**List Interface trong Java**

**1. Đặc điểm**

List Interface là một loại Interface Collection. Như đã nói trong bài [Tổng quan về Collection trong Java](https://freetuts.net/tong-quan-ve-collections-trong-java-1070.html" \o "tong quan ve collections trong java 1070 html" \t "https://freetuts.net/_blank), các phần tử của List Interface được sắp xếp có thứ tự và giá trị của các phần tử này có thể trùng nhau.

Các phần tử trong List có thể được truy cập, thêm, sửa hoặc xóa thông qua vị trí của nó trong danh sách, và phần tử đầu tiên trong List sẽ có chỉ số là 0.

Dưới đây là hình ảnh minh họa một List Interface:



Trong hình trên các bạn thấy List này có 5 phần tử là apple, lemon, banana, orange và grape. Phần tử đầu tiên là apple có chỉ số là 0 và phần tử cuối cùng là grape có chỉ số là 4.

**2. Các phương thức phổ biến**

**2.1. Tạo mới một List Interface**

Trong bài trước, tôi có trình bày những thành phần của Collections Framework, trong đó tôi có đề cập đến **Implementations**là sự triển khai các Interface (ví dụ như các Class), **vì vậy để khai báo một List chúng ta cần phải dùng đến các Class để triển khai nó**, trong phần này chúng ta sẽ sử dụng 2 loại phổ biến nhất là ArrayList và LinkedList.

**Cú pháp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public static void main(String[] args) {      // khai báo List Interface tên listString có kiểu là String      // và sử dụng Class là ArrayList để triển khai      // ArrayList là 1 Class Collection      // các phần tử trong listString cũng có kiểu là String      List<String> listString = new ArrayList<String>();        // khai báo List Interface tên listInteger có kiểu là Integer      // và sử dụng Class là LinkedList để triển khai      // LinkedList là 1 Class Collection      // các phần tử trong listInteger cũng có kiểu là Integer      List<Integer> listInteger = new LinkedList<Integer>();  } |

Ngoài ra, nếu chúng ta khai báo một List có Class triển khai là ArrayList và chúng ta đã biết trước số lượng phần tử thì chúng ta có thể khai báo kèm với số lượng phần tử của nó. Ví dụ dưới đây sẽ khai báo một List có tên là listFloat, kiểu là Float và có 1000 phần tử:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3 | public static void main(String[] args) {      List<Float> listFloat = new ArrayList<Float>(1000);  } |

**Lưu ý:** Để khai báo List chúng ta cần phải import gói thư viện java.util.List, đối với ArrayList thì import gói thư viện java.util.ArrayList và với LinkedList thì import gói thư viện java.util.LinkedList. Đây đều là 3 gói thư viện có sẵn của Java. Cú pháp import như sau:

**Cú pháp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | // Khai báo List  // thì import gói thư viện java.util.List  import java.util.List;  public class TênClass {      // ...  }    // Khai báo ArrayList  // thì import gói thư viện java.util.ArrayList  import java.util.ArrayList;  public class TênClass {      // ...  }    // Khai báo LinkedList  // thì import gói thư viện java.util.LinkedList  import java.util.LinkedList;  public class TênClass {      // ...  } |

**2.2. Hiển thị các phần tử có trong List**

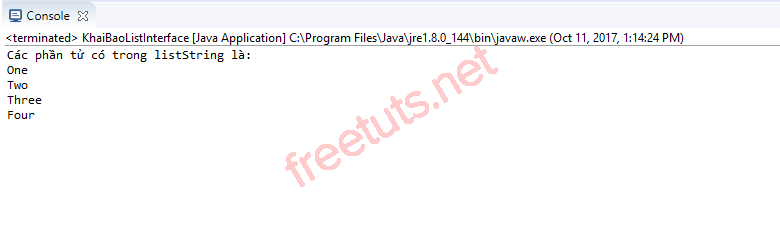
Để hiển thị các phần tử có trong List, chúng ta có các cách như sau:

**Sử dụng vòng lặp**for**thông thường.**

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | public static void main(String[] args) {      // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new LinkedList<String>();        // thêm các phần tử      listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");        // duyệt theo kích thước của listString      // và sau đó lấy phần tử tại vị trí thứ i thông qua hàm get()      // sau đó hiển thị giá trị phần tử đó ra      for (int i = 0; i < listString.size(); i++) {          System.out.println(listString.get(i));      }  } |

**Kết quả sau khi biên dịch chương trình:**

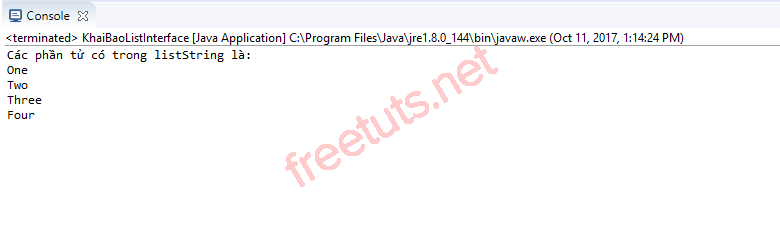


**Sử dụng vòng lặp**for**cải tiến duyệt theo đối tượng trong danh sách.**

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | public static void main(String[] args) {      // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new LinkedList<String>();        // thêm các phần tử      listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");        // hiển thị các phần tử có trong listString      // bằng cách sử dụng vòng lặp for duyệt theo đối tượng      // trong đó kiểu dữ liệu của biến element      // phải trùng với kiểu dữ liệu của listString      System.out.println("Các phần tử có trong listString là: ");      for (String element : listString) {          System.out.println(element);      }  } |

**Kết quả sau khi biên dịch chương trình:**



Bên cạnh đó, trong bài trước tôi có đề cập về cách sử dụng Iterator để lặp các phần tử trong một danh sách. Ví dụ dưới đây sẽ minh họa 2 cách sử dụng Iterator để hiển thị các phần tử có trong Java:

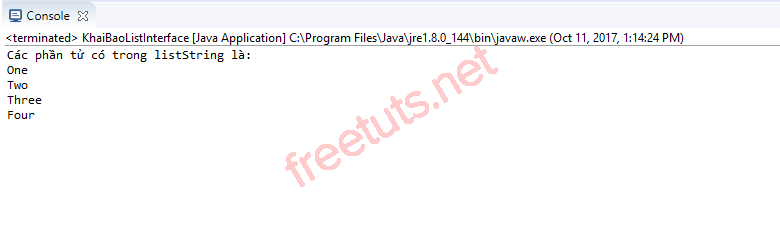
**Sử dụng Iterator.**

Iterator được sử dụng chung cho cả List, Set và Map. Để sử dụng được Iterator chúng ta cần phải import gói thư viện java.util.Iterator của Java:

**Ví dụ**

|  |  |
| --- | --- |
| 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 | public static void main(String[] args) {      // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new LinkedList<String>();        // khai báo một Iterator      Iterator<String> iterator = null;        // thêm các phần tử      listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");        System.out.println("Các phần tử có trong listString là: ");        // Lấy ra đối tượng iterator để truy cập vào các phần tử của tập hợp.      // Đối tượng iterator này chỉ chứa các String.      // Lúc này iterator sẽ trỏ vào      // chỉ số trước chỉ số của phần tử đầu tiên trong listString      iterator = listString.iterator();        // Kiểm tra xem Iterator còn phần tử tiếp theo hay không?      // Nếu có thì sẽ di chuyển vị trí mà iterator      // đang trỏ vào sang vị trí của phần tử kế tiếp      // và hiển thị phần tử đó ra      while (iterator.hasNext()) {          System.out.println(iterator.next());      }  } |

**Kết quả sau khi biên dịch chương trình:**



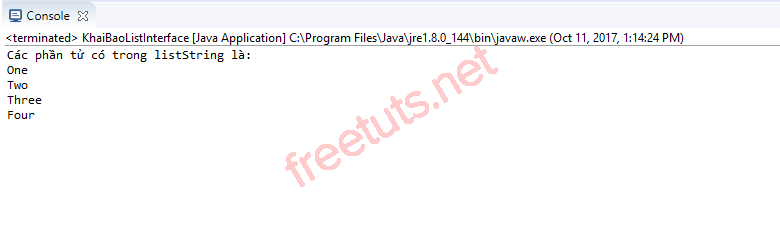
**Sử dụng ListIterator.**

ListIterator chỉ áp dụng riêng đối với List (List Interface, ArrayList, LinkedList,...), không áp dụng đối với Set. Để sử dụng được ListIterator chúng ta cần phải import gói thư viện java.util.ListIterator của Java:

**Bài giải**

|  |  |
| --- | --- |
| 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 | public static void main(String[] args) {      // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new LinkedList<String>();        // khai báo ListIterator      ListIterator<String> listIterator = null;        // thêm các phần tử      listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");        // Lấy ra đối tượng listIterator để truy cập vào các phần tử của tập hợp.      // Đối tượng listIterator này chỉ chứa các String.      // Lúc này listIterator sẽ trỏ vào      // chỉ số trước chỉ số của phần tử đầu tiên trong listString      listIterator = listString.listIterator();        System.out.println("Các phần tử có trong listString là: ");        // Kiểm tra xem ListIterator còn phần tử tiếp theo hay không?      // Nếu có thì sẽ di chuyển vị trí mà listIterator      // đang trỏ vào sang vị trí của phần tử kế tiếp      // và hiển thị phần tử đó ra      while (listIterator.hasNext()) {          System.out.println(listIterator.next());      }  } |

**Kết quả sau khi biên dịch chương trình:**



**2.3 Thêm phần tử vào trong List Interface**

**Thêm phần tử sử dụng phương thức add().**

**Ví dụ**

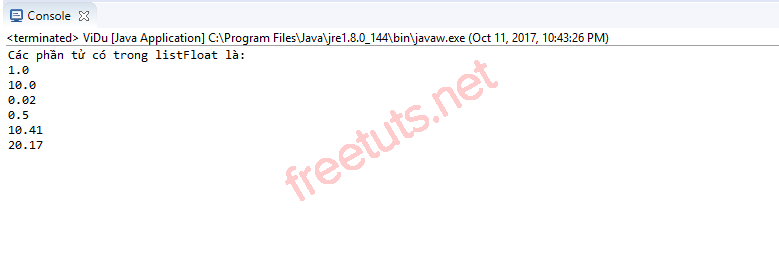
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public static void main(String[] args) {      List<String> listString = new ArrayList<String>();        listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");      listString.add("Five");  } |

Ngoài ra, phương thức add() còn cho phép chúng ta thêm mới một phần tử mới vào một vị trí bất kỳ trong danh sách.

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | public static void main(String[] args) {      List<Float> listFloat = new ArrayList<Float>(1000);        listFloat.add(1.0f);      listFloat.add(10f);      listFloat.add(0.02f);      listFloat.add(10.41f);      listFloat.add(20.17f);        // thêm phần tử có giá trị 0.5f      // vào vị trí số 3 trong danh sách listFloat      listFloat.add(3, 0.5f);        System.out.println("Các phần tử có trong listFloat là: ");      for (float numberFloat : listFloat) {          System.out.println(numberFloat);      }  } |

Các bạn thấy ví dụ trên tôi tiến hành thêm vào trong danh sách listFloat một phần tử mới tại vị trí số 3 và có giá trị là 0.5f. Lúc này chương trình sẽ chèn phần tử có giá trị là 0.5f vào vị trí của phần tử đang có chỉ số là 3 trong danh sách ban đầu đó là phần tử 10.41f, và sau khi chèn thì phần tử này sẽ có chỉ số là 4.



**Thêm phần tử sử dụng phương thức addAll().**

Java cung cấp cho chúng ta phương thức addAll() để thêm tất cả các phần tử của một List vào cuối List đã tồn tại. **Lưu ý:** kiểu dữ liệu của 2 List này phải giống nhau. Ví dụ dưới đây sẽ khởi tạo 2 List đó là listString và listWord có cùng kiểu dữ liệu là String. Phương thức addAll() sẽ thêm các phần tử có trong listWord vào cuối danh sách của listString.

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | public static void main(String[] args) {      // khai báo một List Interface có tên là listWord      // listWord có kiểu là String      List<String> listWord = new ArrayList<String>();      listWord.add("Apple");      listWord.add("Banana");      listWord.add("Orange");        // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new ArrayList<String>();      listString.add("Lemon");      listString.add("Grape");        // thêm các phần tử của listWord      // vào cuối của listString      listString.addAll(listWord);        System.out.println("Các phần tử có trong listString là: ");      for (String str : listString) {          System.out.println(str);      }  } |

**Kết quả sau khi biên dịch chương trình:**



Bên cạnh đó, chúng ta có thể sử dụng phương thức addAll() để thêm tất cả các phần tử của một List khác vào 1 vị trí bất kỳ trong List đã tồn tại. Ví dụ dưới đây sẽ thêm các phần tử của listWord vào vị trí số 1 trong listString:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | public static void main(String[] args) {      // khai báo một List Interface có tên là listWord      // listWord có kiểu là String      List<String> listWord = new ArrayList<String>();      listWord.add("Apple");      listWord.add("Banana");      listWord.add("Orange");        // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new ArrayList<String>();      listString.add("Lemon");      listString.add("Grape");        // thêm các phần tử của listWord      // vào vị trí số 1 của listString      listString.addAll(1, listWord);        System.out.println("Các phần tử có trong listString là: ");      for (String str : listString) {          System.out.println(str);      }  } |

Kết quả sau khi biên dịch chương trình:



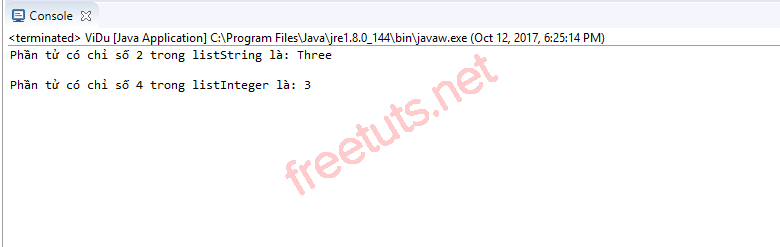
**2.4. Truy cập phần tử**

Java cung cấp cho chúng ta phương thức get() để truy cập đến 1 phần tử bất kỳ trong List thông qua chỉ số của phần tử đó. Ví dụ dưới đây sẽ truy cập phần tử có chỉ số là 2 trong 1 List có tên là listString và truy cập phần tử có chỉ số là 4 trong 1 List có tên listInteger:

**Ví dụ**

|  |  |
| --- | --- |
| 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 | public static void main(String[] args) {      // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new LinkedList<String>();        // thêm các phần tử      listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");        // khai báo một List Interface có tên là listInteger      // listWord có kiểu dữ liệu là Integer      List<Integer> listInteger = new LinkedList<Integer>();        // thêm các phần tử      listInteger.add(1);      listInteger.add(8);      listInteger.add(0);      listInteger.add(32);      listInteger.add(3);      listInteger.add(12);        // truy cập phần tử có chỉ số 2 trong listString      // tương ứng với chuỗi "Three"      // vì listString có kiểu là String      // nên các phần tử con của nó cũng có kiểu dữ liệu là String      System.out.print("Phần tử có chỉ số 2 trong listString là: ");      String str = listString.get(2);      System.out.println(str);        // truy cập phần tử có chỉ số 4 trong listInteger      // tương ứng với số 3      System.out.print("\nPhần tử có chỉ số 4 trong listInteger là: ");      int number = listInteger.get(4);      System.out.println(number);  } |

**Kết quả sau khi biên dịch chương trình:**



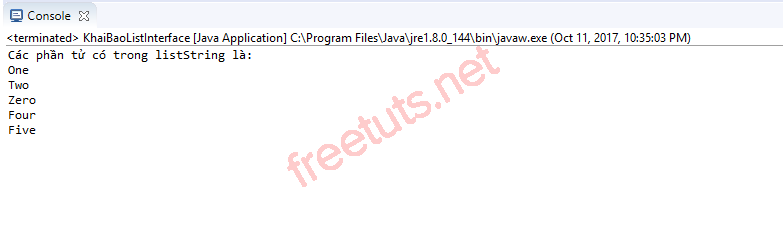
**2.5. Cập nhật giá trị của phần tử**

Để cập nhật giá trị của phần tử trong List, 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. Để hiểu kỹ hơn về phương thức này các bạn hãy xem ví dụ sau:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | public static void main(String[] args) {      // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new ArrayList<String>();        // thêm các phần tử      listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");      listString.add("Five");        // cập nhật giá trị của phần tử thứ 2      // trong listString bằng phương thức set()      listString.set(2, "Zero");        // hiển thị các phần tử trong listString      System.out.println("Các phần tử có trong listString là: ");      for (String stringNumber : listString) {          System.out.println(stringNumber);      }  } |

Các bạn thấy trong ví dụ trên, tôi đã tiến hành thay thế giá trị của phần tử có chỉ số là 2 trong danh sách. Kết quả là chương trình sẽ thay thế chuỗi "Three" của phần tử này thành "Zero". Kết quả sau khi biên dịch chương trình như sau:



**2.6. Xóa phần tử**

Để xóa một phần tử khỏi List, Java cung cấp cho chúng ta 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).

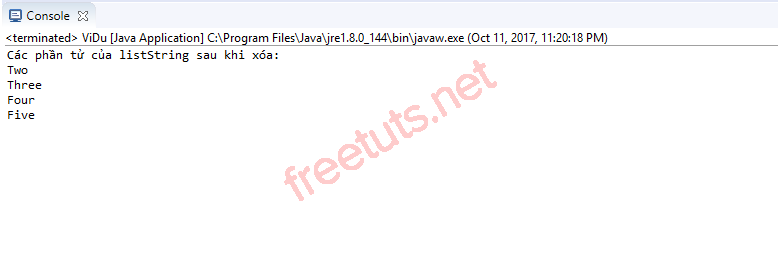
**Xóa dựa vào chỉ số của phần tử.**

Ví dụ dưới đây sẽ xóa một phần tử có chỉ số là 0 trong List có tên là listString:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | public static void main(String[] args) {      // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new ArrayList<String>();        // thêm các phần tử      listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");      listString.add("Five");        // xóa phần tử có chỉ số là 0      listString.remove(0);        System.out.println("Các phần tử của listString sau khi xóa: ");      for (String str : listString) {          System.out.println(str);      }  } |

Chúng ta thấy trong ví dụ trên danh sách listString có 5 phần tử và phần tử có chỉ số 0 là "One". Sau khi xóa thì kết quả hiển thị danh sách các phần tử của listString như sau:



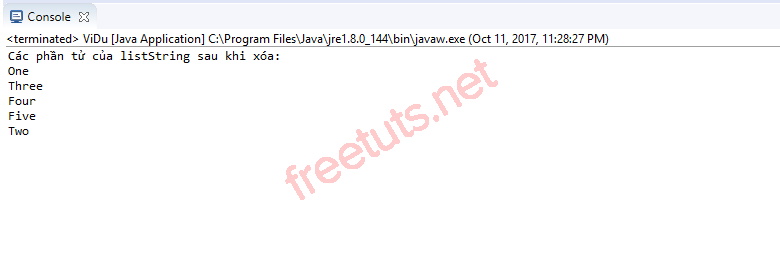
**Xóa trực tiếp một phần tử.**

Với cách xóa này, nếu trong List có 2 phần tử giống nhau thì chương trình sẽ xóa phần tử đầu tiên trong 2 phần tử giống nhau đó ra khỏi danh sách (tức là phần tử có chỉ số index nhỏ hơn). Ví dụ dưới đây sẽ xóa phần tử "Two" ra khỏi listString:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | public static void main(String[] args) {      // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new ArrayList<String>();        // thêm các phần tử      listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");      listString.add("Five");      listString.add("Two");        // xóa phần tử "Two" khỏi danh sách      listString.remove("Two");        System.out.println("Các phần tử của listString sau khi xóa: ");      for (String str : listString) {          System.out.println(str);      }  } |

Trong ví dụ này, danh sách listString có 2 phần tử là "Two" nên khi xóa thì chương trình sẽ tìm đến phần tử có chỉ số index thấp hơn và xóa nó đi. Kết quả sau khi biên dịch chương trình thì phần tử "Two" có chỉ số là 1 sẽ bị xóa:



**2.7. Tìm kiếm một phần tử**

Để tìm kiếm một phần tử trong List thì chúng ta có 3 phương pháp tìm kiếm như sau:

**Tìm kiếm trực tiếp phần tử.**

Để tìm kiếm trực tiếp phần tử, chúng ta sẽ sử dụng phương thức contains(). Kết quả trả về là true nếu tìm thấy, ngược lại trả về false.

**Cú pháp**

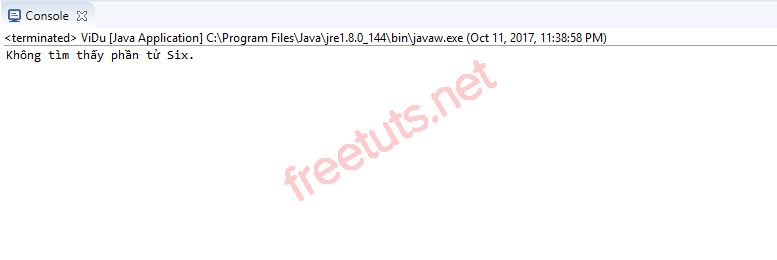
boolean contains(phần\_tử\_cần\_tìm);

Ví dụ dưới đây sẽ tìm kiếm phần tử "Six" trong danh sách listString. Nếu tìm thấy thì thông báo "Có phần tử Six trong listString.", ngược lại hiển thị thông báo "Không tìm thấy phần tử Six.".

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | public static void main(String[] args) {      // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new ArrayList<String>();        // thêm các phần tử      listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");      listString.add("Five");      listString.add("Two");        // tìm kiếm phần tử "Six" trong danh sách      if (listString.contains("Six")) {          System.out.println("Có phần tử Six trong listString.");      } else {          System.out.println("Không tìm thấy phần tử Six.");      }  } |

Trong listString không có phần tử "Six" nên kết quả biên dịch của chương trình như sau:



**Tìm kiếm vị trí xuất hiện đầu tiên của 1 phần tử trong List.**

Để tìm kiếm vị trí xuất hiện đầu tiên của 1 phần tử trong List, 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 danh sách, ngược lại nếu không tìm thấy trả về -1.

**Cú pháp**

int indexOf(phần\_tử\_cần\_tìm);

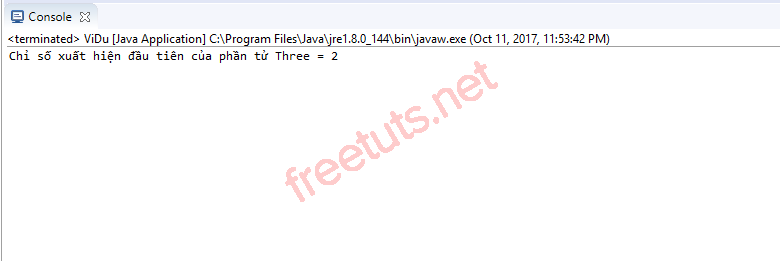
Ví dụ dưới đây sẽ tìm kiếm vị trí xuất hiện đầu tiên của phần tử "Three" trong listString:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | public static void main(String[] args) {      // khai báo List Interface có tên là listString      // kiểu dữ liệu là String      List<String> listString = new ArrayList<String>();        // thêm các phần tử      listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");      listString.add("Five");      listString.add("Three");        // tìm kiếm phần tử "Three" trong danh sách      int firstIndex = listString.indexOf("Three");      System.out.println("Chỉ số xuất hiện đầu tiên của phần tử Three = " +          firstIndex);  } |

Vì trong listString có 2 phần tử "Three" nên kết quả của ví dụ trên sẽ hiển thị chỉ số của phần tử "Three" đầu tiên, đó là chỉ số 2.

Kết quả sau khi biên dịch chương trình:



**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 List, 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 danh sách, ngược lại nếu không tìm thấy trả về -1.

**Cú pháp**

int lastIndexOf(phần\_tử\_cần\_tìm);

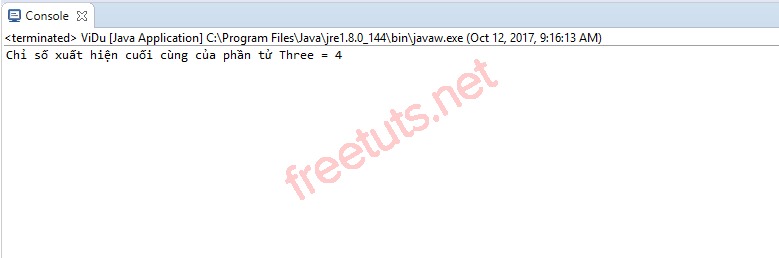
Ví dụ dưới đây sẽ tìm kiếm vị trí xuất hiện cuối cùng của phần tử "Three" trong listString:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public static void main(String[] args) {      List<String> listString = new ArrayList<String>();        listString.add("One");      listString.add("Two");      listString.add("Three");      listString.add("Four");      listString.add("Three");      listString.add("Five");        // tìm kiếm vị trí xuất hiện cuối cùng      // của phần tử "Three" trong listString      int lastIndex = listString.lastIndexOf("Three");      System.out.println("Chỉ số xuất hiện cuối cùng của phần tử Three = " +          lastIndex);  } |

Vì trong listString có 2 phần tử "Three" nên kết quả của ví dụ trên sẽ hiển thị chỉ số của phần tử "Three" cuối cùng, đó là chỉ số 2.

Kết quả sau khi biên dịch chương trình:



**2.8. Sắp xếp các phần tử**

Để sắp xếp các phần tử trong List, Java cung cấp cho chúng ta một phương thức đơn giản nhất đó là sử dụng phương thức Collections.sort(). Mặc định phương thức này sẽ sắp xếp tăng dần các phần tử trong List.

**Lưu ý:** Để sử dụng được phương thức này, chúng ta cần phải import gói thư viện java.util.Collections của Java

**Cú pháp**

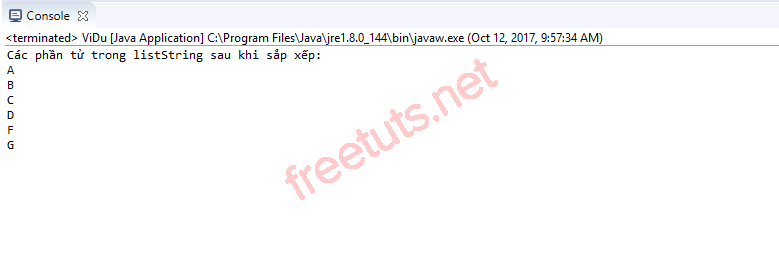
Collections.sort(tên\_list);

Ví dụ dưới đây sẽ sắp xếp các phần tử trong listString theo thứ tự tăng dần các phần tử:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | public static void main(String[] args) {      List<String> listString = new ArrayList<String>();        listString.add("F");      listString.add("B");      listString.add("D");      listString.add("C");      listString.add("G");      listString.add("A");        // sắp xếp các phần tử trong listString      // sử dụng phương thức Collections().sort()      Collections.sort(listString);        System.out.println("Các phần tử trong listString sau khi sắp xếp: ");      for (String str : listString) {          System.out.println(str);      }  } |

Kết quả sau khi biên dịch chương trình:



### **2.9. Sao chép danh sách**

Đối với List, chúng ta có thể sao chép các phần tử của 1 List này vào trong 1 List khác bằng cách sử dụng phương thức Collections.copy(). **Lưu ý:** Khi sử dụng phương thức này để sao chép thì số phần tử của List cần sao chép (List nguồn) phải lớn hơn hoặc bằng với số phần tử của danh sách sao chép (List đích).

**Cú pháp**

|  |  |
| --- | --- |
| 1 | Collections.copy(danh\_sách\_đích, danh\_sách\_nguồn); |

Ví dụ dưới đây sẽ sao chép các phần tử của listSource vào listDest và hiển thị các phần tử của listDest ra màn hình:

**Ví dụ**

|  |  |
| --- | --- |
| 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 | public static void main(String[] args) {      // danh sách nguồn      List<String> listSource = new ArrayList<String>();        listSource.add("A");      listSource.add("B");      listSource.add("C");      listSource.add("D");        // danh sách đích      // số phần tử của listDest phải lớn hơn hoặc bằng      // với số phần tử của listSource      List<String> listDest = new ArrayList<String>();        listDest.add("V");      listDest.add("W");      listDest.add("X");      listDest.add("Y");      listDest.add("Z");        // sao chép các phần tử của listSource      // vào trong listDest      Collections.copy(listDest, listSource);        System.out.println("Các phần tử có trong listDest: ");      for (String str : listDest) {          System.out.println(str);      }  } |

Các bạn thấy trong ví dụ này, listSource có 4 phần tử là "A", "B", "C" và "D", còn listDest có 5 phần tử là "V", "W", "X", "Y", "Z" nên khi biên dịch, chương trình sẽ tiến hành sao chép 4 phần tử trong listSource và thay thế 4 phần tử này cho 4 phần tử đầu tiên trong listDesc, và phần tử cuối cùng của listDest là "Z" vẫn giữ nguyên.



### **2.10. Hoán vị các phần tử**

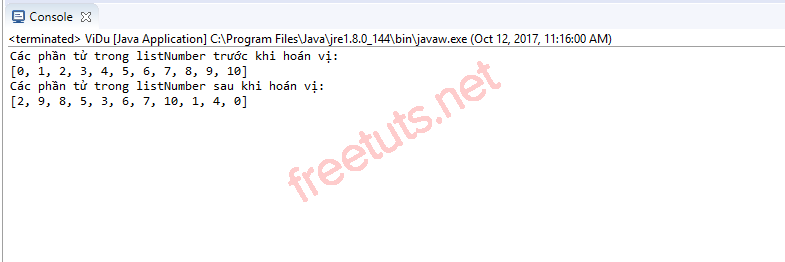
Để hoán vị các phần tử trong List, chúng ta sẽ sử dụng phương thức Collections.shuffle(). **Lưu ý**: phải import gói thư viện java.util.Collections.

Ví dụ dưới đây sẽ minh họa cách sử dụng Collections.shuffle() để hoán vị các phần tử của listNumber:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | public static void main(String[] args) {      // tạo 1 listNumber có kiểu dữ liệu là Integer      List<Integer> listNumber = new ArrayList<Integer>();        for (int i = 0; i <= 10; i++) {          listNumber.add(i);      }        System.out.println("Các phần tử trong listNumber trước khi hoán vị: ");      // hiển thị các phần tử trong listNumber ở dạng mảng      System.out.println(listNumber);        Collections.shuffle(listNumber);        System.out.println("Các phần tử trong listNumber sau khi hoán vị: ");      System.out.println(listNumber);  } |

Kết quả sau khi biên dịch chương trình (lưu ý kết quả sau mỗi lần biên dịch sẽ khác nhau):



**3. Ví dụ tổng hợp**

Viết chương trình thực hiện các yêu cầu sau:

* Khai báo 1 List có Class triển khai là ArrayList, kiểu dữ liệu là String. Sau đó thêm vào phần tử là các thứ trong tuần cho List này (giá trị của các phần tử được nhập từ bàn phím).
* Hiển thị các phần tử có trong List vừa nhập bằng 2 cách: sử dụng Iterator và ListIterator.
* Thay đổi phần tử bất kỳ trong List và hiển thị lại List sau khi thay đổi.
* Xóa phần tử vừa thay đổi trong List và hiển thị lại các phần tử còn lại của List.
* Hiển thị các phần tử của List theo thứ tự từ phần tử cuối trở về phần tử đầu tiên.

**Bài giải**

|  |  |
| --- | --- |
| 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  51  52  53  54  55 | public static void main(String[] args) {      List<String> listString = new ArrayList<String>();      Scanner scanner = new Scanner(System.in);        // thêm phần tử vào listString      listString.add("Monday");      listString.add("Tuesday");      listString.add("Wednesday");      listString.add("Thursday");      listString.add("Friday");      listString.add("Saturday");      listString.add("Sunday");        // hiển thị các phần tử sử dụng Iterator      Iterator<String> iterator = listString.iterator();        System.out.println("Hiển thị phần tử sử dụng Iterator: ");      while (iterator.hasNext()) {          System.out.println(iterator.next());      }        // hiển thị các phần tử sử dụng ListIterator      ListIterator<String> listIterator = listString.listIterator();        System.out.println("\nHiển thị phần tử sử dụng ListIterator: ");      while (listIterator.hasNext()) {          System.out.println(listIterator.next());      }        System.out.println("\nHiển thị các phần tử của List theo thứ tự từ dưới lên: ");      while (listIterator.hasPrevious()) {          System.out.println(listIterator.previous());      }        // thay đổi phần tử bất kỳ trong listString      System.out.println("\nNhập vào phần tử mới: ");      String element = scanner.nextLine();      System.out.println("Nhập vào giá trị của phần tử cần thay đổi: ");      int index = scanner.nextInt();      listString.set(index, element);        System.out.println("Các phần tử có trong listString sau khi thay đổi: ");      for (String str : listString) {          System.out.println(str);      }        // xóa phần tử vừa thay đổi trong listString      listString.remove(index);        System.out.println("\nCác phần tử có trong listString sau khi xóa: ");      for (String str : listString) {          System.out.println(str);      }  } |

Kết quả sau khi biên dịch chương trình:

