**LinkedList (Danh sách liên kết) trong Java**

**1. Đặc điểm**

LinkedList (Danh sách liên kết) là một lớp triển khai của List Interface trong Collections Framework nên nó sẽ có một vài đặc điểm và phương thức tương đồng với List. 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), LinkedList là 1 cấu trúc dữ liệu lưu trữ các phần tử dưới dạng danh sách, các phần tử trong LinkedList được sắp xếp có thứ tự và có thể có giá trị giống nhau.

**2. Các thao tác cơ bản trên danh sách liên kết**

### **2.1. Tạo mới một danh sách liên kết.**

Để khai báo một danh sách liên kết, chúng ta cần phải import gói thư viện java.util.LinkedList của Java. Cú pháp import như sau:

**Cú pháp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | // Khai báo LinkedList  // thì import gói thư viện java.util.LinkedList  import java.util.LinkedList;  public class TênClass {      // ...  } |

Sau đây là ví dụ cách tạo mới một danh sách liên kết trong Java:

**Cú pháp**

|  |  |
| --- | --- |
| 1  2  3  4  5 | public static void main(String[] args) {      // khai báo 1 danh sách liên kết có tên là linkedList      // có kiểu là Integer      LinkedList<Integer> linkedList = new LinkedList<>();  } |

### **2.2. Hiển thị các phần tử có trong LinkedList.**

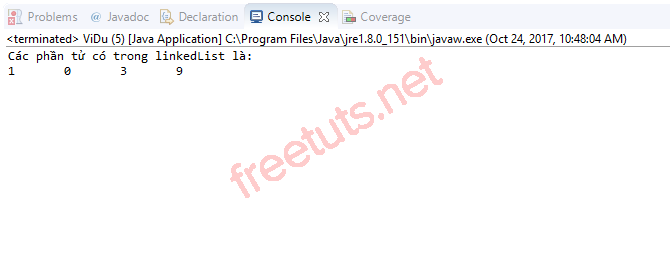
Để hiển thị các phần tử có trong LinkedList, chúng ta có các cách như sau:

**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  19  20  21 | public static void main(String[] args) {      // khai báo 1 danh sách liên kết đơn có tên là linkedList      // có kiểu là Integer      LinkedList<Integer> linkedList = new LinkedList<>();        // thêm các phần tử sử dụng phương thức add()      linkedList.add(1);      linkedList.add(0);      linkedList.add(3);      linkedList.add(9);        // sử dụng vòng lặp for      // duyệt theo kích thước của linkedList      // sử dụng phương thức linkedList.size()      // 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      System.out.println("Các phần tử có trong linkedList là: ");      for (int i = 0; i < linkedList.size(); i++) {          System.out.print(linkedList.get(i) + "\t");      }  } |

Kết quả sau khi biên dịch chương trình:

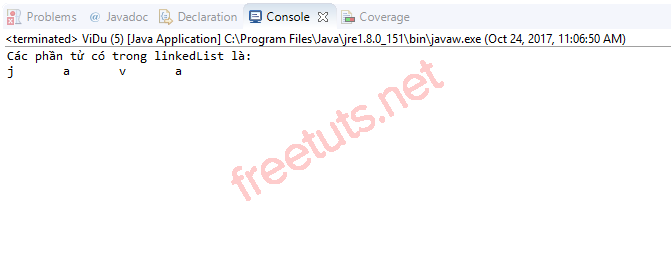


**Sử dụng vòng lặp**for**cải tiến duyệt theo đối tượng trong danh sách.**

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | public static void main(String[] args) {      // khai báo 1 danh sách liên kết đơn có tên là linkedList      // có kiểu là Character      LinkedList<Character> linkedList = new LinkedList<>();        linkedList.add('j');      linkedList.add('a');      linkedList.add('v');      linkedList.add('a');        // hiển thị các phần tử có trong linkedList      // bằng cách sử dụng vòng lặp for duyệt theo đối tượng      // trong đó kiểu dữ liệu của biến node      // phải trùng với kiểu dữ liệu của linkedList      System.out.println("Các phần tử có trong linkedList là: ");      for (char node : linkedList) {          System.out.print(node + "\t");      }  } |

Kết quả sau khi biên dịch chương trình:



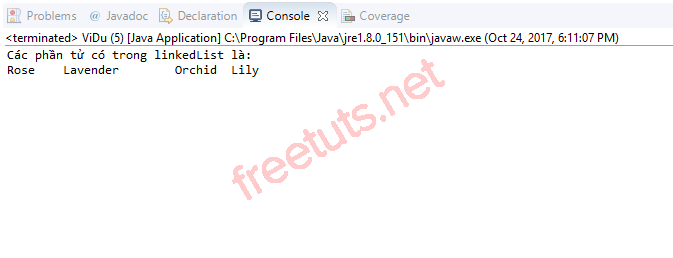
**Sử dụng Iterator.**

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 | public static void main(String[] args) {      // khai báo 1 danh sách liên kết đơn có tên là linkedList      // có kiểu là String      LinkedList<String> linkedList = new LinkedList<>();        // thêm các phần tử      // sử dụng phương thức add()      linkedList.add("Rose");      linkedList.add("Lavender");      linkedList.add("Orchid");      linkedList.add("Lily");        // khai báo một Iterator có kiểu là String      Iterator<String> iterator = linkedList.iterator();        // hiển thị các phần tử có trong linkedList      // bằng cách sử dụng Iterator      System.out.println("Các phần tử có trong linkedList là: ");      while (iterator.hasNext()) {          System.out.print(iterator.next() + "\t");      }  } |

Kết quả sau khi biên dịch chương trình:



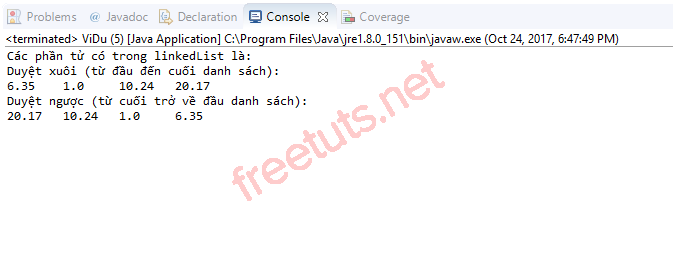
**Sử dụng ListIterator.**

Để sử dụng được ListIterator chúng ta cần phải import gói thư viện java.util.ListIterator 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 | public static void main(String[] args) {      // khai báo 1 danh sách liên kết đơn có tên là linkedList      // có kiểu là Float      LinkedList<Float> linkedList = new LinkedList<>();        // thêm các phần tử      linkedList.add(6.35f);      linkedList.add(1.0f);      linkedList.add(10.24f);      linkedList.add(20.17f);        ListIterator<Float> listIterator = linkedList.listIterator();        // hiển thị các phần tử có trong linkedList      // bằng cách sử dụng ListIterator      System.out.println("Các phần tử có trong linkedList là: ");      System.out.println("Duyệt xuôi (từ đầu đến cuối danh sách):");      while (listIterator.hasNext()) {          System.out.print(listIterator.next() + "\t");      }        System.out.println("\nDuyệt ngược (từ cuối trở về đầu danh sách):");      while (listIterator.hasPrevious()) {          System.out.print(listIterator.previous() + "\t");      }  } |

Kết quả sau khi biên dịch chương trình:



### **2.3. Thêm phần tử vào trong danh sách liên kết.**

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

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public static void main(String[] args) {      // khai báo 1 danh sách liên kết đơn có tên là linkedList      // có kiểu là Integer      LinkedList<Integer> linkedList = new LinkedList<>();        // thêm 4 phần tử (nút) có kiểu Integer      linkedList.add(6);      linkedList.add(1);      linkedList.add(10);      linkedList.add(20);  } |

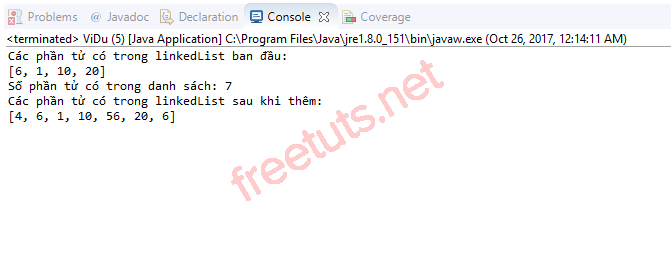
Ngoài ra, phương thức add() còn cho phép chúng ta thêm mới một phần tử mới vào một vị trí bất kỳ trong danh sách.

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | public static void main(String[] args) {      // khai báo 1 danh sách liên kết đơn có tên là linkedList      // có kiểu là Integer      LinkedList<Integer> linkedList = new LinkedList<>();        // thêm 4 phần tử (nút) có kiểu Integer      linkedList.add(6);      linkedList.add(1);      linkedList.add(10);      linkedList.add(20);        System.out.println("Các phần tử có trong linkedList ban đầu:");      System.out.println(linkedList);        // thêm phần tử có giá trị 56      // vào vị trí số 3 trong linkedList      linkedList.add(3, 56);        // thêm 2 phần tử 4 và 6 vào đầu và cuối danh sách      // sử dụng phương thức addFirst() và addLast()      linkedList.addFirst(4);      linkedList.addLast(6);        // sử dụng phương thức size() để đếm số phần tử có trong danh sách      System.out.println("Số phần tử có trong danh sách: " + linkedList.size());        System.out.println("Các phần tử có trong linkedList sau khi thêm:");      System.out.println(linkedList);  } |

Các bạn thấy ví dụ trên tôi tiến hành thêm vào trong danh sách linkedList một phần tử mới tại vị trí số 3 và có giá trị là 56. Lúc này chương trình sẽ chèn phần tử có giá trị là 56 vào vị trí của phần tử đang có chỉ số là 3 trong danh sách ban đầu đó là phần tử 20, và sau khi chèn thì phần tử này sẽ có chỉ số là 4. Bên cạnh đó, trong ví dụ này, tôi có sử dụng 2 phương thức đó là addFirst() và addLast(), phương thức addFirst() sẽ thêm phần tử vào đầu danh sách và phương thức addLast() sẽ thêm phần tử vào cuối danh sách.

Kết quả sau khi biên dịch chương trình:



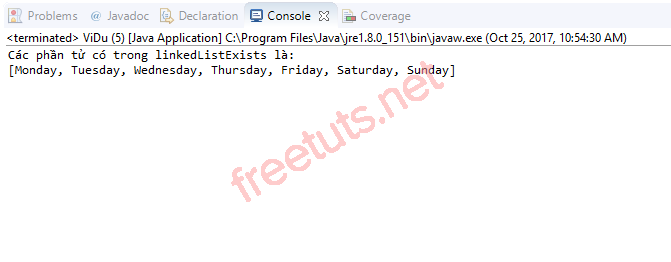
**Thêm phần tử sử dụng phương thức addAll().**

Java cung cấp cho chúng ta phương thức addAll() để thêm tất cả các phần tử của một Collection khác vào cuối 1 LinkedList đã tồn tại. **Lưu ý:** kiểu dữ liệu của Collection cần thêm vào và LinkedList này phải giống nhau. Ví dụ dưới đây sẽ khởi tạo 2 LinkedList đó là linkedList và linkedListExists có cùng kiểu dữ liệu là String. Phương thức addAll() sẽ thêm các phần tử có trong linkedList vào cuối danh sách linkedListExists.

**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 1 danh sách liên kết đơn có tên là linkedListExists      // có kiểu là String      LinkedList<String> linkedListExists = new LinkedList<>();      linkedListExists.add("Monday");      linkedListExists.add("Tuesday");      linkedListExists.add("Wednesday");      linkedListExists.add("Thursday");        // khai báo 1 danh sách liên kết đơn có tên là linkedList      // có kiểu là String      LinkedList<String> linkedList = new LinkedList<>();      linkedList.add("Friday");      linkedList.add("Saturday");      linkedList.add("Sunday");        // thêm các phần tử của linkedList      // vào cuối của linkedListExists      linkedListExists.addAll(linkedList);        System.out.println("Các phần tử có trong linkedListExists là: ");      System.out.print(linkedListExists + "\t");  } |

Kết quả sau khi biên dịch chương trình:



### **2.4. Truy cập phần tử**

Để truy cập các phần tử (các nút) có trong 1 danh sách liên kết, Java cung cấp cho chúng ta 3 phương thức đó là get(), getFirst() và getLast(). Sau đây chúng ta sẽ tìm hiểu về 3 phương thức này.

**Phương thức get().**

Công dụng: truy cập đến 1 phần tử bất kỳ trong LinkedList thông qua chỉ số của phần tử đó.

**Cú pháp**

|  |  |
| --- | --- |
| 1 | get(int index); |

, trong đó index là chỉ số của phần tử đó trong danh sách liên kết (0 <= index < (kích thước của danh sách - 1)).

**Phương thức getFirst().**

Công dụng: truy cập đến phần tử đầu tiên trong danh sách.

**Phương thức getLast().**

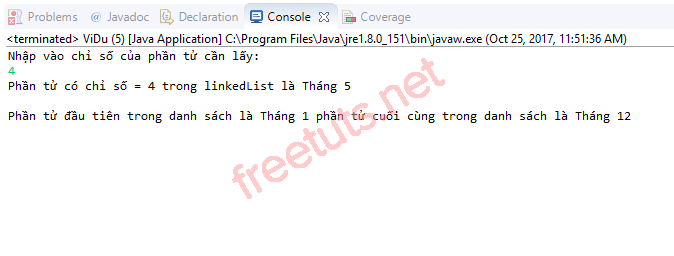
Công dụng: truy cập đến phần tử cuối cùng trong danh sách.

Sau đây là ví dụ minh họa 3 phương thức này.

**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  38  39  40  41  42  43  44  45  46  47  48 | public static void main(String[] args) {      Scanner scanner = new Scanner(System.in);        // khai báo 1 danh sách liên kết      // lưu trữ các tháng trong năm      LinkedList<String> linkedList = new LinkedList<>();      linkedList.add("Tháng 1");      linkedList.add("Tháng 2");      linkedList.add("Tháng 3");      linkedList.add("Tháng 4");      linkedList.add("Tháng 5");      linkedList.add("Tháng 6");      linkedList.add("Tháng 7");      linkedList.add("Tháng 8");      linkedList.add("Tháng 9");      linkedList.add("Tháng 10");      linkedList.add("Tháng 11");      linkedList.add("Tháng 12");        System.out.println("Nhập vào chỉ số của phần tử cần lấy: ");      int index = scanner.nextInt();        // kiểm tra nếu chỉ số lớn hơn hoặc bằng 0      // và nhỏ hơn kích thước của linkedList - 1 thì mới lấy      // ngược lại thông báo lỗi      if ((index < 0) || (index > (linkedList.size() - 1))) {          System.out.println("Chỉ số cần lấy phải lớn hơn 0 và nhỏ hơn "              + (linkedList.size()-1));      } else {          // truy cập phần tử có chỉ số index trong linkedList          // vì linkedList có kiểu là String          // nên các phần tử con của nó cũng có kiểu dữ liệu là String          String node = linkedList.get(index);          System.out.println("Phần tử có chỉ số = " + index + " trong linkedList là " + node);      }        // truy cập phần tử đầu tiên trong danh sách      // sử dụng phương thức getFirst()      String firstNode = linkedList.getFirst();        // truy cập phần tử đầu tiên trong danh sách      // sử dụng phương thức getLast()      String lastNode = linkedList.getLast();        System.out.println("\nPhần tử đầu tiên trong danh sách là " + firstNode +          " phần tử cuối cùng trong danh sách là " + lastNode);    } |

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 LinkedList, Java cung cấp cho chúng ta phương thức set().

Cú pháp:

**Cú pháp**

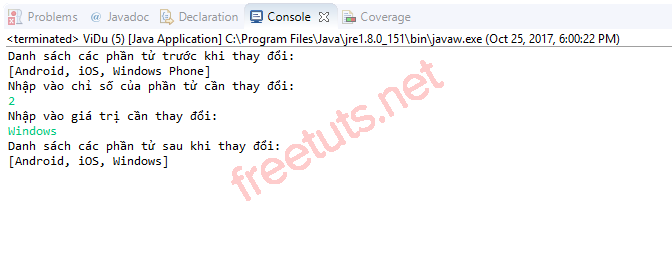
|  |  |
| --- | --- |
| 1 | set(int index, E element); |

, trong đó index là chỉ số của phần tử cần cập nhật và element là phần tử mới để thay thế.

**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 | public static void main(String[] args) {      Scanner scanner = new Scanner(System.in);        // khai báo 1 danh sách liên kết      // có kiểu là String      LinkedList<String> linkedList = new LinkedList<>();      linkedList.add("Android");      linkedList.add("iOS");      linkedList.add("Windows Phone");        System.out.println("Danh sách các phần tử trước khi thay đổi: ");      System.out.print(linkedList + "\t");        System.out.println("\nNhập vào chỉ số của phần tử cần thay đổi: ");      int index = Integer.parseInt(scanner.nextLine());   // hạn chế hiện tượng trôi lệnh      System.out.println("Nhập vào giá trị cần thay đổi: ");      String str = scanner.nextLine();        // kiểm tra nếu chỉ số lớn hơn hoặc bằng 0      // và nhỏ hơn kích thước của linkedList - 1 thì mới cập nhật      // ngược lại thông báo lỗi      if ((index < 0) || (index > (linkedList.size() - 1))) {          System.out.println("Chỉ số cần thay đổi phải lớn hơn 0 và nhỏ hơn "              + (linkedList.size()-1));      } else {          String node = linkedList.set(index, String.valueOf(str));      }        System.out.println("Danh sách các phần tử sau khi thay đổi: ");      System.out.print(linkedList + "\t");  } |

Kết quả sau khi biên dịch chương trình:



### **2.6. Xóa phần tử**

Để xóa phần tử trong LinkedList, Java cung cấp cho chúng ta 4 phương thức đó là remove(), removeFist(), removeLast() và removeAll(). Sau đây chúng ta sẽ cùng tìm hiểu về 4 phương thức này.

| **Tên phương thức** | **Công dụng** |
| --- | --- |
| remove() | Phương thức này được sử dụng trong 2 trường hợp: xóa dựa vào chỉ số của phần tử và xóa trực tiếp phần tử đó (không cần biết đến chỉ số của nó). |
| removeFirst() | Xóa phần tử đầu tiên trong danh sách. |
| removeLast() | Xóa phần tử cuối cùng trong danh sách. |
| removeAll() | Xóa những phần tử có trong danh sách này mà cũng tồn tại trong 1 Collection khác (List, Set,...) |

Ví dụ dưới đây sẽ tổng hợp cách sử dụng các phương thức này.

**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 | public static void main(String[] args) {      Scanner scanner = new Scanner(System.in);        // khai báo 1 danh sách liên kết      // có kiểu là String      LinkedList<String> linkedList = new LinkedList<>();      linkedList.add("Six");      linkedList.add("Five");      linkedList.add("Three");      linkedList.add("One");      linkedList.add("Six");        // tạo bản sao của linkedList sử dụng phương thức clone()      // lúc này linkedListCopy1 và linkedListCopy2 sẽ có các phần tử của linkedList      LinkedList<String> linkedListCopy1 = (LinkedList<String>) linkedList.clone();      LinkedList<String> linkedListCopy2 = (LinkedList<String>) linkedList.clone();        // tạo 1 List Interface      List<String> listString = new LinkedList<>();      listString.add("Two");      listString.add("Five");      listString.add("Four");        // xóa phần tử đầu tiên và cuối cùng trong linkedList      // sử dụng phương thức removeFirst() và removeLast()      linkedList.removeFirst();      linkedList.removeLast();      System.out.println("Các phần tử có trong linkedList sau khi xóa: " + linkedList);        // xóa phần tử có chỉ số 2 trong linkedListCopy1      // sử dụng phương thức remove()      // phần tử có chỉ số 2 là "Three"      linkedListCopy1.remove(2);        // xóa phần tử có giá trị "Six" trong linkedListCopy1      // trong danh sách này có 3 phần tử có giá trị là "Six"      // nên chương trình sẽ xóa phần tử "Six" đầu tiên      linkedListCopy1.remove("Six");      System.out.println("Các phần tử có trong linkedListCopy1 sau khi xóa: "          + linkedListCopy1);        // xóa những phần tử có trong linkedListCopy2 mà cũng có trong listString      // sử dụng phương thức remove()      linkedListCopy2.removeAll(listString);      System.out.println("Các phần tử có trong linkedListCopy2 sau khi xóa: "          + linkedListCopy2);    } |

Kết quả sau khi biên dịch chương trình:

