Java

# ArrayList

## Khởi tạo ArrayList trong Java

ArrayList list = **new** ArrayList(); // non-generic - kiểu cũ

ArrayList<String> list = **new** ArrayList<String>(); // generic - kiểu mới

VD :

// import gói thư viện java.util.ArrayList

**import** java.util.ArrayList;

**public** **class** KhoiTaoArrayList {

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

        // khai báo 1 ArrayList có tên là listString

        // có kiểu là String

        ArrayList<String> listString = **new** ArrayList<String>();

    }

}

## Constructor của lớp ArrayList

|  |  |
| --- | --- |
| **Constructor** | **Mô tả** |
| ArrayList() | Nó được sử dụng để khởi tạo một danh sách mảng trống. |
| ArrayList(Collection c) | Nó được sử dụng để xây dựng một danh sách mảng được khởi tạo với các phần tử của collection c. |
| ArrayList(int capacity) | Nó được sử dụng để xây dựng một danh sách mảng mà có dung lượng ban đầu được chỉ định. |

## 1.3 **Phương thức của lớp 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. |

## 1.4 Hiển thị các phần tử có trong ArrayList

**import** java.util.ArrayList;

**public** **class** DuyetArrayList1 {

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

        // khai báo 1 ArrayList có tên là list

        // có kiểu là String

        ArrayList<String> list = **new** ArrayList<String>();

        // thêm các phần tử vào list

        list.add("Java");

        list.add("C++");

        list.add("PHP");

        list.add("Java");

        // hiển thị các phần tử của list

        System.out.println("Các phần tử có trong list là: ");

        System.out.println(list);

    }

}

## 1.5 **Duyệt các phần tử của ArrayList - sử dụng vòng lặp for**

**import** java.util.ArrayList;

**public** **class** DuyetArrayList2 {

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

        // khai báo 1 ArrayList có tên là list

        // có kiểu là String

        ArrayList<String> list = **new** ArrayList<String>();

        // thêm các phần tử vào list

        list.add("Java");

        list.add("C++");

        list.add("PHP");

        list.add("Java");

        // sử dụng vòng lặp for - hiển thị các phần tử của list

        System.out.println("Các phần tử có trong list là: ");

**for** (**int** i = 0; i < list.size(); i++) {

            System.out.println(list.get(i));

        }

    }

}

### 1.5.1 **Duyệt các phần tử của ArrayList - sử dụng vòng lặp for cải tiến**

**import** java.util.ArrayList;

**public** **class** DuyetArrayList3 {

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

        // khai báo 1 ArrayList có tên là list

        // có kiểu là String

        ArrayList<String> list = **new** ArrayList<String>();

        // thêm các phần tử vào list

        list.add("Java");

        list.add("C++");

        list.add("PHP");

        list.add("Java");

        // sử dụng vòng lặp for cải tiến - hiển thị các phần tử của list

        System.out.println("Các phần tử có trong list là: ");

**for** (String str : list) {

            System.out.println(str);

        }

    }

}

### 1.5.2 Duyệt các phần tử của ArrayList - sử dụng Iterator.

**public** **class** DuyetArrayList4 {

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

        // khai báo 1 ArrayList có tên là list

        // có kiểu là String

        ArrayList<String> list = **new** ArrayList<String>();

        // thêm các phần tử vào list

        list.add("Java");

        list.add("C++");

        list.add("PHP");

        list.add("Java");

        // sử dụng Iterator - hiển thị các phần tử của list

        Iterator<String> iterator = list.iterator();

        System.out.println("Các phần tử có trong list là: ");

**while** (iterator.hasNext()) {

            System.out.println((String) iterator.next());

        }

    }

}

### 1.5.3 **Duyệt các phần tử của ArrayList - sử dụng ListIterator.**

**import** java.util.ArrayList;

**import** java.util.ListIterator;

**public** **class** DuyetArrayList5 {

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

        // khai báo 1 ArrayList có tên là list

        // có kiểu là String

        ArrayList<String> list = **new** ArrayList<String>();

        // thêm các phần tử vào list

        list.add("Java");

        list.add("C++");

        list.add("PHP");

        list.add("Java");

        // sử dụng ListIterator - hiển thị các phần tử của list

        ListIterator<String> iterator = list.listIterator();

        System.out.println("Các phần tử có trong list là: ");

**while** (iterator.hasNext()) {

            System.out.println((String) iterator.next());

        }

    }

}

## 1.6 Sử dụng addAll(), removeAll(), retainAll() của lớp ArrayList

**import** java.util.ArrayList;

**public** **class** PhuongThucArrayList1 {

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

        // khai báo 1 ArrayList có tên là list

        // có kiểu là String

        ArrayList<String> list = **new** ArrayList<String>();

        // Add objects to list

        list.add("Java");

        list.add("C++");

        list.add("PHP");

        list.add("Java");

        System.out.println("ví dụ sử dụng phương thức addAll()");

        System.out.println("-----------------------------------");

        // thêm các phần tử của list vào listA

        ArrayList<String> listA = **new** ArrayList<String>();

        listA.addAll(list);

        System.out.print("listA:");

        showList(listA);

        System.out.println("\nví dụ sử dụng phương thức retainAll()");

        System.out.println("-----------------------------------");

        // khởi tạo listB

        ArrayList<String> listB = **new** ArrayList<String>();

        listB.add("Java");

        // xóa những phần tử không thuộc listB khỏi listA

        listA.retainAll(listB);

        System.out.print("listA:");

        showList(listA);

        System.out.println("\nví dụ sử dụng phương thức removeAll()");

        System.out.println("-----------------------------------");

        // xóa những phần tử thuộc listB  khỏi list

        list.removeAll(listB);

        System.out.print("list:");

        showList(list);

    }

**public** **static** **void** showList(ArrayList<String> list) {

        // Show list through for-each

**for** (String obj : list) {

            System.out.print("\t" + obj + ", ");

        }

        System.out.println();

    }

}

## 1.7 Truy cập 1 phần tử bất kì trong ArrayList

Java cung cấp cho chúng ta phương thức get() để truy cập đến 1 phần tử bất kỳ trong ArrayList thông qua chỉ số của phần tử đó. Chỉ số của ArrayList trong Java bắt đầu từ 0.

[?](https://viettuts.vn/java-collection/arraylist-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | **package** vn.viettuts.arraylist;    **import** java.util.ArrayList;    **public** **class** TruyCapArrayList1 {  **public** **static** **void** main(String[] args) {          // khai báo 1 ArrayList có tên là list          // có kiểu là String          ArrayList<String> list = **new** ArrayList<String>();          // thêm các phần tử vào list          list.add("Java");          list.add("C++");          list.add("PHP");          list.add("Java");            // truy cập phần tử có chỉ số 3 của list          System.out.println(list.get(3));      }  } |

Kết quả:

Java

## 1.8 Cập nhật giá trị của phần tử ArrayList

Để cập nhật giá trị của phần tử trong ArrayList, Java cung cấp cho chúng ta phương thức set(index, element), trong đó index là chỉ số của phần tử cần cập nhật và element là phần tử mới để thay thế.

[?](https://viettuts.vn/java-collection/arraylist-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | **package** vn.viettuts.arraylist;    **import** java.util.ArrayList;    **public** **class** CapNhatArrayList1 {  **public** **static** **void** main(String[] args) {          // khai báo 1 ArrayList có tên là list          // có kiểu là String          ArrayList<String> list = **new** ArrayList<String>();          // thêm các phần tử vào list          list.add("Java");          list.add("C++");          list.add("PHP");          list.add("Java");            System.out.println("list trước khi cập nhật: ");          System.out.println(list);          // cập nhật giá trị cho phần tử có chỉ số là 3 (Java)          list.set(3, "Python");          System.out.println("list trước khi cập nhật: ");          System.out.println(list);      }  } |

Kết quả:

list trước khi cập nhật:

[Java, C++, PHP, Java]

list trước khi cập nhật:

[Java, C++, PHP, Python]

## 1.9 Xóa 1 phần tử trong ArrayList

### 1.9.1 Xóa bằng clear()

Phương thức clear() sẽ xóa tất cả các phần tử có trong ArrayList. Sau đây là ví dụ minh họa phương thức này.

[?](https://viettuts.vn/java-collection/arraylist-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | **package** vn.viettuts.arraylist;    **import** java.util.ArrayList;    **public** **class** XoaArrayList1 {  **public** **static** **void** main(String[] args) {          // khai báo 1 ArrayList có tên là list          // có kiểu là String          ArrayList<String> list = **new** ArrayList<String>();          // thêm các phần tử vào list          list.add("Java");          list.add("C++");          list.add("PHP");          list.add("Python");            System.out.println("Số phần tử của list ban đầu : " + list);          System.out.println("Các phần tử của list ban đầu: " + list.size());          // clear list          list.clear();          System.out.println("\nSố phần tử của list sau khi clear: " + list);          System.out.println("Các phần tử của list sau khi clear: " + list.size());      }  } |

Kết quả:

Số phần tử của list ban đầu : [Java, C++, PHP, Python]

Các phần tử của list ban đầu: 4

Số phần tử của list sau khi clear: []

Các phần tử của list sau khi clear: 0

### 1.9.2 Xóa bằng remove()

**Phương thức remove()**

Phương thức remove() sẽ xóa phần tử ra khỏi ArrayList theo 2 cách đó là xóa dựa vào chỉ số của phần tử và xóa trực tiếp phần tử đó (không cần biết đến chỉ số của nó). Ví dụ dưới đây sẽ minh họa 2 cách xóa này:

[?](https://viettuts.vn/java-collection/arraylist-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | **package** vn.viettuts.arraylist;    **import** java.util.ArrayList;    **public** **class** XoaArrayList1 {  **public** **static** **void** main(String[] args) {          // khai báo 1 ArrayList có tên là list          // có kiểu là String          ArrayList<String> list = **new** ArrayList<String>();          // thêm các phần tử vào list          list.add("Java");          list.add("C++");          list.add("PHP");          list.add("Python");            System.out.println("Số phần tử của list ban đầu : " + list);          System.out.println("Các phần tử của list ban đầu: " + list.size());          // remove phần tử có chỉ số index = 1 khỏi list          list.remove(1);          System.out.println("\nSố phần tử của list sau khi remove phan tu co index = 1: "                  + list);          System.out.println("Các phần tử của list sau khi remove phan tu co index = 1: "                  + list.size());          // remove phần tử có chỉ số index = 1 khỏi list          list.remove("PHP");          System.out.println("\nSố phần tử của list sau khi remove phan tu \"PHP\": "                  + list);          System.out.println("Các phần tử của list sau khi remove phan tu \"PHP\": "                  + list.size());      }  } |

Kết quả:

Số phần tử của list ban đầu : [Java, C++, PHP, Python]

Các phần tử của list ban đầu: 4

Số phần tử của list sau khi remove phan tu co index = 1: [Java, PHP, Python]

Các phần tử của list sau khi remove phan tu co index = 1: 3

Số phần tử của list sau khi remove phan tu "PHP": [Java, Python]

Các phần tử của list sau khi remove phan tu "PHP": 2

## 1.10 Tìm kiếm 1 phần tử trong ArrayList

### 1.10.1 **Tìm kiếm trực tiếp phần tử bằng Contains()**

|  |
| --- |
| **import** java.util.ArrayList;    **public** **class** TimKiemArrayList1 {  **public** **static** **void** main(String[] args) {          // khai báo 1 ArrayList có tên là list          // có kiểu là String          ArrayList<String> list = **new** ArrayList<String>();          // thêm các phần tử vào list          list.add("Java");          list.add("C++");          list.add("PHP");          list.add("Python");            // kiểm tra xem PHP có tồn tại trong list hay không?          System.out.println(list.contains("PHP"));          // kiểm tra xem ANDROID có tồn tại trong list hay không?          System.out.println(list.contains("ANDROID"));      }  } |

Kết quả:

true

false

### 1.10.2 **Tìm kiếm vị trí xuất hiện đầu tiên của 1 phần tử trong ArrayList.**

Để tìm kiếm vị trí xuất hiện đầu tiên của 1 phần tử trong ArrayList, chúng ta sẽ sừ dụng phương thức indexOf(). Kết quả của phương thức này sẽ trả về chỉ số xuất hiện đầu tiên của phần tử đó trong ArrayList, ngược lại nếu không tìm thấy trả về -1.

[?](https://viettuts.vn/java-collection/arraylist-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | **package** vn.viettuts.arraylist;    **import** java.util.ArrayList;    **public** **class** TimKiemArrayList2 {  **public** **static** **void** main(String[] args) {          // khai báo 1 ArrayList có tên là list          // có kiểu là String          ArrayList<String> list = **new** ArrayList<String>();          // thêm các phần tử vào list          list.add("Java");          list.add("C++");          list.add("PHP");          list.add("Python");            // kiểm tra xem Java có tồn tại trong list hay không?          System.out.println(list.indexOf("Java"));          // kiểm tra xem ANDROID có tồn tại trong list hay không?          System.out.println(list.indexOf("ANDROID"));      }  } |

Kết quả:

0

-1

### 1.10.3 Tìm kiếm vị trí xuất hiện cuối cùng của 1 phần tử trong List.

Để tìm kiếm vị trí xuất hiện cuối cùng của 1 phần tử trong ArrayList, chúng ta sẽ sừ dụng phương thức lastIndexOf(). Kết quả của phương thức này sẽ trả về chỉ số xuất hiện cuối cùng của phần tử đó trong ArrayList, ngược lại nếu không tìm thấy trả về -1.

[?](https://viettuts.vn/java-collection/arraylist-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | **package** vn.viettuts.arraylist;    **import** java.util.ArrayList;    **public** **class** TimKiemArrayList3 {  **public** **static** **void** main(String[] args) {          // khai báo 1 ArrayList có tên là list          // có kiểu là String          ArrayList<String> list = **new** ArrayList<String>();          // thêm các phần tử vào list          list.add("Java");          list.add("C++");          list.add("PHP");          list.add("Java");            // kiểm tra xem Java có tồn tại trong list hay không?          System.out.println(list.lastIndexOf("Java"));          // kiểm tra xem ANDROID có tồn tại trong list hay không?          System.out.println(list.lastIndexOf("ANDROID"));      }  } |

Kết quả:

3

-1

### 1.11 **Chuyển ArrayList sang mảng (Array) trong Java**

|  |
| --- |
| **import** java.util.ArrayList;    **public** **class** ConvertToArray {  **public** **static** **void** main(String[] args) {          // khai báo 1 ArrayList có tên là list          // có kiểu là String          ArrayList<String> list = **new** ArrayList<String>();          // thêm các phần tử vào list          list.add("Java");          list.add("C++");          list.add("PHP");          list.add("Java");            // sử dụng phương thức toArray() chuyển list thành mảng          // kết quả của phương thức này sẽ trả về mảng arr          Object[] arr = list.toArray();            // hiển thị các phần tử có trong mảng arr  **for** (**int** i = 0; i < arr.length; i++) {              System.out.println("Phần tử tại vị trí " + i + " "                      + "trong arr là " + arr[i]);          }      }  } |

Kết quả:

Phần tử tại vị trí 0 trong arr là Java

Phần tử tại vị trí 1 trong arr là C++

Phần tử tại vị trí 2 trong arr là PHP

Phần tử tại vị trí 3 trong arr là Java

## 1.12 Show list

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** ArrayListExample1 {

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

        // Creating arraylist

        ArrayList<String> list = **new** ArrayList<String>();

        // Add objects to arraylist

        list.add("Java");

        list.add("C++");

        list.add("PHP");

        list.add("Java");

        // Show list through Iterator

        Iterator<String> itr = list.iterator();

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

            System.out.print(itr.next() + ", ");

        }

        // Show list through for-each

        System.out.println();

**for** (String obj : list) {

            System.out.print(obj + ", ");

        }

        // Show list through index

        System.out.println();

**int** size = list.size();

**for** (**int** i = 0; i < size; i++) {

            System.out.print(list.get(i) + ", ");

        }

    }

}

## 1.13 VD hoàn chỉnh toString() , create obj , showlist

|  |
| --- |
| **import** java.util.ArrayList;    **class** Student {  **private** String name;  **private** **int** age;  **public** Student(String name, **int** age) {  **super**();  **this**.name = name;  **this**.age = age;      }  **public** String getName() {  **return** name;      }  **public** **void** setName(String name) {  **this**.name = name;      }  **public** **int** getAge() {  **return** age;      }  **public** **void** setAge(**int** age) {  **this**.age = age;      }      @Override  **public** String toString() {  **return** "Student@[name=" + name + ", age=" + age + "]";      }  }    **public** **class** ArrayListExample2 {  **public** **static** **void** main(String[] args) {          // Create listStudent          ArrayList<Student> listStudent = **new** ArrayList<Student>();          // Create students          Student student1 = **new** Student("Bac", 17);          Student student2 = **new** Student("Nam", 20);          Student student3 = **new** Student("Trung", 19);          // Add objects to listStudent          listStudent.add(student1);          listStudent.add(student2);          listStudent.add(student3);          // Show listStudent  **for** (Student student : listStudent) {              System.out.println(student.toString());          }      }  } |

Output:

Student@[name=Bac, age=17]

Student@[name=Nam, age=20]

Student@[name=Trung, age=19]

# Collections

## Thêm tất cả các phần tử quy định vào bộ sưu tập đc chỉ định

### Collections.addAll (Collection<? superT>c, T ... elements)

|  |
| --- |
| 1. **import** java.util.ArrayList; 2. **import** java.util.Collections; 3. **import** java.util.List; 5. **public** **class** CollectionsExample1 { 6. **public** **static** **void** main(String a[]) { 7. // create list 8. List<String> list = **new** ArrayList<String>(); 9. // add elements to list 10. list.add("C++"); 11. list.add("Java"); 12. list.add("PHP"); 13. // print list 14. System.out.println(list); 16. // add elements to list 17. Collections.addAll(list, "Servlet", "JSP"); 18. // print list 19. System.out.println(list); 21. // create array of String 22. String[] strArr = { "C#", ".Net" }; 23. // add elements to list 24. Collections.addAll(list, strArr); 25. // print list 26. System.out.println(list); 27. } 28. } |

Kết quả:

[C++, Java, PHP]

[C++, Java, PHP, Servlet, JSP]

[C++, Java, PHP, Servlet, JSP, C#, .Net]

## 2.2 Collections.max

=> Collections.max() được sử dụng để trả về phần tử max của bộ sưu tập đã cho, theo thứ tự được sắp xếp bởi comparator được chỉ định.

|  |
| --- |
| **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.List;    **public** **class** CollectionsExample2 {  **public** **static** **void** main(String a[]) {          List<Integer> list = **new** ArrayList<Integer>();          list.add(46);          list.add(67);          list.add(24);          list.add(16);          list.add(8);          list.add(12);          list.add(67);          System.out.println("Max = " + Collections.max(list));      }  } |

Kết quả:

Max = 67

## 2.3 Collections.min

=> Collections.min() được sử dụng để trả lại phần tử min của bộ sưu tập đã cho, theo thứ tự được sắp xếp bởi comparable được chỉ định.

|  |
| --- |
| **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.List;    **public** **class** CollectionsExample3 {  **public** **static** **void** main(String a[]) {          List<Integer> list = **new** ArrayList<Integer>();          list.add(46);          list.add(67);          list.add(24);          list.add(16);          list.add(8);          list.add(12);          list.add(67);          System.out.println("Min = " + Collections.min(list));      }  } |

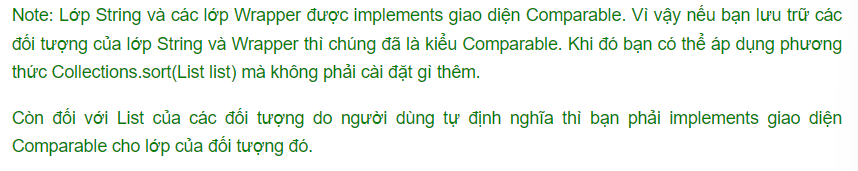
Kết quả:

Min = 8

## 2.4 Các phương thức của lớp Collections trg Java

|  |  |
| --- | --- |
| **Method** | **Description** |
| static <T> boolean addAll(Collection<? super T> c, T... elements) | Nó được sử dụng để thêm tất cả các phần tử quy định vào bộ sưu tập được chỉ định. |
| static <T> Queue<T> asLifoQueue(Deque<T> deque) | Nó được sử dụng để trả về một đối tượng Deque như một hàng đợi LIFO (Last-In-First-Out). |
| static <T> int binarySearch(List<? extends T> list, T key, Comparator<? super T< c) | Nó được sử dụng để tìm kiếm danh sách được chỉ định cho đối tượng được chỉ định sử dụng thuật toán tìm kiếm nhị phân. |
| static <E> List<E> checkedList(List<E> list, Class<E> type) | Nó được sử dụng để trả về một chế độ xem kiểu động của đối tượng list được chỉ định. |
| static <E> Set<E> checkedSet(Set<E> s, Class<E> type) | Nó được sử dụng để trả về một chế độ xem kiểu động của đối tượng set được chỉ định. |
| static <E> SortedSet<E>checkedSortedSet(SortedSet<E> s, Class<E> type) | Nó được sử dụng để trả về một chế độ xem kiểu động của đối tượng SortedSet được chỉ định. |
| static void reverse(List<?> list) | Nó được sử dụng để đảo ngược thứ tự của các phần tử trong danh sách được chỉ định.. |
| static <T> T max(Collection<? extends T> coll, Comparator<? super T> comp) | Nó được sử dụng để trả về phần tử max của bộ sưu tập đã cho, theo thứ tự được sắp xếp bởi comparator được chỉ định. |
| static <T extends Object & Comparable<? super T>>T min(Collection<? extends T> coll) | Nó được sử dụng để trả về phần tử min của bộ sưu tập đã cho, theo thứ tự được sắp xếp bởi comparable được chỉ định. |
| static boolean replaceAll(List list, T oldVal, T newVal) | Nó được sử dụng để thay thế tất cả các lần xuất hiện của một giá trị được chỉ định trong danh sách với một giá trị khác. |

## 2.5 Sorting trong Collections



### 2.5.1 Sắp xếp vs String

#### 2.5.1.1 Collections.sort(list) sắp xếp tăng dần

|  |
| --- |
| **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.List;    **public** **class** CollectionsExample3 {  **public** **static** **void** main(String args[]) {          // create list          List<String> list = **new** ArrayList<String>();          // add elements to list          list.add("Java");          list.add("C");          list.add("C++");          list.add("PHP");          list.add("Python");          // sort list          Collections.sort(list);          // show list  **for** (String element : list) {              System.out.println(element);          }      }  } |

Kết quả:

C

C++

Java

PHP

Python

#### 2.5.1.2 Collections.sort(list) sắp xếp giảm dần

|  |
| --- |
| **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.Comparator;  **import** java.util.List;    /\*\*   \* CollectionsExample4 class   \*   \* @author viettuts.vn   \*/  **public** **class** CollectionsExample4 {  **public** **static** **void** main(String args[]) {          // create list          List<String> list = **new** ArrayList<String>();          // add elements to list          list.add("Java");          list.add("C");          list.add("C++");          list.add("PHP");          list.add("Python");          // sort list          Collections.sort(list, **new** Comparator<String>() {              @Override  **public** **int** compare(String o1, String o2) {  **return** o2.compareTo(o1);              }          });          // show list  **for** (String element : list) {              System.out.println(element);          }      }  } |

Kết quả:

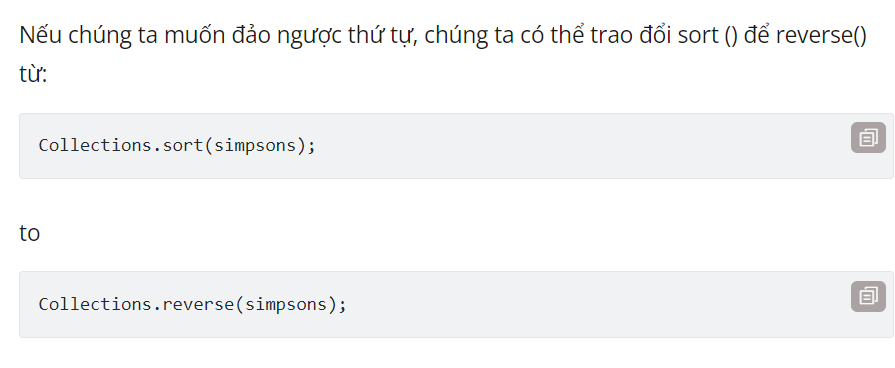
Python

PHP

Java

C++

C



### 2.5.2 Sắp xếp vs lớp Wrapper (Double , vv)

|  |
| --- |
| **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.Comparator;  **import** java.util.List;    /\*\*   \* CollectionsExample5 class   \*   \* @author viettuts.vn   \*/  **public** **class** CollectionsExample5 {  **public** **static** **void** main(String[] args) {          // create list          List<Double> list = **new** ArrayList<Double>();          // add elements to list          list.add(15D);          list.add(20D);          list.add(16.5D);            // sort list ASC          Collections.sort(list);          // show list          System.out.println("show list ASC:");  **for** (Double element : list) {              System.out.println(element);          }            // sort list DESC          Collections.sort(list, **new** Comparator<Double>() {              @Override  **public** **int** compare(Double o1, Double o2) {  **return** o1 > o2 ? -1 : 1;              }          });          // show list          System.out.println("show list DESC:");  **for** (Double element : list) {              System.out.println(element);          }      }  } |

Kết quả:

show list ASC:

15.0

16.5

20.0

show list DESC:

20.0

16.5

15.0

### 2.5.3 Sắp xếp theo tự định nghĩa

**Cách 1**: lớp Student implements giao diện **java.lang.Comparable** để cài đặt **phương thức compareTo()**. Bạn cần phải cài đặt tiêu chí để so sánh trong phương thức compareTo() để so sánh các đối tượng Student với nhau, dưới đây là ví dụ sắp xếp các đối tượng Student tăng dần theo name:

Tạo lớp Student.java

[?](https://viettuts.vn/java-collection/sorting-trong-collections)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | **package** vn.viettuts.collection;    /\*\*   \* Student class   \*   \* @author viettuts.vn   \*/  **class** Student **implements** Comparable<Student> {  **private** **int** id;  **private** String name;  **private** **int** age;  **private** String address;    **public** Student() {      }    **public** Student(**int** id, String name, **int** age, String address) {  **super**();  **this**.id = id;  **this**.name = name;  **this**.age = age;  **this**.address = address;      }        // getter & setter        @Override  **public** String toString() {  **return** "Student@id=" + id + ",name=" + name                  + ",age=" + age + ",address=" + address;      }        @Override  **public** **int** compareTo(Student student) {          // sort student's name by ASC  **return** **this**.getName().compareTo(student.getName());      }  } |

Tạo lớp CollectionsExample6.java

[?](https://viettuts.vn/java-collection/sorting-trong-collections)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | **package** vn.viettuts.collection;    **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.List;    /\*\*   \* CollectionsExample6 class   \*   \* @author viettuts.vn   \*/  **public** **class** CollectionsExample6 {  **public** **static** **void** main(String[] args) {          // create list students          List<Student> listStudents = **new** ArrayList<Student>();          // add students to list          listStudents.add(**new** Student(1, "Vinh", 19, "Hanoi"));          listStudents.add(**new** Student(2, "Hoa", 19, "Hanoi"));          listStudents.add(**new** Student(3, "Phu", 20, "Hanoi"));          listStudents.add(**new** Student(4, "Quy", 22, "Hanoi"));          // sort list student          Collections.sort(listStudents);          // show list students  **for** (Student student : listStudents) {              System.out.println(student.toString());          }      }  } |

Kết quả:

Student@id=2,name=Hoa,age=19,address=Hanoi

Student@id=3,name=Phu,age=20,address=Hanoi

Student@id=4,name=Quy,age=22,address=Hanoi

Student@id=1,name=Vinh,age=19,address=Hanoi

**Cách 2**: Tạo đối tượng nặc danh **java.util.Comparator** để cài đặt **phương thức compare()**. Bạn cần phải cài đặt tiêu chí để so sánh trong phương thức compare() để so sánh các đối tượng Student với nhau, dưới đây là ví dụ sắp xếp các đối tượng Student tăng dần theo name:

Tạo lớp Student.java

[?](https://viettuts.vn/java-collection/sorting-trong-collections)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | **package** vn.viettuts.collection;    /\*\*   \* Student class   \*   \* @author viettuts.vn   \*/  **class** Student {  **private** **int** id;  **private** String name;  **private** **int** age;  **private** String address;    **public** Student() {      }    **public** Student(**int** id, String name, **int** age, String address) {  **super**();  **this**.id = id;  **this**.name = name;  **this**.age = age;  **this**.address = address;      }        // getter & setter        @Override  **public** String toString() {  **return** "Student@id=" + id + ",name=" + name                  + ",age=" + age + ",address=" + address;      }  } |

Tạo lớp CollectionsExample7.java

[?](https://viettuts.vn/java-collection/sorting-trong-collections)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | **package** vn.viettuts.collection;    **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.Comparator;  **import** java.util.List;    /\*\*   \* CollectionsExample7 class   \*   \* @author viettuts.vn   \*/  **public** **class** CollectionsExample7 {  **public** **static** **void** main(String[] args) {          // create list students          List<Student> listStudents = **new** ArrayList<Student>();          // add students to list          listStudents.add(**new** Student(1, "Vinh", 19, "Hanoi"));          listStudents.add(**new** Student(2, "Hoa", 19, "Hanoi"));          listStudents.add(**new** Student(3, "Phu", 20, "Hanoi"));          listStudents.add(**new** Student(4, "Quy", 22, "Hanoi"));          // sort list student          Collections.sort(listStudents, **new** Comparator<Student>() {              @Override  **public** **int** compare(Student o1, Student o2) {  **return** o1.getName().compareTo(o2.getName());              }          });          // show list students  **for** (Student student : listStudents) {              System.out.println(student.toString());          }      }  } |

Kết quả:

Student@id=2,name=Hoa,age=19,address=Hanoi

Student@id=3,name=Phu,age=20,address=Hanoi

Student@id=4,name=Quy,age=22,address=Hanoi

Student@id=1,name=Vinh,age=19,address=Hanoi

# 3. Comparator trong java

Tạo đối tượng nặc danh **java.util.Comparator** để cài đặt **phương thức compare()**. Bạn cần phải cài đặt tiêu chí để so sánh trong phương thức compare() để so sánh các đối tượng Student với nhau, dưới đây là ví dụ sắp xếp các đối tượng Student tăng dần theo name:

Tạo lớp Student.java

[?](https://viettuts.vn/java-collection/comparator-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | **package** vn.viettuts.collection;    /\*\*   \* Student class   \*   \* @author viettuts.vn   \*/  **class** Student {  **private** **int** id;  **private** String name;  **private** **int** age;  **private** String address;    **public** Student() {      }    **public** Student(**int** id, String name, **int** age, String address) {  **super**();  **this**.id = id;  **this**.name = name;  **this**.age = age;  **this**.address = address;      }        // getter & setter        @Override  **public** String toString() {  **return** "Student@id=" + id + ",name=" + name                  + ",age=" + age + ",address=" + address;      }  } |

Tạo lớp ComparatorExample.java

Trong ví dụ này chúng ta tạo ra 2 đối tượng nặc danh của lớp Comparator như 2 tiêu chí sắp xếp các đối tượng của lớp Student theo **name** và **age**.

[?](https://viettuts.vn/java-collection/comparator-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | **package** vn.viettuts.collection;    **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.Comparator;  **import** java.util.List;      /\*\*   \* ComparatorExample class   \*   \* @author viettuts.vn   \*/  **public** **class** ComparatorExample {  **public** **static** **void** main(String[] args) {          // create list students          List<Student> listStudents = **new** ArrayList<Student>();          // add students to list          listStudents.add(**new** Student(1, "Vinh", 19, "Hanoi"));          listStudents.add(**new** Student(2, "Hoa", 24, "Hanoi"));          listStudents.add(**new** Student(3, "Phu", 20, "Hanoi"));          listStudents.add(**new** Student(4, "Quy", 22, "Hanoi"));            // sort list student by it's name ASC          System.out.println("sort list student by it's name ASC: ");          Collections.sort(listStudents, **new** Comparator<Student>() {              @Override  **public** **int** compare(Student o1, Student o2) {  **return** o1.getName().compareTo(o2.getName());              }          });          // show list students  **for** (Student student : listStudents) {              System.out.println(student.toString());          }            // sort list student by it's age ASC          System.out.println("sort list student by it's age ASC: ");          Collections.sort(listStudents, **new** Comparator<Student>() {              @Override  **public** **int** compare(Student o1, Student o2) {  **return** o1.getAge() > o2.getAge() ? 1 : -1;              }          });          // show list students  **for** (Student student : listStudents) {              System.out.println(student.toString());          }      }  } |

Kết quả:

sort list student by it's name ASC:

Student@id=2,name=Hoa,age=24,address=Hanoi

Student@id=3,name=Phu,age=20,address=Hanoi

Student@id=4,name=Quy,age=22,address=Hanoi

Student@id=1,name=Vinh,age=19,address=Hanoi

sort list student by it's age ASC:

Student@id=1,name=Vinh,age=19,address=Hanoi

Student@id=3,name=Phu,age=20,address=Hanoi

Student@id=4,name=Quy,age=22,address=Hanoi

Student@id=2,name=Hoa,age=24,address=Hanoii

## 3.1 So sánh với Comparator

1 dieu kien (them reversed neu giam dan) //

Collections.sort(t.subList(0, maxInd+1), Comparator.comparing(Brick::getPrice).reversed());

// 2 dieu kien them thenComparing, dao nguoc dieu kien nao thi them ,Comparator.reverseOrder() vao (dieu kien comparingInt, comparingDouble thi tao ham moi)

Collections.sort(t.subList(0,maxInd),Comparator.comparingInt(Brick::getMinusPrice).thenComparing(Brick::getPlace,Comparator.reverseOrder())); }

# 4. Comparable trg java

## 4.1 Comparable vs Tên

Lớp Student implements giao diện **java.lang.Comparable** để cài đặt **phương thức compareTo()**. Bạn cần phải cài đặt tiêu chí để so sánh trong phương thức compareTo() để so sánh các đối tượng Student với nhau, dưới đây là ví dụ sắp xếp các đối tượng Student tăng dần theo name:

Tạo lớp Student.java

[?](https://viettuts.vn/java-collection/comparable-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | **package** vn.viettuts.collection;    /\*\*   \* Student class   \*   \* @author viettuts.vn   \*/  **class** Student **implements** Comparable<Student> {  **private** **int** id;  **private** String name;  **private** **int** age;  **private** String address;    **public** Student() {      }    **public** Student(**int** id, String name, **int** age, String address) {  **super**();  **this**.id = id;  **this**.name = name;  **this**.age = age;  **this**.address = address;      }        // getter & setter        @Override  **public** String toString() {  **return** "Student@id=" + id + ",name=" + name                  + ",age=" + age + ",address=" + address;      }        @Override  **public** **int** compareTo(Student student) {          // sort student's name by ASC  **return** **this**.getName().compareTo(student.getName());      }  } |

Tạo lớp ComparableExample.java

[?](https://viettuts.vn/java-collection/comparable-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | **package** vn.viettuts.collection;    **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.List;    /\*\*   \* ComparableExample class   \*   \* @author viettuts.vn   \*/  **public** **class** ComparableExample {  **public** **static** **void** main(String[] args) {          // create list students          List<Student> listStudents = **new** ArrayList<Student>();          // add students to list          listStudents.add(**new** Student(1, "Vinh", 19, "Hanoi"));          listStudents.add(**new** Student(2, "Hoa", 19, "Hanoi"));          listStudents.add(**new** Student(3, "Phu", 20, "Hanoi"));          listStudents.add(**new** Student(4, "Quy", 22, "Hanoi"));          // sort list student          Collections.sort(listStudents);          // show list students  **for** (Student student : listStudents) {              System.out.println(student.toString());          }      }  } |

Kết quả:

Student@id=2,name=Hoa,age=19,address=Hanoi

Student@id=3,name=Phu,age=20,address=Hanoi

Student@id=4,name=Quy,age=22,address=Hanoi

Student@id=1,name=Vinh,age=19,address=Hanoi

## 4.2 Comparable 2 yếu tố (Tên + gpa)

public int compareTo(Student student) {

// sort student's name by ASC

int nameComparison = this.getName().compareTo(student.getName());

if (nameComparison != 0) {

return nameComparison;

} else {

// If names are the same, compare GPAs

if (this.getGpa() < student.getGpa()) {

return -1;

} else if (this.getGpa() > student.getGpa()) {

return 1;

} else {

return 0;

}

} }

# 5. 1 số Thuật toán hay gặp nhất

## 5.1 Tìm kiếm tên theo chữ cái đầu tiên trong List

public List<Student> find\_by\_partial\_name(List<Student> g , String m ){

List<Student> f = new ArrayList<Student>();

for(int i=0 ; i< g.size() ; i++) {

if (g.get(i).getName().length()>=m.length())

if (g.get(i).getName().substring(0, m.length()).equalsIgnoreCase(m)){

f.add(g.get(i)) ;

}

}

return f ;

}

### 5.1.1 Xóa các obj trong 1 list mà name bắt đầu vs sâu cho trước

public void f1(List<Fan> a, String code) {

for (int i=a.size() -1 ; i >=0 ; i-- ){

if(a.get(i).getCode().length() >= code.length())

if (a.get(i).getCode().substring(0, code.length()).equals(code))

// if (a.get(i).getCode().startsWith(code ))

a.remove(i) ;

}

}

### 5.1.2 Đếm và trả lại số lượng obj trg list a mà price <= price cho trước

public int f2(List<Fan> a, double price) {

int counter = 0 ;

for(int i=a.size() -1 ; i >=0 ; i-- ){

if (a.get(i).getPrice()<= price ) counter ++ ;

}

return counter ;

}

### 5.1.3 Di chuyển các Obj trg list a có price >= price cho lên đầu của list mà ko làm thay đổi thứ tự gốc của list đó

@Override

public void f3(List<Fan> a, double price ) {

for(int i= 0 ; i < a.size() ; i++ ){

if (a.get(i).getPrice()< price )

{

a.add(a.size(),a.get(i)) ;

a.remove(i);

if (i== a.size()-1) break ;

i-- ;

}

}

## 5.2 Chuẩn hóa sâu

### 5.2.1 Viết hoa , ko viết hoa và xóa dấu cách

#### 5.2.1.1 Viết hoa tất cả các ký từ đầu tiên của từ

Public String chuanhoaten (String name )

{

String result = “” ;

String[] tokens = name.split(“[\\s\\.+](\\\\s\\\\.+)”) ;

For (int i=0 ; i< tokens.length ; i++ ) {

String firstChar = tokens[i].substring(0,1).toUpperCase() ;

String otherChar = tokens[i].substring(1).toLowerCase() ;

Return += firstChar + otherChar + “ “ ;

}

Return result.trim() ;

}

#### 5.2.1.2

public String Nomalization (String s )

{

// thêm dấu cách vào sau dấu . ,

s = s.toLowerCase().replace(".", ". ").replace(",", ", ").replaceAll("\\s+", " ");

// xóa dấu cách ở trước dấu , .

s = s.replace(" .", ".").replace(" ,", ",");

// Viết hoa các chữ cái đầu của từ đứng sau dấu .

char[] a = s.toCharArray();

for (int i = 2; i < a.length; i++) {

if (a[i - 1] == ' ' && a[i - 2] == '.') {

a[i] = Character.toUpperCase(a[i]);

}

}

s = new String(a);

return s.trim();

}

### 5.2.2 Kiểm tra digit , letter

Kiểm tra xem 1 chuỗi có chứa ít nhất 1 even digit :

(?=.\*[02468].\*).+

Kiểm tra xem 1 chuỗi có chứa ít nhất 1 digit :

(?=.\*\d+.\*).+

Kiểm tra xem 1 chuỗi có chứa ít nhất 1 letter

(?=.\*[a-zA-Z].\*).+

VD:

 public int f1(String str) {

       int  count = 0 ;

        String[] t = str.split("\\s+") ;

        for (int i = 0; i < t.length; i++) {

        if (t[i].matches("(?=.\*[02468].\*).+"))

        {

            count++ ;

        } }

        return count ;

    }

#### 5.2.2.1 Tách các số và các chữ cái viết hoa trong sâu cho trước

@Override

public String getCode(String rc ) {

String results = "";

for (int i=0 ; i< rc.length(); i++){

if ((rc.charAt(i)>='0' && rc.charAt(i)<='9') || (rc.charAt(i)>='A'&& rc.charAt(i)<='Z'))

results += rc.charAt(i) ;

}

return results ;

}

#### 5.2.2.2 Tìm digits chẵn trong 1 String và cộng lấy tổng

@Override

public int sum(String rc ) {

int results = 0 ;

for (int i=0 ; i< rc.length(); i++){

if (rc.charAt(i)>='0' && rc.charAt(i)<='9' && rc.charAt(i)%2==0 )

results += Character.getNumericValue(rc.charAt(i));

// C2: results += rc.charAt(i) - '0' ;

}

return results ;

}

### 5.2.3 Kiểm tra Palindrome (đối xứng trg từ )

public boolean IsPalindrome(String s){

        int l = s.length() ;

        int i=0 ;

        while (i<l) {

            if (s.charAt(i)!= s.charAt(l-1)) {

                return false;

                }

            i++ ;l-- ;

        } return true ;

    }

public boolean IsPalindrome(String s){

int l = s.length() ;

int i=0 ;

while (i<l) {

if (s.charAt(i)!= s.charAt(l-1)) {

return false;

}

i++ ;l-- ;

} return true ;

}

## 5.3 Tìm số lớn thứ 2 trong list

@Override

public void f3(List<Brick> list) {

double maxprice = 0 ;

double temp = 0 ;

int ga =0 ;

for(int i=0 ; i < list.size(); i++ ){

if(list.get(i).getPrice() > maxprice ){

maxprice = list.get(i).getPrice() ;

}

}

for(int i=0 ; i < list.size(); i++ ){

if(list.get(i).getPrice() > temp && list.get(i).getPrice() < maxprice ){

temp = list.get(i).getPrice() ;

ga=i ;

}

}

list.remove(ga);

}

## 5.4 Cho sâu , tìm các sâu con chứa toàn digit đầu tiên và cuối cùng , tính tổng các digits của 2 sâu đó

public int sum(String string) {

String[] t = string.split("\\s+");

String ficha = "" , lacha = "" ;

int u1 =0 , u2=0 ;

for (int i = 0; i < t.length; i++) {

if (t[i].matches("[0-9]+") )

lacha = t[i] ;

}

for(int i=0 ; i<lacha.length(); i++){

u1 += Character.getNumericValue(lacha.charAt(i)) ;

}

for (int i = t.length-1; i >= 0; i--) {

if (t[i].matches("[0-9]+") )

ficha = t[i];

}

for(int i=0 ; i<ficha.length(); i++){

u2 += Character.getNumericValue(ficha.charAt(i)) ;

}

return u1+u2 ;

}

## 5.5 Đảo ngược 1 sâu

for(int i=ficha.length()-1 ; i>=0 ; i--){

finalficha += ficha.charAt(i) ;

}

## 5.6 Phương thức replace()

### 5.6.1 **Cú pháp của phương thức replace()**

## 5.7 Cho 1 sâu , tìm từ trong sâu có các chữ cái không theo thứ tự alphabet

VD: abc bca

=> Xóa bca

package com.mycompany.testproject;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.Scanner;

public class TestProject {

public static void main(String[] args) {

String s="ab bc de hoa ef la la";

System.out.println(edit(s));

}

public static String edit(String s) {

List<String

public static String edit(String s)

{

List<String> list = new ArrayList<>

(Arrays.asList(s.split("\\s+")));

int ind=-1;

for (int i=0; i<list.size(); i++) {

String word = list.get(i);

for (int j=1; j<word.length();j++) {

if

(Character.compare(word.charAt(j),word.charAt(j-1))<0)

ind=i;

} if (ind!=-1)

{ list.remove(ind);

break; }

}