Java Core

1. **Java.util**
   1. **List**

Khởi tạo:

List<TYPE> NAME = **new** ArrayList<TYPE>();

Ví dụ:

List<String> newList = **new** ArrayList<String>();

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Demo\_List {

**public** **static** **void** main(String[] args) {

List<String> newList = **new** ArrayList<String>();

newList.add("A");

newList.add("C");

newList.add(1,"B");

List<String> secondList = **new** ArrayList<String>();

secondList.add("D");

newList.addAll(2,secondList);

newList.forEach(newLists->

{

System.***out***.println(newLists);

});

}

}

* + 1. **Các thủ tục hỗ trợ:**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void add(int index, E element) | Chèn phần tử được chỉ định vào vị trí đã chỉ định trong danh sách. |
| boolean add(E e) | Nối phần tử được chỉ định vào cuối danh sách. |
| boolean addAll(Collection<? extends E> c) | Nối tất cả các thành phần trong Collection đã chỉ định vào cuối danh sách. |
| boolean addAll(int index, Collection<? extends E> c) | Nối thêm tất cả các thành phần trong bộ sưu tập đã chỉ định, bắt đầu từ vị trí đã chỉ định của danh sách. |
| void clear() | Sử dụng để loại bỏ tất cả các yếu tố khỏi danh sách này. |
| boolean equals(Object o) | Sử dụng để so sánh đối tượng được chỉ định với các thành phần của danh sách. |
| int hashcode() | Sử dụng để trả về giá trị mã băm cho một danh sách. |
| E get(int index) | Sử dụng để lấy phần tử từ vị trí cụ thể của danh sách. |
| boolean isEmpty() | Trả về true nếu danh sách trống, ngược lại là false. |
| int lastIndexOf(Object o) | Trả về chỉ mục trong danh sách này về lần xuất hiện cuối cùng của phần tử đã chỉ định hoặc -1 nếu danh sách không chứa phần tử này. |
| Object[] toArray() | Trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự. |
| T[] toArray(T[] a) | Trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự. |
| boolean contains(Object o) | Trả về true nếu danh sách chứa phần tử đã chỉ định. |
| boolean containsAll(Collection<?> c) | Trả về true nếu danh sách chứa tất cả phần tử đã chỉ định |
| int indexOf(Object o) | Trả về chỉ mục trong danh sách này về lần xuất hiện đầu tiên của phần tử đã chỉ định hoặc -1 nếu Danh sách không chứa phần tử này. |
| E remove(int index) | loại bỏ phần tử có mặt tại vị trí đã chỉ định trong danh sách. |
| boolean remove(Object o) | Loại bỏ sự xuất hiện đầu tiên của phần tử được chỉ định. |
| boolean removeAll(Collection<?> c) | Loại bỏ tất cả các yếu tố khỏi danh sách. |
| void replaceAll(UnaryOperator operator) | Thay thế tất cả các yếu tố từ danh sách bằng các yếu tố được chỉ định. |
| void retainAll(Collection<?> c) | Giữ lại tất cả các yếu tố trong danh sách có trong bộ sưu tập được chỉ định. |
| E set(int index, E element) | Thay thế phần tử được chỉ định trong danh sách, có mặt tại vị trí đã chỉ định. |
| void sort(Comparator<? super E> c) | Sắp xếp các thành phần của danh sách trên cơ sở so sánh được chỉ định. |
| Spliterator spliterator() | Tạo spliterator trên các phần tử trong danh sách. |
| List<E> subList(int fromIndex, int toIndex) | Tìm nạp tất cả các yếu tố nằm trong phạm vi nhất định. |
| int size() | Trả về số lượng phần tử có trong danh sách. |

* 1. **Map, HashMap, HashTable:**
     1. **Khởi tạo:**

Map<K, V> Name = **new** HashMap<K, V>();

Map<Integer, String> newMap = **new** HashMap<Integer, String>();

**import** java.util.\*;

**import** java.util.Map.Entry;

**public** **class** Demo\_Map {

**public** **static** **void** main(String[] args) {

// Create new map

Map<Integer, String> newMap = **new** HashMap<Integer, String>();

// Put element to map at 0

newMap.put(0, "First Element");

Set<Entry<Integer, String>> set=newMap.entrySet();

Iterator<Entry<Integer, String>> itr=set.iterator();

**while**(itr.hasNext()){

Entry<Integer, String> entry=itr.next();

System.***out***.println(entry.getKey()+" "+entry.getValue());

}

}

}

* + 1. **Các thủ tục hỗ trợ:**

|  |  |
| --- | --- |
| **Method** | **Description** |
| V put(Object key, Object value) | Được sử dụng để chèn một mục trong bản đồ. |
| void putAll(Map map) | Được sử dụng để chèn Map được chỉ định trong Map. |
| V putIfAbsent(K key, V value) | Chỉ chèn giá trị được chỉ định bằng khóa được chỉ định trong Map nếu chưa được chỉ định. |
| V remove(Object key) | Được sử dụng để xóa một mục nhập cho khóa được chỉ định. |
| boolean remove(Object key, Object value) | Loại bỏ các giá trị được chỉ định bằng các khóa được chỉ định liên quan khỏi Map. |
| Set keySet() | Trả về khung nhìn Set chứa tất cả các phím. |
| Set<Map.Entry<K,V>> entrySet() | Trả về khung nhìn Set chứa tất cả các khóa và giá trị. |
| void clear() | Được sử dụng để thiết lập lại Map. |
| boolean containsKey(Object key) | Phương thức này trả về true nếu một số khóa bằng với khóa tồn tại trong Map, một số khác trả về false. |
| boolean equals(Object o) | Được sử dụng để so sánh Đối tượng được chỉ định với Map. |
| void forEach(BiConsumer<? super K,? super V> action) | Thực hiện hành động đã cho cho từng mục trong Map cho đến khi tất cả các mục đã được xử lý hoặc hành động đưa ra một ngoại lệ. |
| V get(Object key) | Phương thức này trả về đối tượng chứa giá trị được liên kết với khóa. |
| V getOrDefault(Object key, V defaultValue) | Trả về giá trị mà khóa được chỉ định được ánh xạ hoặc defaultValue nếu Map không chứa ánh xạ cho khóa. |
| int hashCode() | Trả về giá trị mã băm cho Map |
| boolean isEmpty() | Phương thức này trả về true nếu Map trống; trả về false nếu chứa ít nhất một khóa. |
| V replace(K key, V value) | Thay thế giá trị được chỉ định cho một khóa được chỉ định. |
| boolean replace(K key, V oldValue, V newValue) | Thay thế giá trị cũ bằng giá trị mới cho một khóa được chỉ định. |
| void replaceAll(BiFunction<? super K,? super V,? extends V> function) | Thay thế giá trị của mỗi mục nhập bằng kết quả của việc gọi hàm đã cho trên mục đó cho đến khi tất cả các mục đã được xử lý hoặc hàm ném ngoại lệ. |
| Collection values() | Trả về chế độ xem bộ sưu tập các giá trị có trong Map. |
| int size() | Phương pháp này trả về số lượng mục trong Map. |

* 1. **Set:**
     1. **Khởi tạo:**

Set<E> Name = **new** HashSet<E>();

Set<String> hash\_Set = **new** HashSet<String>();

**import** java.util.\*;

**public** **class** Demo\_Map {

**public** **static** **void** main(String[] args) {

Set<String> hash\_Set = **new** HashSet<String>();

hash\_Set.add("Geeks");

hash\_Set.add("For");

hash\_Set.add("Geeks");

hash\_Set.add("Example");

hash\_Set.add("Set");

System.***out***.print("Set output without the duplicates");

System.***out***.println(hash\_Set);

System.***out***.print("Sorted Set after passing into TreeSet");

Set<String> tree\_Set = **new** TreeSet<String>(hash\_Set);

System.***out***.println(tree\_Set);

}

}