

Data structure

Nguyễn Anh Tuấn

KTECH
COLLEGE

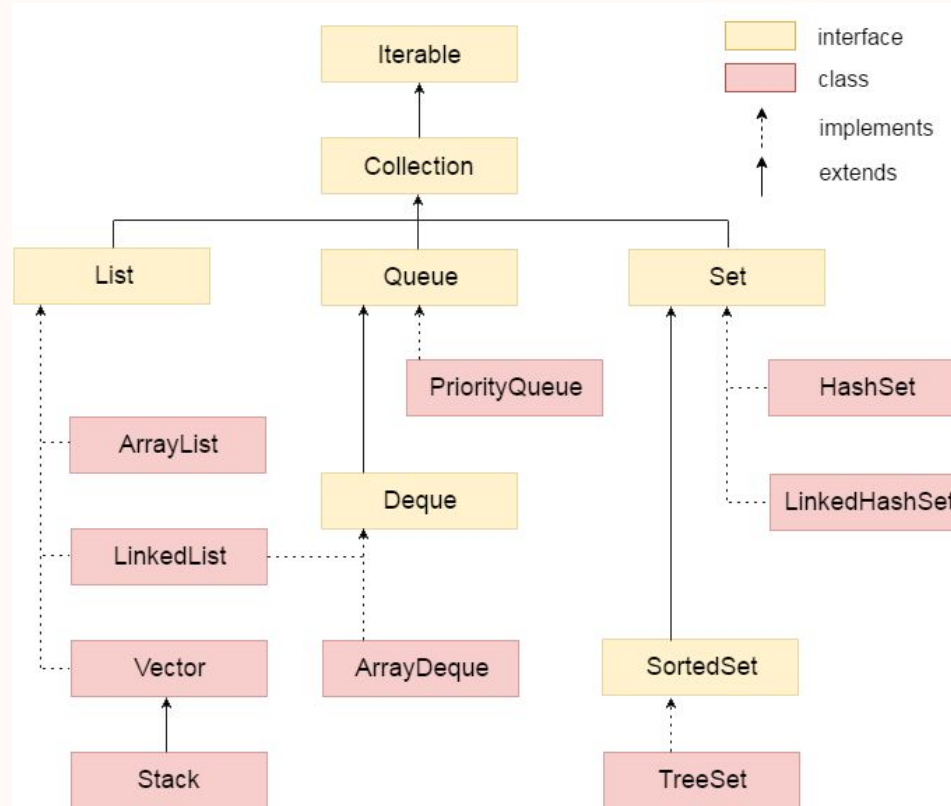


Nội dung bài giảng

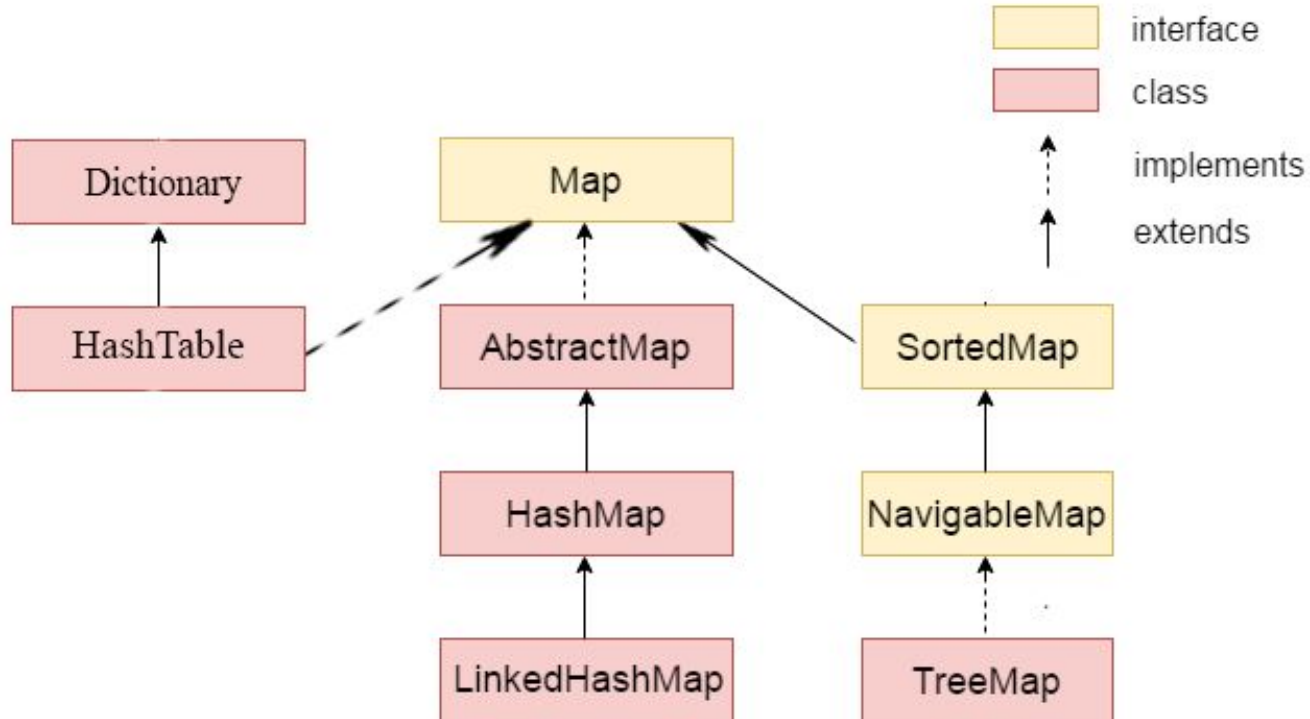
- 1 Phân cấp trong Collection framework
- 2 Hashmap và Hashtable
- 3 Loại bỏ các phần tử trùng trong ArrayList
- 4 Chuyển đổi giữa Array - ArrayList

Phân cấp trong Collection framework

Phân cấp trong Collection framework



Hệ thống phân cấp Collection framework



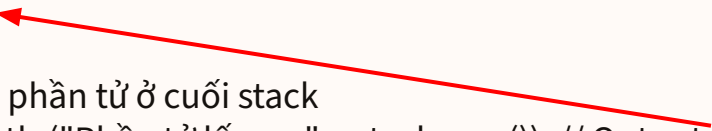
Cấu trúc dữ liệu Stack (ngăn xếp)

- **Stack:**
 - **Stack** (ngăn xếp) trong Java là một cấu trúc dữ liệu tuân theo nguyên tắc **LIFO (Last In First Out)**, tức là phần tử được thêm vào cuối cùng sẽ được lấy ra đầu tiên.
- **Stack** là một cấu trúc dữ liệu hữu ích trong nhiều tình huống khác nhau như:
 - Quản lý call Stack
 - Duyệt đồ thị
 - Kiểm tra biểu thức
 - Tính toán biểu thức
 - Triển khai đệ quy



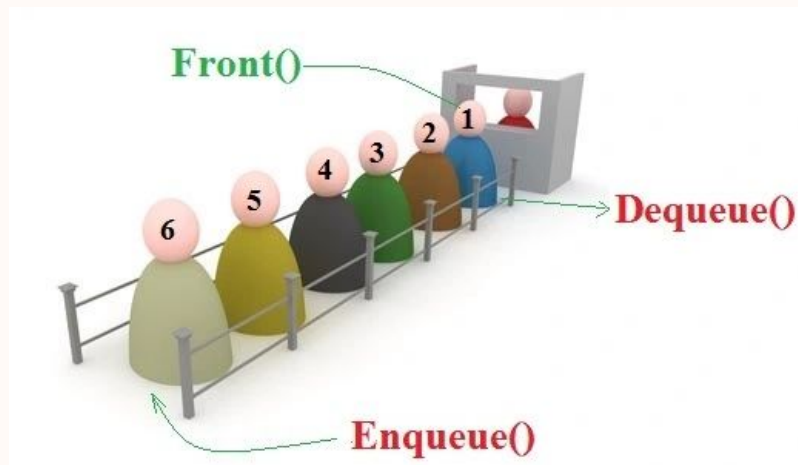
Cấu trúc dữ liệu Stack (ngăn xếp)

```
public static void main(String[] args) {  
    Stack<Integer> stack = new Stack<>();  
  
    // Thêm phần tử vào stack  
    stack.push(1);  
    stack.push(2);  
  
    // Lấy và loại bỏ phần tử ở cuối stack  
    System.out.println("Phần tử lấy ra: " + stack.pop()); // Output: 2  
  
    // Xem phần tử ở đỉnh ngăn xếp  
    System.out.println("Phần tử ở đỉnh stack: " + stack.peek()); // Output: 1  
  
    // Kiểm tra stack có rỗng không  
    System.out.println("Stack có rỗng không: " + stack.isEmpty()); // Output: false  
}
```



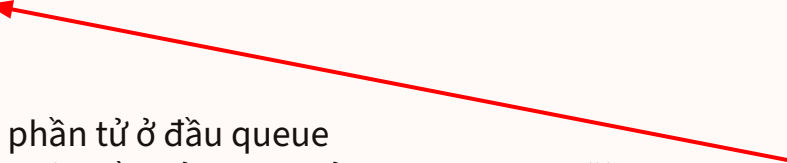
Cấu trúc dữ liệu Queue (hàng đợi)

- **Queue:**
 - **Queue** (hàng đợi) trong Java là một cấu trúc dữ liệu tuân theo nguyên tắc **FIFO (First In First Out)**, tức là phần tử được thêm vào đầu tiên sẽ được lấy ra đầu tiên.
- **Queue** là một cấu trúc dữ liệu hữu ích trong nhiều tình huống khác nhau như:
 - Quản lý công việc (task schedule)
 - Lập lịch CPU (CPU Schedule)
 - Truyền thông tin giữa các Thread
 - Hệ thống in ấn
 - Xử lý Stream dữ liệu



Cấu trúc dữ liệu Queue (hàng đợi)

```
public static void main(String[] args) {  
    Queue<Integer> queue = new LinkedList<>();  
  
    // Thêm phần tử vào queue  
    queue.add(1);  
    queue.add(2);  
  
    // Lấy và loại bỏ phần tử ở đầu queue  
    System.out.println("Phần tử bị loại bỏ: " + queue.poll()); // Output: 1  
  
    // Xem phần tử ở đầu queue  
    System.out.println("Phần tử ở đầu queue: " + queue.peek()); // Output: 2  
  
    // Kiểm tra kích thước của queue  
    System.out.println("Kích thước queue: " + queue.size()); // Output: 1  
}
```



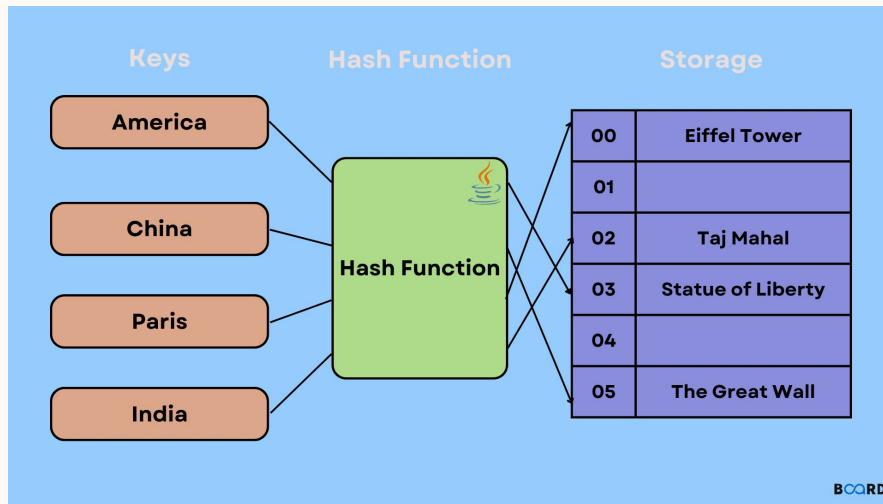
Các cấu trúc dữ liệu phổ biến khác

- **Hashtable:**

- Là một cấu trúc dữ liệu lưu trữ các cặp **key-value**. Nó sử dụng hàm băm để ánh xạ khóa đến một vị trí trong một bảng băm, nơi giá trị tương ứng được lưu trữ.

- **Set:**

- Set là một tập hợp các phần tử không trùng lặp. Nó không cho phép chứa các phần tử giống nhau.



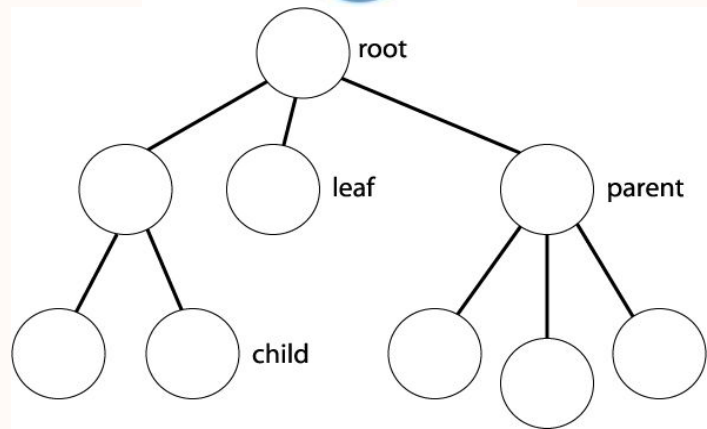
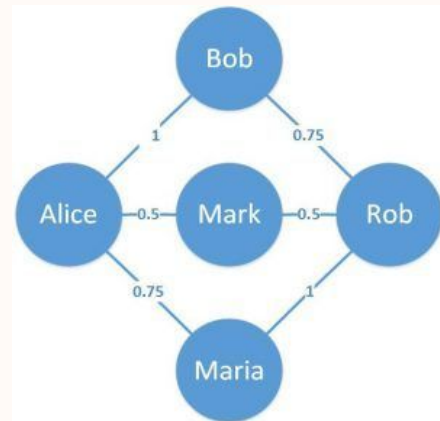
Các cấu trúc dữ liệu phổ biến khác

- **Graph:**

- Đồ thị (**graph**) là một tập hợp các đỉnh (**nodes** hoặc **vertices**) và các cạnh (**edges**) nối các đỉnh lại với nhau. Đồ thị có thể là có hướng (directed) hoặc vô hướng (undirected).

- **Tree:**

- Cây (**tree**) là một cấu trúc dữ liệu phân cấp bao gồm các nút (**nodes**). Cây có một nút gốc (**root**) và các nút con (**children**). Mỗi nút có thể có các nút con và mỗi nút chỉ có một nút cha (**parent**), ngoại trừ nút gốc không có nút cha.

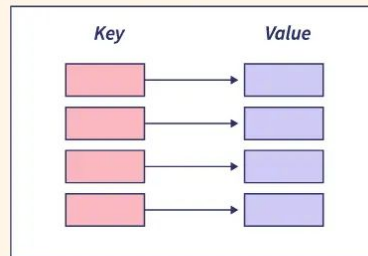


HashMap và Hashtable

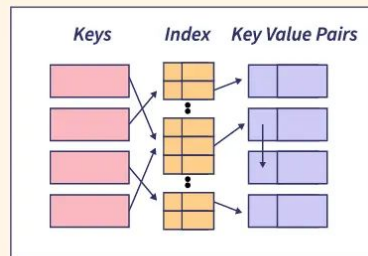
HashMap và Hashtable

- **Giống nhau:**
 - Cả **HashMap** và **Hashtable** đều cài đặt interface **Map**.
 - **HashMap** và **Hashtable** đều được sử dụng để lưu trữ dữ liệu ở dạng cặp **key** và **value**.
 - Cả hai đều đang sử dụng kỹ thuật băm để lưu trữ các khóa duy nhất.

HashMap



Hashtable



HashMap và Hashtable

- **Khác nhau:**

HashMap	HashTable
<ul style="list-style-type: none"> ● HashMap cho phép một key là null và nhiều giá trị null. 	<ul style="list-style-type: none"> ● Hashtable không cho phép bất kỳ key hoặc giá trị null.
<ul style="list-style-type: none"> ● HashMap không đồng bộ. 	<ul style="list-style-type: none"> ● Hashtable là đồng bộ.
<ul style="list-style-type: none"> ● HashMap được ưa thích trong các ứng dụng đơn luồng (single-thread). ● Có thể sử dụng phương thức <code>Collections.synchronizedMap()</code> trong ứng dụng đa luồng (mulit-thread). 	<ul style="list-style-type: none"> ● Hashtable có để sử dụng trong các ứng dụng đa luồng (multi-thread). ● <code>ConcurrentHashMap</code> là lựa chọn tốt hơn Hashtable trong ứng dụng đa luồng.

Phần tử trùng trong ArrayList

Loại bỏ phần tử trùng trong ArrayList

- **ArrayList:**

- **ArrayList** là một trong những Collection được sử dụng nhiều nhất trong java. Nó cung cấp cho sự linh hoạt của việc thêm nhiều phần tử null, phần tử trùng lặp và cũng duy trì thứ tự chèn của các phần tử.

ArrayList of Integer Object Type

2	5	12	1	79	11
0	1	2	3	4	5

Inter type (for all indices) Data

Sử dụng for

```
public static void main(String[] args) {  
    // Khởi tạo 1 danh sách ArrayList có 1 phần tử trùng lặp giá trị Java  
    List<String> listDuplicate = new ArrayList<String>();  
    listDuplicate.add("JAVA");  
    listDuplicate.add("PHP");  
    listDuplicate.add("JAVA");  
  
    // Khởi tạo 1 ArrayList mới để lưu từng phần tử của ArrayList cũ  
    List<String> listNonDuplicate = new ArrayList<String>();  
    for (String element : listDuplicate) {  
        // Kiểm tra nếu có phần tử đã đc thêm thì bỏ qua  
        if (!listNonDuplicate.contains(element))  
            listNonDuplicate.add(element);  
    }  
    System.out.println("ArrayList sau khi xoá phần tử trùng: " + listNonDuplicate);  
}
```

Sử dụng HashSet

```
public static void main(String[] args) {  
    // Khởi tạo 1 danh sách ArrayList có 1 phần tử trùng lặp giá trị Java  
    List<String> listDuplicate = new ArrayList<String>();  
    listDuplicate.add("JAVA");  
    listDuplicate.add("PHP");  
    listDuplicate.add("JAVA");  
  
    // Khởi tạo 1 HashSet bằng listDuplicate  
    Set<String> set = new HashSet<String>(listDuplicate);  
  
    // Khởi tạo 1 ArrayList listNonDuplicate bằng set  
    List<String> listNonDuplicate = new ArrayList<String>(set);  
  
    System.out.println("ArrayList sau khi xoá phần tử trùng: " + listNonDuplicate);  
}
```

Sử dụng LinkedHashSet

```
public static void main(String[] args) {  
    // Khởi tạo 1 danh sách ArrayList có 1 phần tử trùng lặp giá trị Java  
    List<String> listDuplicate = new ArrayList<String>();  
    listDuplicate.add("JAVA");  
    listDuplicate.add("PHP");  
    listDuplicate.add("JAVA");  
  
    // Khởi tạo 1 LinkedHashSet bằng listDuplicate  
    Set<String> set = new LinkedHashSet<String>(listDuplicate);  
  
    // Khởi tạo 1 ArrayList listNonDuplicate bằng set  
    List<String> listNonDuplicate = new ArrayList<String>(set);  
  
    System.out.println("ArrayList sau khi xoá phần tử trùng: " + listNonDuplicate);  
}
```

Sử dụng LinkedHashSet

```
public static void main(String[] args) {  
    // Khởi tạo 1 danh sách ArrayList có 1 phần tử trùng lặp giá trị Java  
    List<String> listDuplicate = new ArrayList<String>();  
    listDuplicate.add("JAVA");  
    listDuplicate.add("PHP");  
    listDuplicate.add("JAVA");  
  
    // Khởi tạo 1 LinkedHashSet bằng listDuplicate  
    Set<String> set = new LinkedHashSet<String>(listDuplicate);  
  
    // Khởi tạo 1 ArrayList listNonDuplicate bằng set  
    List<String> listNonDuplicate = new ArrayList<String>(set);  
  
    System.out.println("ArrayList sau khi xoá phần tử trùng: " + listNonDuplicate);  
}
```

Tại sao **HashSet** và **LinkedHashSet**
lại được sử dụng để loại bỏ phần tử
trùng trong **ArrayList**?

Sử dụng phương thức distinct()

```
public static void main(String[] args) {  
    // Khởi tạo 1 danh sách ArrayList có 1 phần tử trùng lặp giá trị Java  
    List<String> listDuplicate = new ArrayList<String>();  
    listDuplicate.add("JAVA");  
    listDuplicate.add("PHP");  
    listDuplicate.add("JAVA");  
  
    // Sử dụng distinct() để loại bỏ phần tử trùng  
    List<String> listNonDuplicate = listDuplicate  
        .stream()  
        .distinct()  
        .toList();  
    System.out.println("ArrayList sau khi xoá phần tử trùng: " + listNonDuplicate);  
}
```

Sử dụng phương thức removeIf()

```
public static void main(String[] args) {  
    // Khởi tạo 1 danh sách ArrayList có 1 phần tử trùng lặp giá trị Java  
    List<String> listDuplicate = new ArrayList<String>();  
    listDuplicate.add("JAVA");  
    listDuplicate.add("PHP");  
    listDuplicate.add("JAVA");  
  
    // Khởi tạo 1 LinkedHashSet để loại bỏ phần tử trùng  
    Set<String> listNonDuplicate = new LinkedHashSet<>();  
    listDuplicate.removeIf(s -> !listNonDuplicate.add(s));  
  
    System.out.println("ArrayList sau khi xoá phần tử trùng: " + listNonDuplicate);  
}
```

Chuyển đổi giữa Array - ArrayList

To ArrayList: Sử dụng phương thức Arrays.asList()

```
public static void main(String[] args) {  
    // Khởi tạo và định nghĩa 1 Array  
    String[] arr = { "JAVA", "REACTJS", "JSP", "VUEJS" };  
  
    // Chuyển đổi Array thành ArrayList  
    ArrayList<String> list = new ArrayList<String>(Arrays.asList(arr));  
  
    // Thêm phần tử vào ArrayList sau khi chuyển đổi  
    list.add("C#");  
    list.add("PHP");  
  
    System.out.println("ArrayList sau khi chuyển đổi từ Array: " + list);  
}
```

To ArrayList: Sử dụng phương thức Collections.addAll()

```
public static void main(String[] args) {  
    // Khởi tạo và định nghĩa 1 Array  
    String[] arr = { "JAVA", "REACTJS", "JSP", "VUEJS" };  
  
    // Khởi tạo 1 ArrayList  
    ArrayList<String> list = new ArrayList<String>();  
  
    // Chuyển đổi Array thành ArrayList  
    Collections.addAll(list, arr);  
  
    // Thêm phần tử vào ArrayList sau khi chuyển đổi  
    list.add("C#");  
    list.add("PHP");  
  
    System.out.println("ArrayList sau khi chuyển đổi từ Array: " + list);  
}
```

To ArrayList: Sử dụng vòng lặp for

```
public static void main(String[] args) {  
    // Khởi tạo và định nghĩa 1 Array  
    String[] arr = { "JAVA", "REACTJS", "JSP", "VUEJS" };  
  
    // Khởi tạo 1 ArrayList  
    ArrayList<String> list = new ArrayList<String>();  
  
    // Chuyển đổi Array thành ArrayList  
    for (String item : arr) {  
        list.add(item);  
    }  
  
    // Thêm phần tử vào ArrayList sau khi chuyển đổi  
    list.add("C#");    list.add("PHP");  
    System.out.println("ArrayList sau khi chuyển đổi từ Array: " + list);  
}
```

To Array: Sử dụng phương thức toArray()

```
public static void main(String[] args) {  
    // Khởi tạo 1 ArrayList chứa các giá trị là số nguyên  
    List<Integer> list = new ArrayList<Integer>();  
    list.add(10);  
    list.add(20);  
    list.add(30);  
  
    // Chuyển đổi ArrayList sang Array  
    Integer[] arr = new Integer[list.size()];  
    arr = list.toArray(arr);  
  
    System.out.println(Arrays.toString(arr));  
}
```

To Array: Sử dụng vòng lặp for

```
public static void main(String[] args) {  
    // Khởi tạo 1 ArrayList chứa các giá trị là số nguyên  
    List<Integer> list = new ArrayList<Integer>();  
    list.add(10);  
    list.add(20);  
    list.add(30);  
  
    // Chuyển đổi ArrayList sang Array  
    Integer[] arr = new Integer[list.size()];  
    for (int i = 0; i < list.size(); i++) {  
        arr[i] = list.get(i);  
    }  
  
    System.out.println(Arrays.toString(arr));  
}
```

ROAD TO KOREA

Nếu có bất kỳ thắc mắc nào, hãy đặt câu hỏi qua

mail@mail.com hoặc Zalo 0xxx xxx xxx