

# Bài 65: Giới thiệu Collections Framework

---

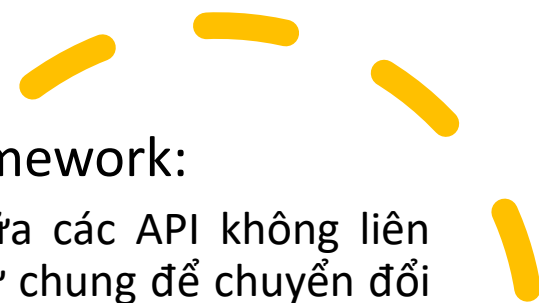
- ✓ Mục đích, đặc điểm
- ✓ Các interface trong framework
- ✓ Interface Collection
- ✓ Interface Iterator
- ✓ Một số lớp trong framework

# Mục đích sử dụng

- Java cung cấp sẵn collections framework
- Một collection là một đối tượng đại diện cho một nhóm các đối tượng nào đó
- Một collections framework là một kiến trúc thống nhất để đại diện và thao tác với các tập hợp.
- Các mục đích của collections framework:
  - Giảm gánh nặng về yêu cầu kĩ năng lập trình bằng cách cung cấp các cấu trúc dữ liệu và giải thuật để lập trình viên không phải tự viết chúng
  - Tăng hiệu năng bằng cách cung cấp các triển khai của cấu trúc dữ liệu và giải thuật hiệu năng cao. Các chương trình có thể chuyển đổi qua lại



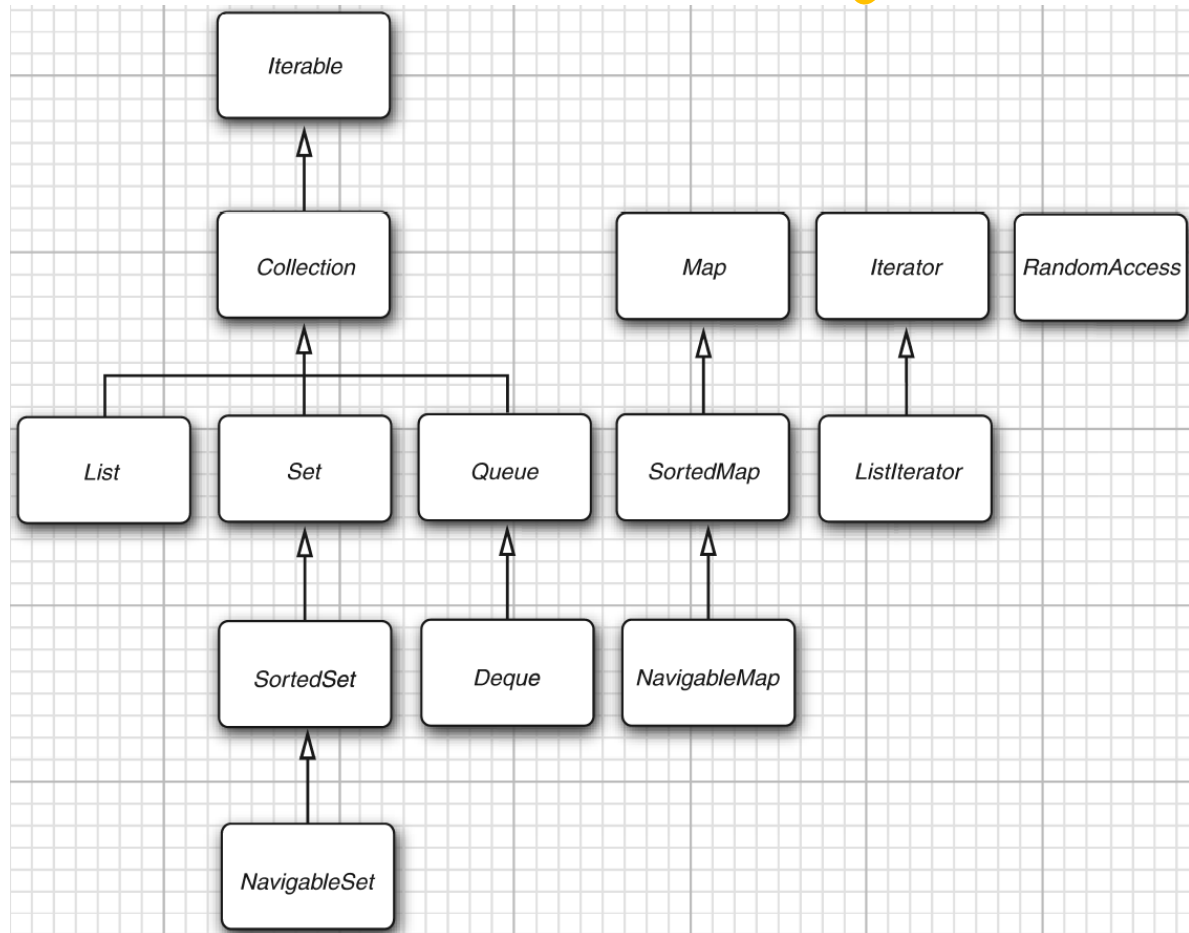
# Mục đích sử dụng

- 
- Các mục đích của collections framework:
    - Cung cấp khả năng tương tác giữa các API không liên quan bằng cách thiết lập ngôn ngữ chung để chuyển đổi các collection qua lại lẫn nhau
    - Giảm nỗ lực cần thiết để thiết kế và triển khai các API bằng cách không yêu cầu người dùng tạo các collection API chuyên biệt
    - Khuyến khích tái sử dụng phần mềm bằng cách cung cấp các interface chuẩn cho các tập hợp và các thuật toán để thao tác với chúng

# Nội dung framework

- Các interface: đây là các kiểu dữ liệu trừu tượng đại diện cho các tập hợp. Các interface cho phép các tập hợp được thao tác độc lập với các biểu diễn chi tiết của chúng
- Các lớp triển khai interface: đây là các lớp triển khai cụ thể của các interface trong tập hợp. Về bản chất chúng là các cấu trúc dữ liệu có thể tái sử dụng
- Các thuật toán: các phương thức thực hiện các tính toán hữu ích như sắp xếp, tìm kiếm các đối tượng. Các thuật toán có thể được thực thi khác nhau tùy vào việc thực thi interface nào.

# Các interface của framework



# Ý nghĩa sử dụng

- Collection: cung cấp các phương thức để làm việc với một nhóm các đối tượng. Đây là 1 interface gốc của framework
- List: kế thừa Collection, dùng để chứa danh sách các phần tử có thứ tự
- Set: kế thừa Collection, dùng để lưu trữ tập các phần tử không trùng nhau
- SortedSet: kế thừa Set, dùng để lưu trữ các phần tử không trùng nhau và được sắp xếp theo thứ tự
- NavigableSet: kế thừa SortedSet, dùng để điều hướng các kết quả tìm kiếm phù hợp nhất

# Ý nghĩa sử dụng

- Queue: hàng đợi dùng để lưu trữ các phần tử theo tiêu chí vào trước ra trước
- DeQueue: kế thừa Queue, cho phép cập nhật danh sách phần tử ở cả hai đầu
- Map: lưu trữ dữ liệu dạng key-value
- SortedMap: kế thừa Map, cung cấp một map với các key được sắp xếp theo thứ tự tăng dần
- Iterator: cung cấp khả năng duyệt, loại bỏ phần tử của tập hợp

# Ý nghĩa sử dụng

- ListIterator: kế thừa Iterator, cho phép duyệt và sửa đổi tập hợp theo cả hai chiều
- RandomAccess: là một interface đánh dấu nhằm chỉ ra rằng tập hợp thực thi nó hỗ trợ truy cập ngẫu nhiên(sử dụng chỉ số phần tử)



# Interface Collection

➤ Một số phương thức và mô tả:

Phương thức	Mô tả
<code>boolean add(E e)</code>	Thêm một phần tử vào tập hợp hiện thời. Nếu thêm thành công return true và ngược lại return false
<code>boolean addAll(Collection&lt;? extends E&gt; c)</code>	Thêm tất cả các phần tử trong một tập hợp cho trước vào tập hợp hiện thời
<code>void clear()</code>	Xóa bỏ toàn bộ các phần tử trong tập hợp hiện thời
<code>boolean contains(Object o)</code>	Kiểm tra xem tập hợp hiện thời chứa đối tượng cho trong tham số hay không
<code>boolean containsAll(Collection&lt;?&gt; c)</code>	Trả về true nếu tập hợp hiện thời chứa tất cả các phần tử trong tập hợp được chỉ ra trong tham số
<code>boolean equals(Object o)</code>	So sánh hai đối tượng về tính tương đương
<code>int hashCode()</code>	Trả về mã băm của đối tượng tập hợp hiện thời
<code>boolean isEmpty()</code>	Trả về true nếu tập hợp hiện thời không có phần tử nào cả
<code>Iterator&lt;E&gt; iterator()</code>	Trả về một iterator của các phần tử trong tập hợp hiện thời
<code>boolean remove(Object o)</code>	Loại bỏ một phần tử được chỉ định nếu nó tồn tại
<code>boolean removeAll(Collection&lt;?&gt;c)</code>	Xóa toàn bộ các phần tử chứa trong tham số nếu nó xuất hiện trong tập hợp hiện thời
<code>int size()</code>	Trả về số phần tử hiện có của tập hợp
<code>Object[] toArray()</code>	Trả về một mảng chứa tất cả các phần tử trong tập hợp hiện thời

# Interface Iterator

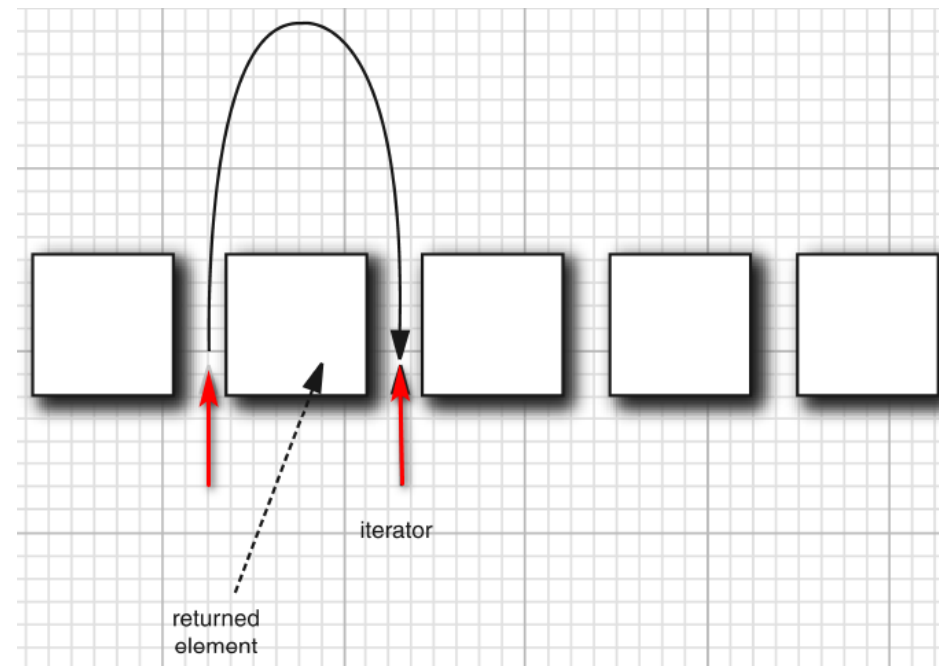
- Là một trình lặp trên tập hợp, tức có thể sử dụng đối tượng của Iterator để thực hiện các thao tác lặp
- Các phương thức của nó và mô tả:

Phương thức	Mô tả
default void forEachRemaining(Consumer<? super E> action)	Thực hiện hành động cho trước với từng phần tử còn lại của tập hợp đến khi tất cả các phần tử đã được xét duyệt hoặc hành động văng một ngoại lệ
boolean hasNext()	Trả về true nếu có phần tử kế tiếp để duyệt
E next()	Trả về phần tử tiếp theo
default void remove()	Xóa bỏ phần tử trả về bởi iterator hiện tại

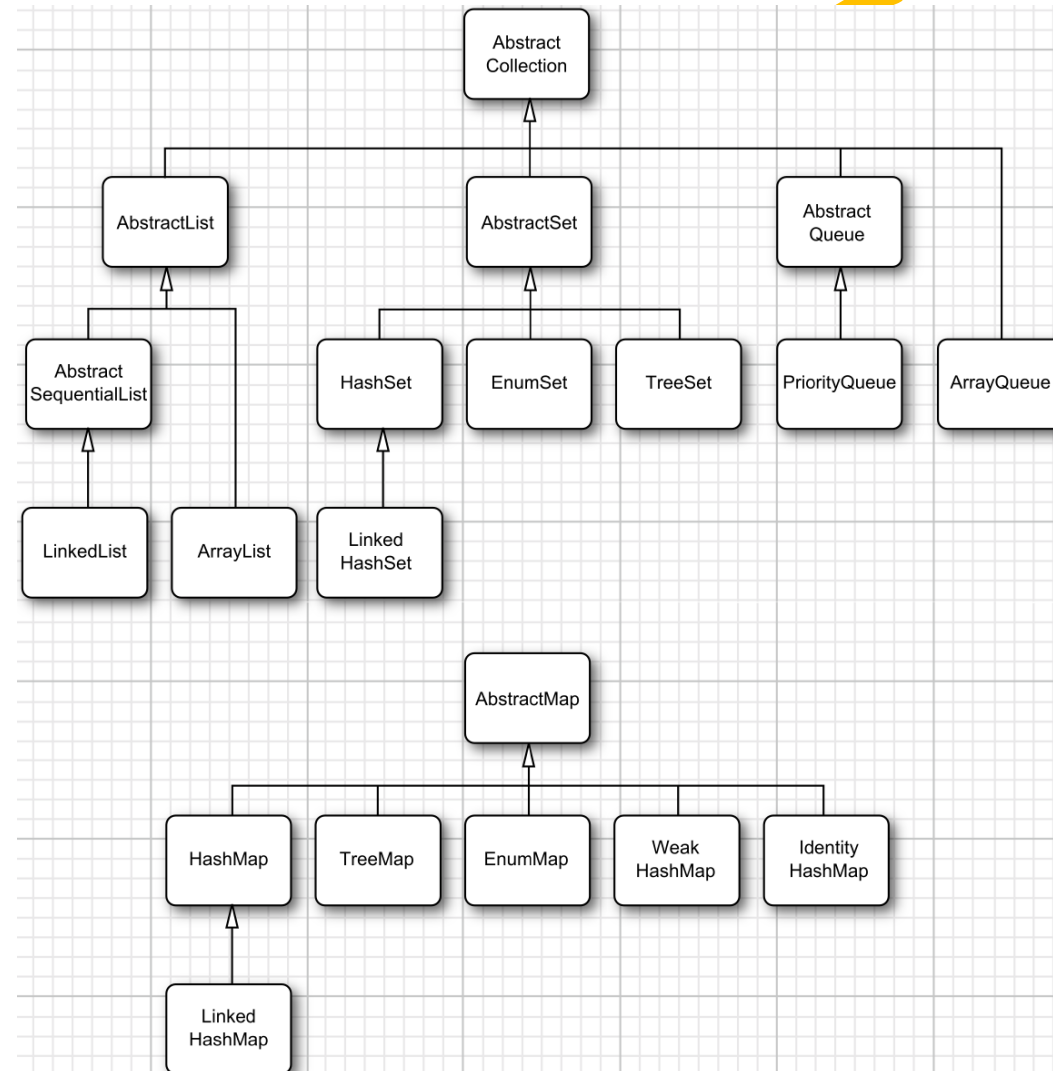
- Interface Collection kế thừa từ Iterable, mà Iterable chứa Iterator nên ta có thể sử dụng foreach cho mọi dạng Collection

# Interface Iterator

- Lưu ý rằng Iterator thực hiện bước nhảy giữa các phần tử chứ không phải nhảy từ phần tử này đến phần tử kia nên khi ta gọi `next()` nó sẽ nhảy qua phần tử kế tiếp rồi trả về tham chiếu đến phần tử vừa nhảy qua



# Các lớp thực thi

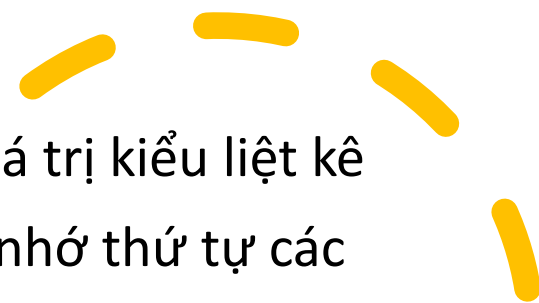


# Ý nghĩa sử dụng

- ArrayList: tập hợp các phần tử có thể tự thay đổi kích thước cho phù hợp, truy cập bằng chỉ số
- LinkedList: tập hợp có thứ tự được sử dụng hiệu quả trong trường hợp chèn, xóa phần tử ở vị trí bất kì
- ArrayDeque: một hàng đợi hai chiều cho phép thực hiện các hành động thêm, xóa phần tử ở hai đầu
- HashSet: tập hợp không có thứ tự và không cho phép trùng lặp phần tử
- TreeSet: tập hợp có thứ tự của các phần tử không trùng nhau



# Ý nghĩa sử dụng

- 
- EnumSet: một tập hợp của các giá trị kiểu liệt kê
  - LinkedHashSet: một tập hợp ghi nhớ thứ tự các phần tử được thêm vào
  - PriorityQueue: hàng đợi ưu tiên
  - HashMap: một cấu trúc dữ liệu dạng key-value
  - TreeMap: một map trong đó các key được sắp xếp theo thứ tự
  - EnumMap: một map của các key kiểu liệt kê
  - LinkedHashMap: một map ghi nhớ thứ tự các phần tử được thêm vào

# Ý nghĩa sử dụng

- WeakHashMap: một map với các giá trị có thể bị thu hồi bởi trình dọn dẹp nếu nó không được sử dụng ở đâu cả
- IdentityHashMap: một map với các key được so sánh bởi == chứ không phải equals()



# Minh họa

➤ Thực hiện trong công cụ lập trình



# Nội dung tiếp theo

Sử dụng một số collection