**Collections trong java**

Collection trong java là một famework cung cấp các kiến trúc để lưu trữ và sử dụng một nhóm các đối tượng.

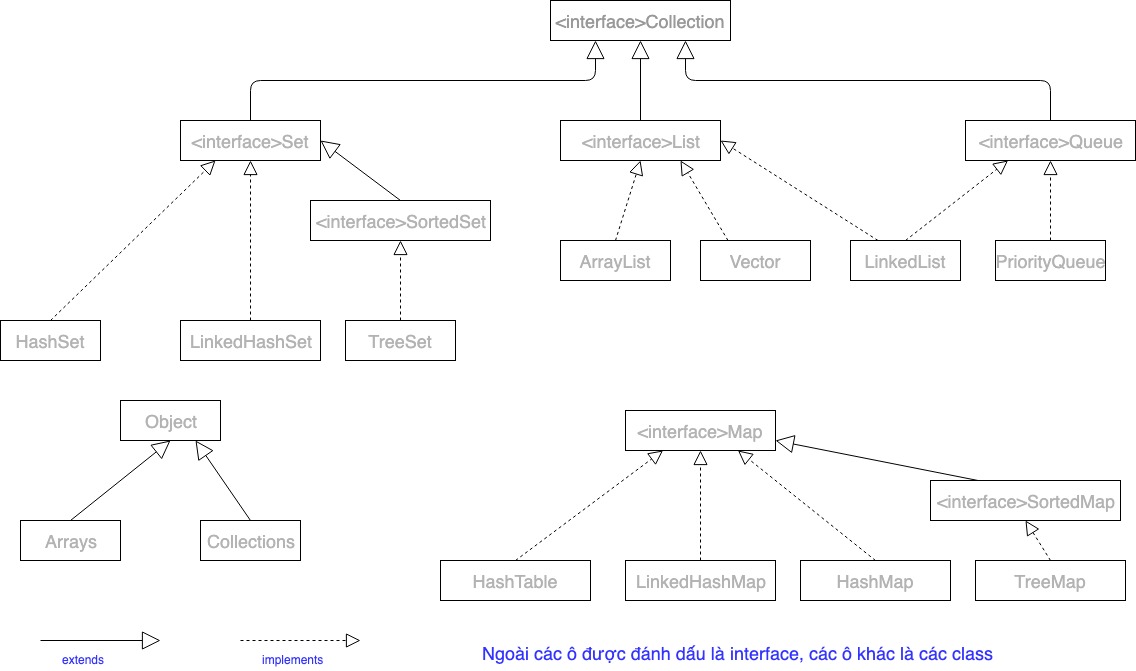
Trước khi *collection*được giới thiệu, Để nhóm các đối tượng trong java chúng ta có *Arrays*, *Vectors* hay là *HashTables.*

Với mỗi loại trên chúng ta sẽ có các cách thêm và truy cập khác nhau, chúng lại không các interface chung khiến cho các java dev lúc bấy giờ cảm thấy khó chịu.

Một hạn chế nữa các method trong ‘Vector’ là final, hiểu đơn giản là bạn sẽ không thể extend ‘Vector’ class để implement các collection tương tự.

Để giải quyết các rắc rối trên, các nhà phát triển java đã cho ra đời *Collection Framework*trong bản JDK 1.2.

**Hệ thống phân cấp của Collection Framework**

[](https://shareprogramming.net/wp-content/uploads/2019/08/Collection.jpg)

Nhìn sơ đồ trên có lẽ chúng ta sẽ thật sự choáng ngợp, thật ra mình cũng vậy. Thế nhưng các bạn đừng lo, hãy xem sơ đồ này như bức tranh toàn cảnh. Chúng ta sẽ đi tìm hiểu những collection hay dùng nhất, sau đó mới tính đến máy cái kia nha.

Giới thiệu sơ một xíu.

*Collection*là một interface với những method căn bản như: add(), remove(), contains(), isEmpty(), addAll(), … etc.

Collection interface **java.util.Collection** và Map interface **java.util.Map**là 2 thành phần chính khác nhau, =) cái này mình thấy phỏng vấn người ta hay hỏi.

Như vậy, với việc các class khác như HashSet, ArrayList etc, mặc dù chúng có các đặc thù riêng nhưng chắc rằng chúng sẽ có các method như trên bởi vì chúng đều implement từ collection interface.

**List:** là một danh sách các phần tử có thứ tự, List có thể chứa hai hoặc nhiều các phần tử giống nhau.

**Set**: Là một danh chứa các phần tử **không có thứ tự,** và không cho phép các phần tử trùng nhau.

**Map:**Lưu trữ các phần tử theo cặp giá trị (key, value), không cho phép có cặp key trùng nhau.

**Queue**: Lưu các phần tử theo công thứ FIFO (First In First Out).

List<String> courses = new ArrayList<>();

ArrayList<Integer> ids = new ArrayList<>();

Chúng ta có *ArrayList implements <interface> List.*thế nên hai cách khai báo ở trên là hoàn toàn hợp lệ. Nhưng mình khuyên nên dùng cách (1)

Vì nó thể hiện rõ phong cách hướng đối tượng của java. Chúng ta có thể linh động hơn trong việc tạo các collection khác nhau implement từ interface List.

ArrayList là một class implement từ interface List và dựa trên [Array data structure.](https://shareprogramming.net/array-data-structure/) ArrayList được sử dụng rất nhiều trong java bởi các chức năng và tính linh hoạt của nó.

ArrayList là một mảng động có thể thay đổi cách thích tuỳ thuộc vào số lượng phần tử được thêm vào hoặc xoá đi.

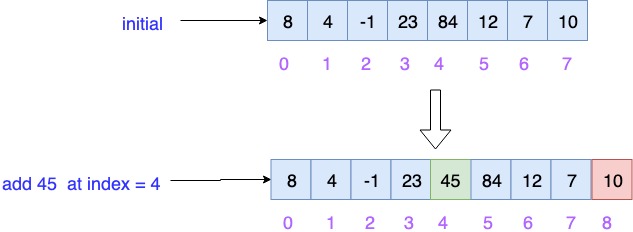
## Sử dụng Array hay ArrayList?

Array là một mảng các số lượng phần tử cố định, nghĩa rằng bạn sẽ  không thể thêm bất kỳ phần tử nào vào Array nếu nó đã chứa đầy phần tử. Mặc khác, nếu bạn xoá một phần tử trong Array vẫn sẽ không co lại và vẫn tốn một lượng memory như vậy.

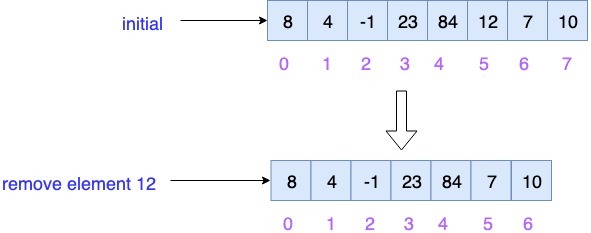
ArrayList là một mảng các các phần tử linh động, nó có thể tăng hoặc giảm kích thước khi bạn thêm hoặc xoá các phần tử trong mảng. Ngoài ra nó còn cung cấp các method giúp chúng ta dễ dàng thao tác với mảng phần tử hơn.

Các bạn hãy tưởng tượng nếu cần thêm hoặc xoá một phần tử tại vị trí index trong Array, thì chúng ta sẽ có nhiều việc để làm lắm đấy. Và tưởng tượng các bạn sẽ phải lặp đi lặp lại các bước như vậy khi cần thao tác với mảng thì thật là khó chịu đúng không nào?

### Minh hoạ thêm phần tử

[](https://shareprogramming.net/wp-content/uploads/2019/08/add-element-to-arraylist.jpg)

### Minh hoạ xoá phần tử

[[](https://shareprogramming.net/wp-content/uploads/2019/08/remove-element-arraylist.jpg)](https://shareprogramming.net/wp-content/uploads/2019/08/remove-element-arraylist.jpg)

## Khởi tạo một ArrayList

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Cách 1  ArrayList<Kiểu dữ liệu> <name> = new ArrayList<>();    // Cách 2  List<Kiểu dữ liệu> <name> = new ArrayList<>(); |

Kiểu dữ liệu dùng để xác định các thành phần có thể có trong ArrayList. Ví dụ

|  |  |
| --- | --- |
| 1 | List<String> strings = new ArrayList<>(); |

Chúng ta sẽ một ArrayList với các phần tử có kiểu String.

Note: Các kiểu [dữ liệu nguyên thuỷ](https://shareprogramming.net/kieu-du-lieu-nguyen-thuy-trong-java/) không được phép trong ArrayList.

|  |  |
| --- | --- |
| 1 | List<int> arrs = new ArrayList<>(); => Error |

## Thêm phần tử vào ArrayList

Để thêm một phần tử vào một ArrayList chúng ta sử dụng method add().

|  |  |
| --- | --- |
| 1  2  3  4  5 | // add the element at the end ArrrayList  strs.add("deftBlog");    // add the element at specified index  strs.add(5, "milo"); |

Ví dụ

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | import java.util.\*;    public class Main {        public static void main(String[] args) {          List<String> strs = new ArrayList<>();            strs.add("str 0");          strs.add("str 1");          strs.add("str 2");          strs.add("str 3");          System.out.println(strs);            strs.add(2, "deft");          System.out.println(strs);      }  } |

Output

[str 0, str 1, str 2, str 3]  
[str 0, str 1, deft, str 2, str 3]

## Xoá phần tử trong ArrayList

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | import java.util.\*;    public class Main {        public static void main(String[] args) {          List<String> strs = new ArrayList<>();            strs.add("str 0");          strs.add("str 1");          strs.add("str 2");          strs.add("str 3");          System.out.println(strs);            // remove by value          strs.remove("str 0");          System.out.println(strs);            // remove by index          strs.remove(2);          System.out.println(strs);      }  } |

output

[str 0, str 1, str 2, str 3]  
[str 1, str 2, str 3]  
[str 1, str 2]

## Các method trong ArrayList

Chúng ta đã tìm hiểu sơ cách ArrayList hoạt động. Ngoài các method cho phép thêm và xoá phần tử thì ArrayList còn cung cấp những method khác giúp cho việc lập trình chúng ta trở nên dễ dàng hơn. Hãy Xem qua và nhớ những method này giải giải quyết cho chúng ta vấn đề gì, chứ không cần học thuộc lòng vì IDE hiện giờ đều có các chức năng gợi ý cho bạn.

1, [boolean add(Object o)](https://shareprogramming.net/arraylist-add-trong-java-voi-vi-du-cu-the/)**:** Thêm một phần tử vào ArrayList

|  |  |
| --- | --- |
| 1 | obj.add("deft"); |

2,[boolean add(int index, Object o)](https://shareprogramming.net/arraylist-add-trong-java-voi-vi-du-cu-the/)**:** Thêm một phần tử vào vị trí index. Nếu index không hợp lệ chúng ta sẽ nhận lại một IndexOutOfBoundsException.

|  |  |
| --- | --- |
| 1 | obj.add(2, "deft"); |

3, [boolean remove(Object o)](https://shareprogramming.net/arraylist-remove-trong-java-voi-vi-du-cu-the/)**:** Xoá object o khỏi ArrayList, object o này phải chứa trong ArrayList.

|  |  |
| --- | --- |
| 1 | obj.remove("deft"); |

4, [boolean remove(int index):](https://shareprogramming.net/arraylist-remove-trong-java-voi-vi-du-cu-the/) Xoá một phần tử tại vị trí index, nếu index không hợp lệ chúng ta sẽ nhận IndexOutOfBoundsException.

|  |  |
| --- | --- |
| 1 | obj.remove(3); |

5, [Object set(int index, Object o)](https://shareprogramming.net/replace-trong-arraylist-voi-vi-du-cu-the/)**:** Cập nhật phần tử tại vị trí index, nếu index không hợp lệ chúng ta sẽ nhận IndexOutOfBoundsException.

|  |  |
| --- | --- |
| 1 | obj.set(3, "update"); |

6, [int indexOf(Object o)](https://shareprogramming.net/arraylist-indexof-lastindexof/): Lấy vị trí index của object o trong ArrayList, nếu object o không chứa trong ArrayList nhận kết -1.

|  |  |
| --- | --- |
| 1 | int pos = obj.indexOf("deft); |

7, [Object get(int index)](https://shareprogramming.net/arraylist-get/)**:** Return object tại vị trí index trong ArrayList.

|  |  |
| --- | --- |
| 1 | String str = obj.get(3); |

8,[int size()](https://shareprogramming.net/arraylist-size-in-java/): lấy số lượng phần tử chứa trong ArrayList

|  |  |
| --- | --- |
| 1 | int size = obj.size(); |

9, [boolean contains(Object o)](https://shareprogramming.net/java-arraylist-contains-method-voi-vi-du-cu-the/): Kiểm tra phần tử object o có chứa trong ArrayList, nếu có return true, ngược lại false.

|  |  |
| --- | --- |
| 1 | boolean isExist = obj.contains("deft"); |

10, [void clear()](https://shareprogramming.net/clear-arraylist-trong-java-voi-vi-du-cu-the/)**:** Xoá tất cả các phần tử trong ArrayList

|  |  |
| --- | --- |
| 1 | obj.clear(); |

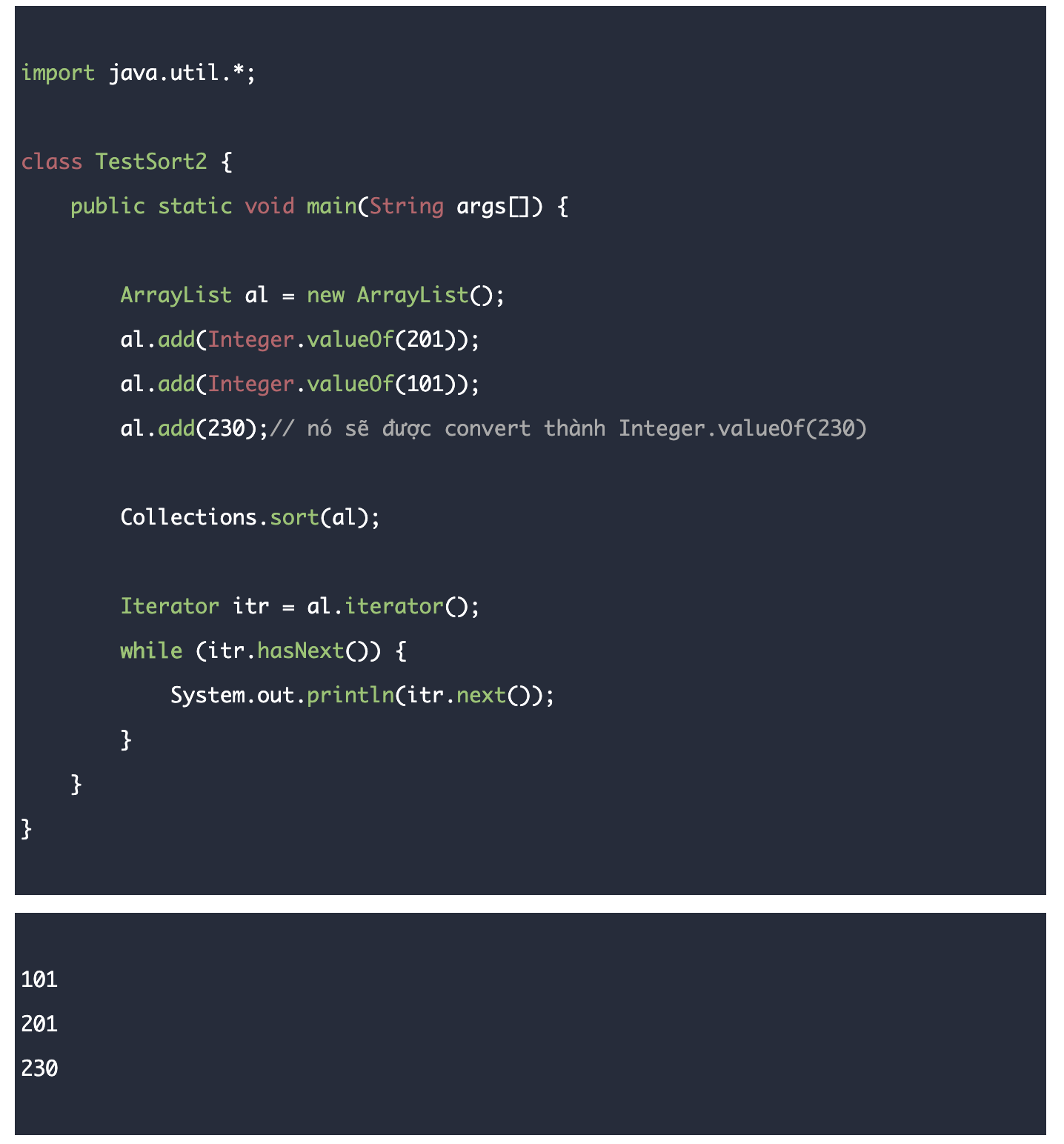
11, int lastIndexOf(): Lấy vị trí xuất hiện cuối cùng của phần tử.

1. String objects
2. Wrapper class objects
3. Người dùng tự định nghĩa(User-defined) class objects

## ****Ví dụ về sắp xếp List chứa các đối tượng String****



## ****Ví dụ về sắp xếp List chứa các đối tượng Wrapper****



## ****Java Comparable interface****

Java Comparable interface được sử dụng để chỉ ra thứ tự của các đối tượng Người dùng tự định nghĩa(User-defined).

Interface này chỉ chứa duy nhất 1 phương thức tên là compareTo(Object).

Nó cung cấp duy nhất 1 trình tự sắp xếp ví dụ như bạn chỉ có thể sắp các phần tử của đối tượng Nhân viên theo ‘mã số’ hoặc ‘tên’ hoặc tuổi’,…





## ****Java Comparator interface****

Java Comparator interface được sử dụng để chỉ ra thứ tự của các đối tượng Người dùng tự định nghĩa(User-defined). Nó định nghĩa 2 phương thức compare(Object obj1,Object obj2) và equals(Object element).

Nó cung cấp nhiều trình tự sắp xếp ví dụ như bạn có thể sắp các phần tử của đối tượng Nhân viên theo ‘mã số’, ‘tên’, tuổi’,…







