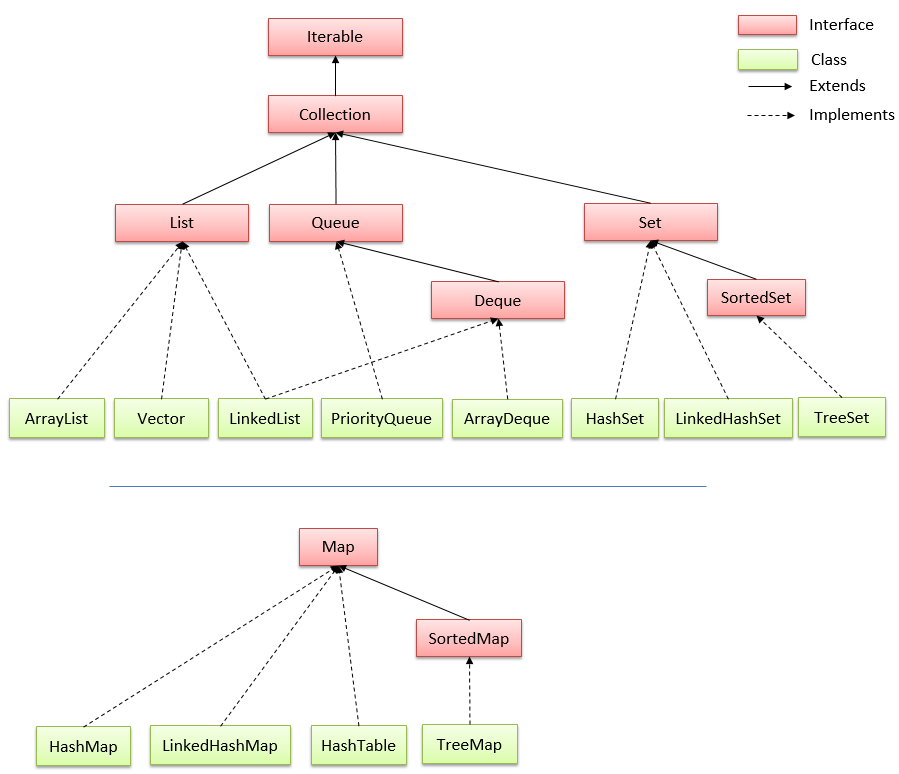
1. **Collections trong java**

Collections framework bao gồm tập các interface và các lớp (class) dùng để hỗ trợ việc thao tác trên tập các đối tượng/ tập danh sách dữ liệu. Java cung cấp interface, class làm việc với collection trong gói **java.util**

Hệ thống cấp bậc Collection trong Java:



Collection trong Java gồm 2 mảng chính là Collection lưu danh sách các đối tượng và Map lưu tập dữ liệu dạng Key, Value.

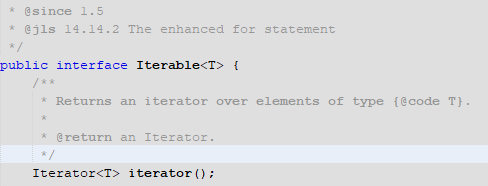
Các interface chính:

|  |  |  |
| --- | --- | --- |
| STT | Interface | Đặc điểm |
| 1 | List (danh sách) | * Là collection có thứ tự. * List có thể chứa các phần tử trùng lặp. * Kiểm soát truy cập vị trí của phần tử trong mảng. Như lấy phần tử, thêm, xóa phần tử tại vị trí index bất kỳ. |
| 2 | Set (tập hợp) | * Là tập hợp không tuần tự * Không cho phép chứa các phần tử trùng nhau * Không thể truy cập Set theo index vì nó không tuần tự. Chỉ có thể thêm, xóa phần tử và tìm khi nó tuần tự |
| 3 | Queue (hàng đợi) | * Là tập hợp tuần tự, nhưng chỉ có thể lấy được phần tử ở đầu hàng đợi. * FIFO vào trước, ra trước. Các phần tử mới sẽ được chèn vào cuối hàng đợi. Lấy các phần tử đầu tiên của hàng đợi. * Không thể truy cập phần tử dựa vào index * Queue có thể chứa các phần tử trùng lặp, nhưng không cho phép chứa phần tử null |
| 4 | Deque (Queue 2 đầu) | * Tương tự queue nhưng có thể chèn và lấy phần tử cả 2 đầu * FIFO vào trước, ra trước, LIFO vào sau, ra trước |
| 5 | Map | * Tập các phần tử, mỗi phần tử ánh xạ cặp key-value, mỗi key ứng với 1 value. * Map không chứa key trùng lặp. * Mỗi key chỉ ánh xạ tới 1 value. * Truy xuất phần từ từ giá trị key |
| 6 | SortedSet ( tập hợp được sắp xếp) | * Con của Set * Chứa các phần tử có thứ tự tăng dần |
| 7 | SortedMap ( bản đồ được sắp xếp) | * Con của Map * Chứa các phần tử có sắp xếp tăng dần của key |

## 2. Giới thiệu về interface Collections:

### 2.1 Interface root Iterable

Inteface cung cấp phương thức để cho phép lặp, duyệt phần tử trong mảng: Iterator

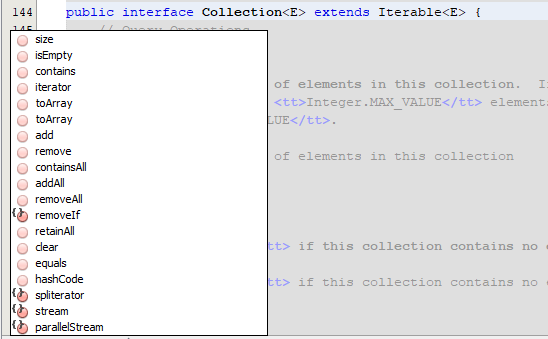


Interface Iterator chứa các phương thức:

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| public boolean hasNext() | Trả về true nếu iterator còn phần tử kế tiếp phần tử đang duyệt. |
| public object next() | Trả về phần tử hiện tại và di chuyển con trỏ trỏ tới phần tử tiếp theo. |
| public void remove() | Loại bỏ phần tử cuối được trả về bởi Iterator. |

### 2.2 Interface Collection

Cung cấp danh sách method:



|  |  |
| --- | --- |
| Phương thức | Mô tả |
| public boolean add(Object element) | Thêm một phần tử vào collection. |
| public boolean addAll(Collection c) | Thêm các phần tử collection được chỉ định vào collection gọi phương thức này. |
| public boolean remove(Object element) | Xóa phần tử ra khỏi collection. |
| public boolean removeAll(Collection c) | Xóa tất cả các phần tử từ collection được chỉ định ra khỏi collection gọi phương thức này. |
| public boolean retainAll(Collection c) | Xóa tất cả các thành phần từ collection gọi phương thức này ngoại trừ collection được chỉ định. |
| public int size() | Trả về tổng số các phần tử trong collection. |
| public void clear() | Xóa tất cả các phần tử trong Collection, sau khi thực hiện phương thức này, Collection sẽ rỗng (empty) |
| public boolean contains(Object element) | Kiểm tra một phần tử có nằm trong Collection không |
| public boolean containsAll(Collection c) | Kiểm tra một Collection có chứa tất cả các phần tử của một Collection khác |
| public Iterator iterator() | Trả về một iterator. |
| public Object[] toArray() | Chuyển đổi collection thành mảng (array). |
| public boolean isEmpty() | Kiểm tra Collection có rỗng hay không. |
| public boolean equals(Object element) | So sánh 2 collection. |
| public int hashCode() | Trả về số hashcode của collection. |

### 2.3 Interface Map

|  |  |
| --- | --- |
| Phương thức | Mô tả |
| Object put(Object key, Object value) | Thêm một cặp key/value (entry) vào map |
| void putAll(Map map) | Thêm các tất cả các cặp key/value được chỉ định vào map gọi phương thức này. |
| Object remove(Object key) | Xóa một entry của map dựa vào key được chỉ định. |
| Object get(Object key) | Get value của map entry dựa vào key được chỉ định. |
| boolean containsKey(Object key) | Kiểm tra map có tồn tại key được chỉ định. |
| Set keySet() | Lấy tập hợp tất cả các key của map. |
| Set entrySet() | Lấy tập hợp các cặp key/value (entry) của map |

## 3. List Interface và các class thực thi của nó

### 3.1. List Interface

**List** là một interface trong java. Nó chứa các phương thức để chèn và xóa các phần tử dựa trên chỉ số index.



Các phương thức của List

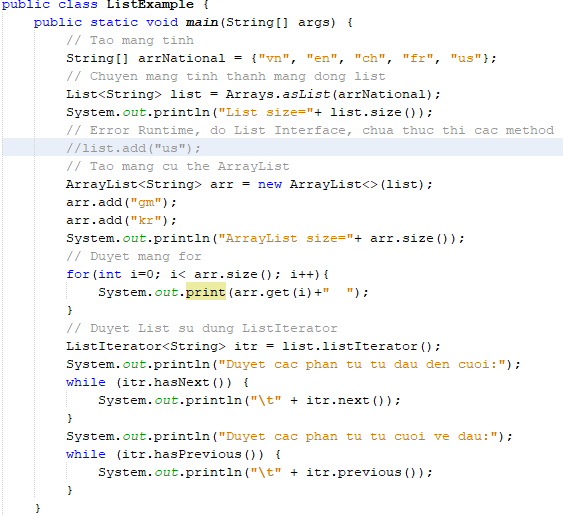
|  |  |
| --- | --- |
| **Method** | **Description** |
| void add(int index,Object element) | Nó được sử dụng để chèn các phần tử vào list tại chỉ số index. |
| boolean addAll(int index,Collection c) | Nó được sử dụng để chèn tất cả các yếu tố của c vào danh sách tại chỉ số index. |
| object get(int index) | Nó được sử dụng để trả về đối tượng được lưu trữ tại chỉ số index trong list. |
| object set(int index,Object element) | Nó được sử dụng để gán phần tử cho vị trí được chỉ định index trong list. |
| object remove(int index) | Nó được sử dụng để xóa các phần tử tại vị trí có chỉ số index và trả về phần tử đã xóa. |
| ListIterator listIterator() | Nó được sử dụng để trả về một Iterator mà bắt đầu từ phần tử đầu tiên của list. |
| ListIterator listIterator(int index) | Nó được sử dụng để trả về một Iterator mà phần tử bắt đầu từ chỉ số index chỉ định. |

* ListIterator trong java

ListIterator là một interface được sử dụng để duyệt các phần tử của List trong java.

|  |  |
| --- | --- |
| **Method** | **Description** |
| boolean hasNext() | Phương pháp này trả về true nếu list interator có tồn tại phần tử kế tiếp phần tử hiện tại. |
| Object next() | Phương thức này trả về phần tử kế tiếp trong danh sách và vị trí con trỏ tăng lên 1. |
| boolean hasPrevious() | Phương pháp này trả về true nếu list interator có tồn tại phần tử kế sau phần tử hiện tại. |
| Object previous() | Phương thức này trả về phần tử kế sau trong danh sách và vị trí con trỏ giảm đi 1. |

VD:

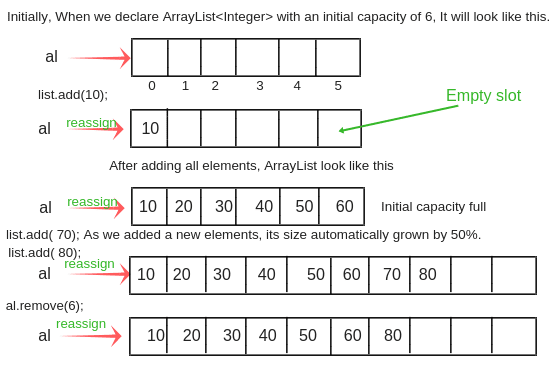


### 3.2 Lớp ArrayList

Lớp ArrayList trong java là một lớp kế thừa lớp AbstractList và triển khai của List Interface trong Collections Framework nên nó sẽ có một vài đặc điểm và phương thức tương đồng với List. ArrayList được sử dụng như một mảng động để lưu trữ các phần tử.



Chú ý: Lớp ArrayList trong java, thao tác chậm vì cần nhiều sự dịch chuyển nếu bất kỳ phần tử nào bị xoá khỏi danh sách.



* Cách khởi tạo đối tượng ArrayList

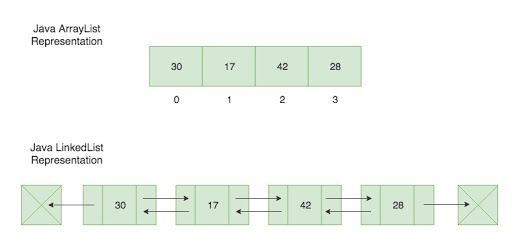
|  |  |
| --- | --- |
| ArrayList arr = new ArrayList(); | khởi tạo một danh sách mảng trống |
| ArrayList arr = new ArrayList(Collection c); | khởi tạo một danh sách mảng được khởi tạo với các phần tử của collection c. |
| ArrayList arr = new ArrayList(int capacity); | khởi tạo một danh sách mảng mà có sức chứa (compacity) ban đầu được chỉ định. Nếu không chỉ định, mặc định là 10. Mỗi lần thêm một phần tử vào danh sách, nếu vượt quá sức chứa cho phép thì danh sách sẽ tự động tăng thêm 50% kích thước hiện có. |

* Phương thức ArrayList

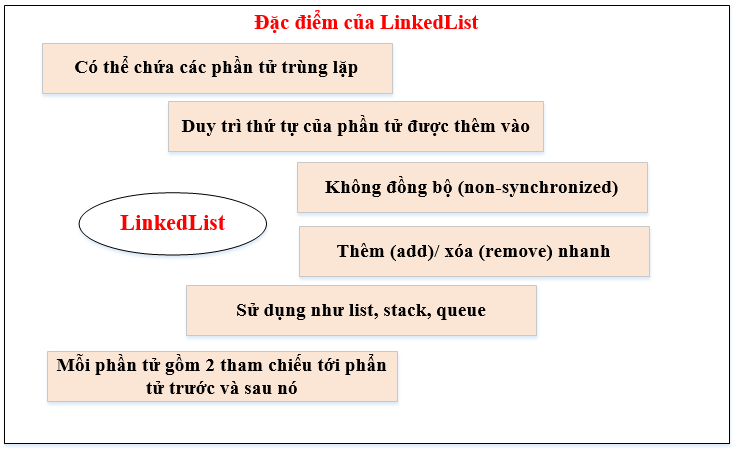
|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| boolean add(Object o) | Nó được sử dụng để nối thêm phần tử được chỉ định vào cuối một danh sách. |
| void add(int index, Object element) | Nó được sử dụng để chèn phần tử element tại vị trí index vào danh sách. |
| boolean addAll(Collection c) | Nó được sử dụng để nối tất cả các phần tử trong collection c vào cuối của danh sách, theo thứ tự chúng được trả về bởi bộ lặp iterator. |
| boolean addAll(int index, Collection c) | Nó được sử dụng để chèn tất cả các phần tử trong collection c vào danh sách, bắt đầu từ vị trí index. |
| void retainAll(Collection c) | Nó được sử dụng để xóa những phần tử không thuộc collection c ra khỏi danh sách. |
| void removeAll(Collection c) | Nó được sử dụng để xóa những phần tử thuộc collection c ra khỏi danh sách. |
| int indexOf(Object o) | Nó được sử dụng để trả về chỉ mục trong danh sách với sự xuất hiện đầu tiên của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa phần tử này. |
| int lastIndexOf(Object o) | Nó được sử dụng để trả về chỉ mục trong danh sách với sự xuất hiện cuối cùng của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa phần tử này. |
| Object[] toArray() | Nó được sử dụng để trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự. |
| Object[] toArray(Object[] a) | Nó được sử dụng để trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự. |
| Object clone() | Nó được sử dụng để trả về một bản sao của ArrayList. |
| void clear() | Nó được sử dụng để xóa tất cả các phần tử từ danh sách này. |
| void trimToSize() | Nó được sử dụng để cắt dung lượng của thể hiện ArrayList này là kích thước danh sách hiện tại. |
| boolean contains(element) | Kết quả trả về là true nếu tìm thấy element trong danh sách, ngược lại trả về false. |

### 3.3 Lớp LinkedList

Lớp LinkedList trong java sử dụng cấu trúc danh sách liên kết Doubly Linked List để lưu trữ các phần tử. 1 phần tử trong Linkedlist chứa 3 thành phần: data, con trỏ next ( giá trị địa chỉ của node sau), con trỏ prev ( giá trị địa chỉ của node trước).



* Đặc điểm của LinkedList



* Khởi tạo LinkedList

|  |  |
| --- | --- |
| LinkedList arr = new LinkedList(); | khởi tạo một danh sách mảng trống |
| LinkedList arr = new LinkedList(Collection c); | khởi tạo một danh sách mảng được khởi tạo với các phần tử của collection, theo thứ tự chúng được trả về bởi iterator của collection. |

* Các phương thức của LinkedList

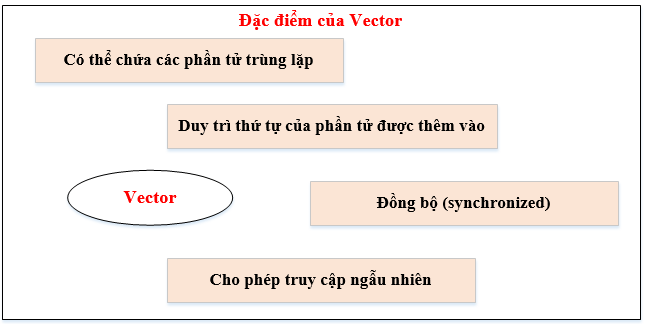
|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| boolean add(Object o) | Nó được sử dụng để nối thêm phần tử được chỉ định vào cuối một danh sách. |
| void add(int index, Object element) | Nó được sử dụng để chèn các phần tử được chỉ định tại các chỉ số vị trí quy định trong một danh sách. |
| void addFirst(Object o) | Nó được sử dụng để chèn phần tử được chỉ định vào đầu danh sách. |
| void addLast(Object o) | Nó được sử dụng để chèn phần tử được chỉ định vào cuối danh sách. |
| int size() | Nó được sử dụng để trả lại số lượng các phần tử trong một danh sách |
| boolean contains(Object o) | Nó được sử dụng để trở về *true* nếu danh sách có chứa một phần tử được chỉ định. |
| boolean remove(Object o) | Nó được sử dụng để xóa phần tử được chỉ định đầu tiên trong một danh sách. |
| Object getFirst() | Nó được sử dụng để trả về phần tử đầu tiên trong một danh sách. |
| Object getLast() | Nó được sử dụng để trả lại phần tử cuối cùng trong một danh sách. |
| int indexOf(Object o) | Nó được sử dụng để trả về chỉ mục trong một danh sách với sự xuất hiện đầu tiên của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa bất kỳ phần tử nào. |
| int lastIndexOf(Object o) | Nó được sử dụng để trả lại chỉ mục trong danh sách với sự xuất hiện cuối cùng của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa bất kỳ phần tử nào. |
| boolean contains(element) | Kết quả trả về là true nếu tìm thấy element trong danh sách, ngược lại trả về false. |

* VD:

|  |
| --- |
| LinkedList<String> llNational = **new** LinkedList<>(); llNational.add(**"vn"**); llNational.add(**"cn"**); llNational.add(**"kr"**); llNational.add(**"fr"**); System.***out***.println(**"Duyet cac phan tu cua Linnkedlist: "**); Iterator<String> ite = llNational.iterator(); **while** (ite.hasNext()){  System.***out***.print(ite.next() + **" "**); } System.***out***.println(**"\nPhan tu thu 4:"**+ llNational.get(3)); |

### 3.4 Vector

Có các đặc điểm của List interface, gần giống ArrayList



* Khởi tạo Vector

|  |  |
| --- | --- |
| Vector arr = new Vector(); | khởi tạo một danh sách mảng trống |
| Vector arr = new Vector(Collection c); | khởi tạo một danh sách mảng được khởi tạo với các phần tử của collection. |
| Vector arr = new Vector(int capacity); | khởi tạo một danh sách mảng mà có sức chứa (compacity) ban đầu được chỉ định. Nếu không chỉ định, mặc định là 10. Mỗi lần thêm một phần tử vào danh sách, nếu vượt quá sức chứa cho phép thì danh sách sẽ tự động tăng thêm 100% kích thước hiện có. |

* Các phương thức của Vector

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| boolean add(Object o) | Thêm phần tử được chỉ định vào cuối một danh sách. |
| void add(int index, Object element) | Chèn một phần tử được chỉ định tại vị trí (index) được chỉ định vào danh sách. Ném IndexOutOfBoundsException nếu index này ở bên ngoài dãy (index < 0 hoặc index > size()). |
| boolean addAll(Collection c) | Thêm tất cả các phần tử trong collection được chỉ định vào cuối của danh sách gọi phương thức, theo thứ tự chúng được trả về bởi bộ lặp iterator. |
| boolean addAll(int index, Collection c) | Thêm tất cả các phần tử trong collection được chỉ định vào danh sách gọi phương thức, bắt đầu từ vị trí đã chỉ định. Ném NullPointerException nếu collection đã cho là null. |
| Object get(int index) | Trả về phần tử tại index đã cho. Ném IndexOutOfBoundsException nếu index đã cho là ở bên ngoài dãy (index < 0 hoặc index >= size()). |
| int indexOf(Object o) | Lấy vị trí (index) trong danh sách với sự xuất hiện đầu tiên của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa phần tử này. |
| int lastIndexOf(Object o) | Lấy vị trí (index) trong danh sách với sự xuất hiện cuối cùng của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa phần tử này. |
| Object remove(int index) | Gỡ bỏ phần tử tại index đã cho. Ném IndexOutOfBoundsException nếu index ở ngoài dãy (index < 0 hoặc index >= size()). |
| void retainAll(Collection c) | Xóa những phần tử không thuộc collection c và không thuộc list hiện tại khỏi list hiện tại. Ném NullPointerException nếu collection đã cho là null. |
| void removeAll(Collection c) | Xóa những phần tử thuộc collection c và thuộc list hiện tại khỏi list hiện tại. Ném NullPointerException nếu collection đã cho là null. |
| Object set(int index, Object element) | Thay thế phần tử tại vị trí đã cho trong list này với phần tử đã xác định. Ném IndexOutOfBoundsException nếu index ở ngoài dãy (index < 0 hoặc index >= size()). |
| Object[] toArray() | Chuyển một danh sách sang mảng và trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự. Ném NullPointerException nếu mảng này là null. |
| Object[] toArray(Object[] a) | Chuyển một danh sách sang mảng và trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự. Kiểu runtime là của mảng trả về giống như mảng đã xác định. |
| Object clone() | Tạo một bản sao của Vector. |
| void clear() | Xóa tất cả các phần tử từ danh sách. |
| void trimToSize() | Cắt dung lượng của thể hiện Vector này là kích thước danh sách hiện tại. |

* VD:

|  |
| --- |
| Vector<String> vNational = **new** Vector<>(); vNational.add(**"vn"**); vNational.add(**"cn"**); vNational.add(**"kr"**); vNational.add(**"fr"**); System.***out***.println(**"Duyet cac phan tu cua Vector: "**); Iterator<String> ite = vNational.iterator(); **while** (ite.hasNext()){  System.***out***.print(ite.next() + **" "**); } System.***out***.println(**"\nPhan tu thu 4:"**+ vNational.get(3)); |

### 3.5 So sánh Array, ArrayList, LinkedList, Vector:

|  |  |  |  |
| --- | --- | --- | --- |
| **Array** | **ArrayList** | **LinkedList** | **Vector** |
| Là mảng nguyên thủy, chỉ có thuộc tính length | Là class implement List, có các method hỗ trợ get/add/remove/… | | |
| Kích thước cố định | Tăng 50% kích thước khi đạt giới hạn | Kích thước động | Tăng 100% kích thước khi đạt giới hạn |
| Lưu kiểu dữ liệu nguyên thủy và đối tượng | Chỉ lưu kiểu dữ liệu đối tượng, từ Java 5 kiểu nguyên thủy được tự động chuyển đổi thành đối tượng để lưu trữ (auto-boxing) | | |
| Tốc độ lưu trữ và truy xuất nhanh | Tốc độ truy xuất theo index nhanh | Tốc độ truy xuất theo index chậm | Tốc độ truy xuất theo index nhanh |
| Rất kém trong việc insert/remove phần tử ở giữa mảng | Tốc độ insert/remove phần tử trong mảng trung bình | Tốc độ insert/remove phần tử trong mảng nhanh | Tốc độ insert/remove phần tử trong mảng trung bình |
| Không hỗ trợ synchronized, dễ mất dữ liệu khi xử lý đa luồng nhưng nhanh | Không hỗ trợ synchronized, dễ mất dữ liệu khi xử lý đa luồng nhưng nhanh | Không hỗ trợ synchronized, dễ mất dữ liệu khi xử lý đa luồng nhưng nhanh | Hỗ trợ synchronized, an toàn dữ liệu khi xử lý đa luồng nhưng chậm |
| Thường dùng với các thao tác bổ sung vào cuối mảng và truy xuất thông thường, sử dụng khi nghiệp vụ đơn giản ít nâng cấp | Sử dụng để lưu trữ và truy xuất dữ liệu cơ bản, ít khi thay đổi | Sử dụng khi cần thao tác thay đổi dữ liệu nhiều, cho các nghiệp vụ phức tạp | Sử dụng khi cần thao tác với nhiều luồng xử lý |

## 4. Set Interface và class thực thi

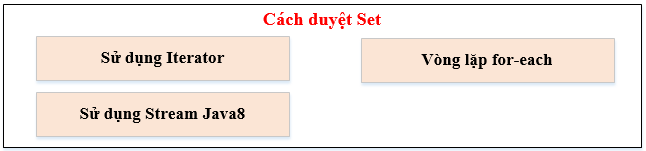
### 4.1 Set Interface

|  |  |
| --- | --- |
| **Set là một interface kế thừa Collection interface trong java và không thể chứa các phần tử trùng lặp.** | |
| **HashSet** | lưu trữ các phần tử của nó trong bảng băm, hay dùng, tuy nhiên nó không đảm bảo về thứ tự các phần tử được chèn vào |
| **TreeSet** | lưu trữ các phần tử của nó trong một cây, sắp xếp các phần tử dựa trên các giá trị của chúng, về cơ bản là chậm hơn HashSet |
| **LinkedHashSet** | được triển khai dưới dạng bảng băm với có cấu trúc dữ liệu danh sách liên kết, sắp xếp các phần tử của nó dựa trên thứ tự chúng được chèn vào tập hợp (thứ tự chèn). |
| **EnumSet** | là abstract class chuyên biệt để sử dụng với các kiểu enum |

* Các phương thức của Set

|  |  |
| --- | --- |
| **Method** | **Description** |
| boolean add(Object element) | chèn các phần tử vào set. |
| boolean addAll(Collection c) | chèn tất cả các phần tử của c vào set. |
| void clear() | Xóa tất cả các phần tử khỏi set. |
| boolean contains(Object element) | Trả về true nếu tập hợp này chứa phần tử đã chỉ định. |
| boolean containsAll(Collection c) | Trả về true nếu set chứa tất cả các phần tử của collection c đã chỉ định. |
| boolean equals(Object o) | So sánh các đối tượng được chỉ định với set. |
| boolean isEmpty() | Trả về true nếu set không chứa phần tử. |
| int hashCode() | Trả về giá trị mã băm |
| Iterator iterator() | Trả về một trình vòng lặp iterator để duyệt qua các phần tử của set. |
| boolean remove(Object o) | Xóa phần tử đã chỉ định khỏi set. |
| boolean removeAll(Collection c) | Xóa khỏi set tất cả các phần tử của nó được chứa trong collection c đã chỉ định. |
| boolean retainAll(Collection c) | Chỉ giữ lại các phần tử trong set được chứa trong collection c đã chỉ định. |
| int size() | Trả về số lượng các phần tử của set. |
| Object[] toArray() | Trả về một mảng chứa tất cả các phần tử trong set. |
| T[] toArray(T[] a) | Trả về một mảng chứa tất cả các phần tử trong set, kiểu run-time của mảng trả về là kiểu đã chỉ định. |

* Cách duyệt tập Set



* VD:

|  |
| --- |
| *// Tao HashSet* Set<String> setNational = **new** HashSet<>(); setNational.add(**"vn"**); setNational.add(**"cn"**); setNational.add(**"kr"**); setNational.add(**"fr"**);  *// Tao TreeSet* Set<String> setString = **new** TreeSet<>(vNational); setString.add(**"ls"**); *// Tao LinkedHashSet* Set<String> setLinked = **new** LinkedHashSet<>(); setLinked.add(**"vn"**); setLinked.add(**"cn"**); setLinked.add(**"kr"**); setLinked.add(**"fr"**); *// xoa phan tu* setNational.remove(**"fr"**); *// Tao EnumSet* Set<days> set = EnumSet.*of*(days.***MONDAY***, days.***TUESDAY***, days.***WEDNESDAY***); *// Duyet EnumSet* Iterator<days> iter = set.iterator(); **while** (iter.hasNext())  System.***out***.println(iter.next()); Set<days> set1 = EnumSet.*allOf*(days.**class**); System.***out***.println(**"Days trong tuan:"** + set1); Set<days> set2 = EnumSet.*noneOf*(days.**class**); System.***out***.println(**"Cac ngay trong tuan:"** + set2); *// Duyet for -each* System.***out***.println(**"Duyet for -each: "**); **for** (String national: setNational){  System.***out***.println(national+ **" "**); } *// Duyet Iterator* System.***out***.println(**"Duyet Iterator: "**); Iterator<String> iterator = setNational.iterator(); **while** (iterator.hasNext()){  System.***out***.println(iterator.next()+ **" "**); } System.***out***.println(**"Duyet parallelStream: "**); setNational.parallelStream().forEach(str -> {  System.***out***.println(str+ **" "**); }); |

## 5. Map Interface và class thực thi

|  |  |
| --- | --- |
| **Map được sử dụng để lưu trữ và truy xuất dữ liệu theo cặp khóa (key) và giá trị (value).  - Mỗi cặp key và value được gọi là entry.  - Map chỉ chứa các giá trị key duy nhất, không chứa các key trùng lặp** | |
| **HashMap** | HashMap lưu trữ dữ liệu dưới dạng cặp key và value - chứa các key duy nhất. - có thể có 1 key là null và nhiều giá trị null. - không đảm bảo thứ tự các entry được thêm vào. HashMap<K,V> Trong đó   K: đây là kiểu key để lưu trữ.  V: đây là kiểu giá trị được ánh xạ. |
| **LinkedHashMap** | LinkedHashMap lưu trữ dữ liệu dưới dạng cặp key và value. - chứa các key duy nhất. - có thể có 1 key là null và nhiều giá trị null. - đảm bảo thứ tự các entry được thêm vào. LinkedHashMap<K,V> Trong đó:   K: đây là kiểu key để lưu trữ.  V: đây là kiểu giá trị được ánh xạ. |
| **TreeMap** | LinkedHashMap lưu trữ dữ liệu dưới dạng cặp key và value. - chứa các key duy nhất. - KHÔNG cho phép bất kỳ key nào là null và nhưng có thể có nhiều giá trị null. - duy trì các phần tử theo thứ tự chèn, dựa vào bộ so sánh Comparator TreeMap<K,V> Trong đó:   K: đây là kiểu key để lưu trữ.  V: đây là kiểu giá trị được ánh xạ. |

* Các phương thức của Map

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| void clear() | Gỡ bỏ tất cả cặp key/value từ Map đang gọi |
| boolean containsKey(Object key) | Trả về true nếu Map đang gọi chứa k như là một key. Nếu không là false |
| boolean containsValue(Object v) | Trả về true nếu Map đang gọi chứa v như là một value. Nếu không là false |
| boolean equals(Object obj) | Trả về true nếu obj là một Map và chứa cùng các Entry. Nếu không là false. |
| Object get(Object key) | Trả về value mà liên kết với key |
| int hashCode() | Trả về hash code cho Map đang gọi |
| boolean isEmpty() | Trả về true nếu Map đang gọi là trống, nếu không là false |
| Object put(Object key, Object value) | Đặt một entry vào Map đang gọi, ghi đè bất kỳ value trước mà liên kết với key. Với key và value tương ứng là k và v. Trả về null nếu key đã không tồn tại. Nếu không thì, value trước mà liên kết với key được trả về |
| void putAll(Map map) | Đặt tất cả entry từ m vào trong Map này |
| Object remove(Object key) | Gỡ bỏ entry mà có khóa là key được chỉ định. |
| int size() | Trả về số các cặp key/value trong Map |
| Collection values( ) | Trả về một tập hợp chứa các value trong Map. Phương thức này cung cấp một collection-view của các value trong Map |
| Set keySet() | Nó được sử dụng để trả đối tượng Set có chứa tất cả các keys. |
| Set entrySet() | Nó được sử dụng để trả lại đối tượng Set có chứa tất cả các keys và values. |

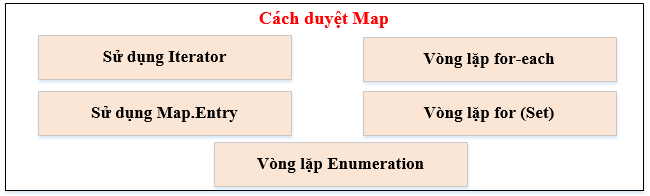
* Map.Entry Interface

Entry là một interface con của Map, định nghĩa/ mô phỏng 1 phần tử trong Map. Vì vậy, để truy cập phần tử dùng Map.Entry. Nó cung cấp các phương pháp để truy xuất các key và value.

Các phương thức của Map.Entry

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| Object getKey() | Nó được dùng để lấy key. |
| Object getValue() | Nó được sử dụng để lấy value. |

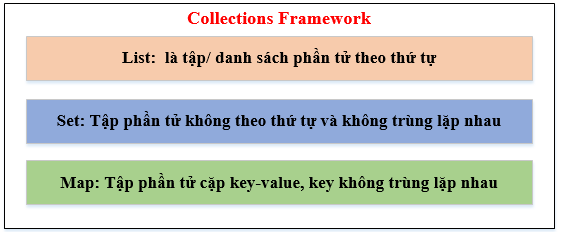
* Các duyệt phần tử của Map



* VD:

|  |
| --- |
| *// Tạo map* Map<Integer, String> mapHash = **new** HashMap<>(); mapHash.put(1,**"vn"**); mapHash.put(2,**"cn"**); mapHash.put(3,**"fr"**); Map<Integer, String> mapLink = **new** LinkedHashMap<>(); mapLink.put(1,**"vn"**); mapLink.put(2,**"cn"**); mapLink.put(3,**"fr"**); Map<Integer, String> mapTree = **new** TreeMap<>(); mapTree.put(1,**"vn"**); mapTree.put(2,**"cn"**); mapTree.put(3,**"fr"**); *// remove phan tu* mapHash.remove(3); mapLink.remove(2); *// them phan tu* mapHash.put(4,**"abc"**);  *// Su dung Set* System.***out***.println(**"Duyet theo Set:"**); Set<Integer> setMap = mapHash.keySet(); **for** (**int** key: setMap){  System.***out***.println(key + **": "**+ mapHash.get(key)); } *// Dung Map.Entry* System.***out***.println(**"Duyet theo Map.Entry:"**); **for** (Map.Entry<Integer, String> entry: mapHash.entrySet()){  System.***out***.println(entry.getKey() + **": "**+ entry.getValue()); } *// Dung Iterator* System.***out***.println(**"Duyet theo Iterator:"**); Iterator<Map.Entry<Integer, String>> iterator1 = mapHash.entrySet().iterator(); **while** (iterator1.hasNext()){  Map.Entry<Integer, String> iter = iterator1.next();  System.***out***.println(iter.getKey() + **": "**+ iter.getValue()); } *// Dung foreach* System.***out***.println(**"Duyet theo foreach:"**); mapHash.forEach((num, value)-> {  System.***out***.println(num + **": "**+ value); }); *// Dung Enumeration* System.***out***.println(**"Duyet theo Enumeration:"**); Enumeration<Integer> enumeration = Collections.*enumeration*(mapHash.keySet()); **while** (enumeration.hasMoreElements()){  **int** key = enumeration.nextElement();  System.***out***.println(key + **": "**+ mapHash.get(key)); } |

6. Tổng kết



1. **Bài tập**
2. Nhập ArrayList tập n số nguyên dương từ bàn phím:

* Duyệt mảng bằng nhiều cách khác nhau
* Bỏ các thành phần trùng nhau, hay hiển thị list chỉ gồm các phần tử không trùng nhau
* Hiển thị số lần xuất hiện của các phần tử trong list
* Nhập 1 số từ bàn phím, hiển thị các chỉ số (index) của phần tử ấy trong mảng
* Đảo ngược và xáo trộn mảng sử dụng Collections.

1. Tạo đối tượng sinh viên có các thuộc tính mã (int, không trùng nhau) -> SET, tên, ngày sinh, địa chỉ, giới tính, mã ngành học (String); Tạo danh sách Sinh viên -> LIST

* Thêm sinh viên vào danh sách Sinh viên. -> ADD SET & ADD LIST
* Sắp xếp danh sách Sinh viên theo thứ tự tăng dần tên và mã ngành học -> COMPARATOR & COLLECTIONS
* Hiển thị tên sinh viên không trùng nhau -> Sử dụng SET của tên sv
* **Hiển thị các sinh viên tìm kiếm theo mã ngành học -> Sử dụng Map -> chuyển từ LIST sinh viên** -> MAP ( MÃ NGANH HỌC, ARRAYLIST<SINHVIEN>)
* Xóa sinh viên khi nhập vào tên và mã ngành học