* In thông thường: **println();**
* For thông thường:

**For (int i = 0; i <** *tên\_List***.size(); i++)**

**Println(***tên\_List***.get(i));**

* For cải tiến:

**For(int** value **:** *tên\_List*)

**Println(**value**);**

* Sử dụng Iterator:

-Khai báo:

**Iterator <datatype>** *tên\_Iterator* = *tên\_List***.iterator();**

**While(***tên\_Iterator***.hasNext())**

**Println(***ten\_Iterator***.Next());**

* Sử dụng ListIterator (tương tự **Iterator)**
* Kiểm tra đã tồn tại hay chưa

tên\_Collection**.contains**

List Interface

Các phần tử được sắp xếp có thứ tự và giá trị của các phần tử này có thể trùng nhau.

*Class phổ biến*: **ArrayList**, **LinkedList**

*Cú pháp:*

**List**<datatype> tên\_List **= new ArrayList**<datatype>();

**List**<datatype> tên\_List = **new LinkedList**<datatype>();

*Hiển thị:*

-For thông thường

-For cải tiến

-Sử dụng Iterator

-Sử dụng ListIterator

*Thêm phần tử*

nameList.**add();**

nameList.**addAll();**  // Thêm toàn bộ phần tử vào cuối list (cùng datatype)

nameList.**addAll**(vị trí cần thêm, nameList) // thêm vào vị trí bất kì

*Truy cập phần tử*

**get();** // truy cập vào phần tử thông qua chỉ số index

*Cập nhật giá trị phần tử*

**set(index, element)**

Index: chỉ số phần tử cần cập nhật

Element: phần tử mới để thay thế

*Xóa phần tử*

**nameList.remove(index)** // xóa dựa vào chỉ số phần tử

**nameList.remove(“ “)**  // xóa trực tiếp phần tử

*Tìm kiếm phần tử*

Tìm trực tiếp

**bool contains(phần\_tử\_cần\_tìm);**

if (nameString.**contains("phần\_tử\_cần\_tìm")**) {

        System.out.println("Có”) }

else {

        System.out.println("Không");

    }

*Tìm kiếm vị trí xuất hiện đầu tiên*

**int indexOf(phần\_tử\_cần\_tìm);**

**int** firstIndex = nameList.**indexOf("phần\_tử\_cần\_tìm");**

*Tìm kiếm vị trí xuất hiện cuối cùng*

**int lastIndexOf(phần\_tử\_cần\_tìm);**

**int** lastIndex = nameList.**lastIndexOf("phần\_tử\_cần\_tìm ");**

*Sắp xếp các phần tử*

**Collections.sort(tên\_list);**

*Sao chép*

**Collections.copy(danh\_sách\_đích, danh\_sách\_nguồn);**

Set interface

Các phần tử là duy nhất (không được giống nhau)

Trường hợp sử dụng: khi chúng ta muốn lưu trữ một dánh sách các phần tử không có sự trùng lập hoặc khi chúng ta không quan tâm đến thứ tự của các phần tử trong danh sách đó.

*Class phổ biến*:

**HashSet:** các phần tử không được sắp xếp theo bất kỳ thứ tự nào.

**TreeSet:** thứ tự các phần tử được sắp xếp tăng dần

*Cú pháp:*

**Set**<datatype> tên\_Set **= new HashSet**<datatype>();

**Set**<datatype> tên\_Set = **new TreeSet**<datatype>();

*Hiển thị:*

-For cải tiến

-Sử dụng Iterator

-Sử dụng ListIterator

*Thêm phần tử*

tên\_Set.**add();**

tên\_Set.**addAll();**  // Thêm toàn bộ phần tử vào cuối list (cùng datatype)

tên\_Set.**addAll**(vị trí cần thêm, nameList) // thêm vào vị trí bất kì

*Truy cập phần tử*

**get();** // truy cập vào phần tử thông qua chỉ số index

*Cập nhật giá trị phần tử*

**set(index, element)**

Index: chỉ số phần tử cần cập nhật

Element: phần tử mới để thay thế

*Xóa phần tử*

tên\_Set**.remove(index)** // xóa dựa vào chỉ số phần tử

tên\_Set**.remove(“ “)**  // xóa trực tiếp phần tử

*Tìm kiếm phần tử*

Tìm trực tiếp

**bool contains(phần\_tử\_cần\_tìm);**

if (nameString.**contains("phần\_tử\_cần\_tìm")**) {

        System.out.println("Có”) }

else {

        System.out.println("Không");

    }

*Tìm kiếm vị trí xuất hiện đầu tiên*

**int indexOf(phần\_tử\_cần\_tìm);**

**int** firstIndex = nameList.**indexOf("phần\_tử\_cần\_tìm");**

*Tìm kiếm vị trí xuất hiện cuối cùng*

**int lastIndexOf(phần\_tử\_cần\_tìm);**

**int** lastIndex = tên\_Set.**lastIndexOf("phần\_tử\_cần\_tìm ");**

*Sắp xếp các phần tử*

**Collections.sort(tên\_list);**

*Sao chép*

**Collections.copy(danh\_sách\_đích, danh\_sách\_nguồn);**