạ điện thoại.

5. List Interface trong Java

List Interface trong Java kế thừa Collection và khai báo các hành vi của một collection mà lưu giữ một dãy các phần tử.

Các phần tử có thể được chèn hoặc được truy cập thông qua vị trí của chúng trong danh sách, bởi sử dụng chỉ mục xây dựng bắt đầu từ 0.

Một list có thể chứa nhiều bản sao phần tử.

Ngoài các phương thức được định nghĩa bởi Collection, List Interface định nghĩa một số phương thức riêng.

Một số phương thức của List Interface sẽ ném một UnsupportedOperationException nếu collection không thể bị sửa đổi, và ném một ClassCastException nếu một đối tượng là không tương thích với đối tượng khác.

6. ArrayList trong java

Lớp ArrayList trong java được sử dụng như một mảng động để lưu trữ các phần tử. Nó kế thừa lớp AbstractList và impliments giao tiếp List.

Những điểm cần ghi nhớ về ArrayList:

- Lớp ArrayList trong java có thể chứa các phần tử trùng lặp.
- Lớp ArrayList duy trì thứ tự của phần tử được thêm vào.
- Lớp ArrayList là không đồng bộ (non-synchronized).
- Lớp ArrayList cho phép truy cập ngẫu nhiên vì nó lưu dữ liệu theo chỉ mục.
- Lớp ArrayList trong java, thao tác chậm vì cần nhiều sự dịch chuyển nếu bất kỳ phần tử nào bị xoá khỏi danh sách.

7. Vector trong Java

Lớp Vector trong Java triển khai một mảng động. Nó tương tự như ArrayList, nhưng với hai điểm khác biệt:

- Vector được đồng bộ.
- Vector chứa các phương thức legacy mà không là một phần của Collection Framework.

8. LinkedList trong java

Lớp LinkedList trong java sử dụng cấu trúc danh sách liên kết Doubly (Doubly Linked List) để lưu trữ các phần tử.

Những điểm cần ghi nhớ về lớp LinkedList:

- Lớp LinkedList trong java có thể chứa các phần tử trùng lặp.
- Lớp LinkedList duy trì thứ tự của phần tử được thêm vào.
- Lớp LinkedList là không đồng bộ (non-synchronized).
- Trong java lớp LinkList, thao tác nhanh vì không cần phải dịch chuyển nếu bất kỳ phần tử nào bị xoá khỏi danh sách.
- Lớp LinkList trong java có thể được sử dụng như list (danh sách), stack (ngăn xếp) hoặc queue (hàng đợi).

9. Set Interface trong Java

Set Interface trong Java là một Collection mà không chứa các bản sao phần tử.

Set Interface trong Java chỉ chứa các phương thức được kế thừa từ Collection và thêm sự giới hạn về việc ngăn cấm các phần tử bản sao.

Set Interface cũng thêm các hoạt động hashCode cho phép Set Interface so sánh một cách có ý nghĩa ngay cả khi kiểu triển khai của nó là khác.

10. SortedSet Interface trong Java

SortedSet Interface trong Java kế thừa Set và khai báo các hành vi của một Set được xếp thứ tự tăng dần. Ngoài những phương thức được định nghĩa bởi Set, thì SortedSet Interface trong Java khai báo các phương thức được liệt kê trong bảng dưới đây.

Một số phương thức ném một NoSuchElementException khi không có item nào được chứa trong Set đang gọi. Một ClassCastException được ném khi một đối tượng không tương thích với phần tử trong một Set.

11. HashSet trong java

Lớp **HashSet trong java** được sử dụng để tạo một bộ sưu tập sử dụng bảng băm để lưu trữ. Nó kế thừa lớp AbstractSet và implements giao diện Set.

Các điểm quan trọng về lớp HashSet trong java là:

- HashSet chỉ chứa các phần tử duy nhất.
- HashSet lưu trữ các phần tử bằng cách sử dụng một cơ chế được gọi là **băm**.

12. LinkedHashSet trong java

Lớp **LinkedHashSet trong java** là một bản cài đặt bảng băm và danh sách liên kết của giao diện Set. Nó kế thừa lớp HashSet và implements giao diện Set.

Những điểm quan trọng về lớp LinkedHashSet trong java là:

- Chỉ chứa các phần tử duy nhất giống như HashSet.
- Cho phép các phần tử null.
- Duy trì thứ tự chèn.

13. TreeSet trong java

Lớp **TreeSet trong java** implements giao diện Set sử dụng cấu trúc cây để lưu trữ các phần tử. Nó kế thừa lớp AbstractSet và implements giao diện NavigableSet. Các đối tượng của lớp TreeSet được lưu trữ theo thứ tự tăng dần.

Các điểm quan trọng về lớp TreeSet trong java là:

- Chỉ chứa các phần tử duy nhất giống như HashSet.

- Thời gian truy xuất nhanh.
- Duy trì thứ tự tăng dần.

14. Map trong java

Trong java, Map được sử dụng để lưu trữ và truy xuất dữ liệu theo cặp key và value. Mỗi cặp key và value được gọi là mục nhập (entry). Map trong java chỉ chứa các giá trị key duy nhất. Map rất hữu ích nếu bạn phải tìm kiếm, cập nhật hoặc xóa các phần tử trên dựa vào các key.

15. HashMap trong java

Những điểm quan trọng về lớp **HashMap trong java** là:

- HashMap lưu trữ dữ liệu dưới dạng cặp key và value.
- Nó chứa các key duy nhất.
- Nó có thể có 1 key là null và nhiều giá trị null.
- Nó duy trì các phần tử KHÔNG theo thứ tự chèn.

16. Hashtable trong java

Những điểm quan trọng về lớp **Hashtable trong java** là:

- Hashtable là một mảng của list. Mỗi list được biết đến như một xô chứa các phần tử. Vị trí của một xô được xác định bằng việc gọi phương thức hashcode(). Hashtable cũng lưu trữ dữ liệu dưới dạng cặp key và value.
- Nó chứa các key duy nhất.
- Nó KHÔNG thể có bất kỳ key hoặc giá trị nào là null.
- Nó được đồng bộ.

17. TreeMap trong java

Các điểm quan trọng về lớp **TreeMap trong java** là:

- TreeMap lưu trữ dữ liệu dưới dạng cặp key và value.
- Nó chứa các key duy nhất.
- Nó KHÔNG cho phép bất kỳ key nào là null nhưng có thể có nhiều giá trị null.
- Nó duy trì các phần tử được thêm vào theo thứ tự key tăng dần.

18. LinkedHashMap trong java

Những điểm quan trọng về lớp **LinkedHashMap trong java** là:

- LinkedHashMaplưu trữ dữ liệu dưới dạng cặp key và value.
- Nó chứa các key duy nhất.
- Nó có thể có 1 key là null và nhiều giá trị null.

- Nó duy trì các phần tử theo thứ tự chèn.

19. Lớp Stack trong Java

Lớp Stack là một lớp phụ của lớp Vector trong Java mà triển khai một last-in-first-out (LIFO) stack. Bạn có thể nghĩ về Stack như là một ngăn xếp thẳng đứng.

Stack chỉ định nghĩa constructor mặc định, mà tạo một stack trống. Lớp Stack bao gồm tất cả phương thức được định nghĩa bởi lớp Vector, và một số phương thức khác của riêng nó.

20. Queue Interface trong Java

Queue là một cấu trúc dữ liệu, dùng để lưu trữ nhiều đối tượng(phần tử dữ liệu) làm việc theo cơ chế "vào trước, ra trước" (First In/First Out (FIFO)) tức là phần tử nào thêm vào đầu tiên sẽ được lấy ra đầu tiên.

21. Deque Interface trong Java

Deque là một interface mới bổ sung vào Java 6. Chúng ta đã biết đến queue – hàng đợi, vào trước thì ra trước, thêm dữ liệu vào ở một đầu và lấy ra và loại bỏ ở một đầu khác. Deque thì khác, mặc dù là thừa kế từ Queue interface nhưng Deque cho phép thêm hoặc lấy ra và loại bỏ dữ liệu ở cả hai đầu: đầu và cuối. Deque giống như là một dạng cấu trúc dữ liệu cộng gộp những đặc trưng từ stack – xếp chồng và queue – hàng đợi. Nó bổ sung một loạt những method để đáp ứng khả năng đó như addFirst – addLast, getFirst – getLast, peekFirst – peekLast, pollFirst – pollLast,...

22. PriorityQueue trong Java

PriorityQueue lưu trữ các phần tử trong nội bộ theo trật tự tự nhiên của các phần tử (nếu các phần tử này là kiểu Comparable), hoặc theo một Comparator (bộ so sánh) được cài đặt cho PriorityQueue.

23. ArrayDeque trong Java: là 1 dạng deque (queue 2 chiều) được implement dựa trên mảng.

24. NavigableSet Interface trong Java

NavigableSet là một interface được kế thừa từ SortedSet. Vì vậy nó mang đầy đủ đặc điểm của SortedSet, điểm khác biệt là NavigableSet có thêm các phương thức bổ sung cho việc sắp xếp các phần tử trong Set. Một điều đáng ngạc nhiên là cả 2 class TreeSet và ConcurrentSkipListSet thực thi giao diện SortedSet ở trên cũng đều thực thi giao diện NavigableSet.

25. ConcurrentSkipListSet trong Java

ConcurrentSkipListSet là một collection được giới thiệu từ Java 6 nằm trong gói java.util.concurrent. Đây là một Set, không chấp nhận chứa cùng 2 đối tượng trong tập động này. Việc kiểm tra sự trùng lặp đối tượng thông qua hàm equals của lớp Object. Chẳng hạn với một ConcurrentSkipListSet chứa tập kiểu integer, sẽ không thể có hai giá trị là 11 cùng nằm trong collection này. ConcurrentSkipListSet là một tập dữ liệu có sắp xếp. Việc sắp xếp các phần tử theo tuần tự tự nhiên (tăng dần) phụ thuộc vào việc khởi tạo đối tượng với có hoặc không có Comparator truyền vào.

26. ConcurrentMap trong Java

ConcurrentMap là một giao diện con của java.util.Map trong đó định nghĩa các hoạt động atomic hữu dụng. Những hoạt động này sẽ xóa hoặc thay thế cặp key-value chỉ khi key tồn tại, hoặc thêm một cặp key-value chỉ khi key không tồn tại. Việc tạo những hoạt động này sẽ giúp tránh được việc đồng bộ hóa. Bộ thực thi theo chuẩn chung của ConcurrentMap là ConcurrentHashMap, đó là tính tương tự đồng thời của HashMap.

27. ConcurrentNavigableMap trong Java

ConcurrentNavigableMap là một giao diện con của ConcurrentMap dùng để hỗ trợ sự tương thích xấp xỉ. Bộ thực thi theo chuẩn chung của ConcurrentNavigableMap là ConcurrentSkipListMap, đó là tính tương tự đồng thời của TreeMap.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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**.

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

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

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

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

36. 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**.

37. 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.

38. 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 trúc dữ liệu cho List, Set, Queue. Nhưng lớp Collections là để sắp xếp và đồng bộ các phần tử Collection.

39. 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.

40. 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.

41. 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.

42. 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().

43. 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){}

44. 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.

45. 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.

46. Lớp Dictionary là gì?

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

47. 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.