**SortedSet Interface trong Java**

**1. Đặc điểm**

SortedSet Interface là 1 dạng riêng của Set Interface nên nó có những đặc điểm của Set đó là các phần tử trong SortedSet là duy nhất (nghĩa là giá trị của các phần tử này không được giống nhau) và SortedSet được sử dụng khi chúng ta muốn lưu trữ một danh sách các phần tử không có sự trùng lặp. Ngoài ra,SortedSet**có điểm vượt trội hơn so với**Set**là thứ tự các phần tử trong**Set**được sắp xếp tăng dần hoặc giảm dần** (**mặc định là tăng dần**).

**2. Các phương thức phổ biến**

Vì SortedSet là 1 dạng riêng của Set nên những phương thức của nó sẽ tương tự như những phương thức có trong Set. Trong bài này, tôi sẽ không trình bày lại những phương thức đó mà tôi sẽ giới thiệu những phương thức riêng của SortedSet để cho các bạn tìm hiểu.

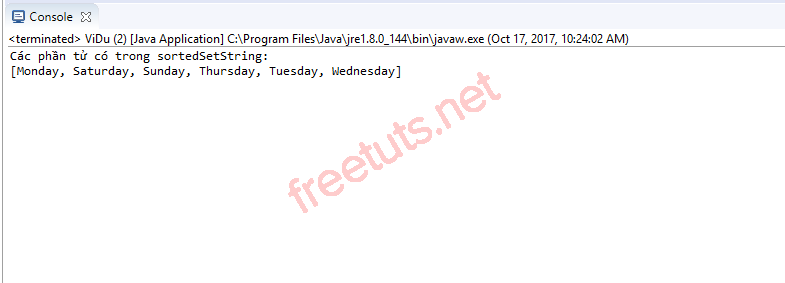
### **2.1. Tạo mới và hiển thị các phần tử của 1 SortedSet**

Để khai báo một SortedSet, chúng ta cần phải dùng đến Class để triển khai nó, trong phần này chúng ta sẽ sử dụng Class là TreeSet bởi vì các phần tử trong TreeSet được sắp xếp theo chiều tăng dần.

**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) {      // khai báo SortedSet Interface tên sortedSetString      // và sử dụng Class là TreeSet để triển khai      // TreeSet là 1 Class Collection      // các phần tử trong sortedSetString cũng có kiểu là String      SortedSet<String> sortedSetString = new TreeSet<String>();        // thêm các phần tử vào trong sortedSetString      sortedSetString.add("Monday");      sortedSetString.add("Tuesday");      sortedSetString.add("Wednesday");      sortedSetString.add("Thursday");      sortedSetString.add("Saturday");      sortedSetString.add("Sunday");        // hiển thị sortedSetString ở dạng mảng      // các phần tử được sắp xếp tăng dần theo chữ cái đầu tiên      System.out.println("Các phần tử có trong sortedSetString: ");      System.out.println(sortedSetString);  } |

Kết quả sau khi biên dịch chương trình:



**Lưu ý:** Để khai báo SortedSet chúng ta cần phải import gói thư viện java.util.SortedSet. Cú pháp import như sau:

**Cú pháp**

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

### **2.2. Trích xuất một phần trong SortedSet**

Đối với SortedSet, Java cung cấp cho chúng ta các phương thức để trích xuất các phần tử trong SortedSet đó là subset(), headset() và tailset(). Sau đây chúng ta sẽ cùng nhau tìm hiểu về 3 phương thức này.

**Phương thức subSet().**

**Cú pháp**

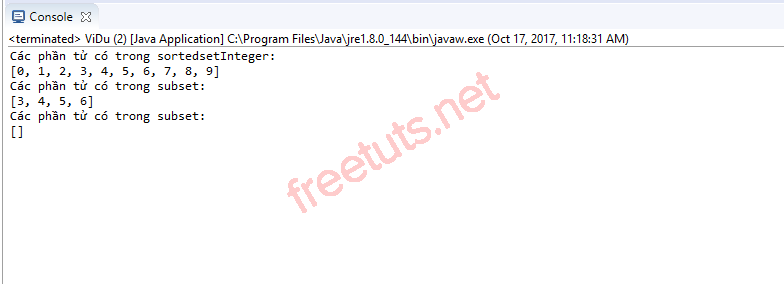
|  |  |
| --- | --- |
| 1 | SortedSet subSet(E fromElement, E toElement) |

Công dụng: Phương thức subSet() sẽ trả về một SortedSet được trích xuất từ phần tử fromElement đến phần tử đứng trước phần tử toElement của một SortedSet cho trước.

**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 | public static void main(String[] args) {      List<Integer> listInteger = new ArrayList<>();        // thêm các phần tử vào trong listInteger      listInteger.add(2);      listInteger.add(1);      listInteger.add(4);      listInteger.add(3);      listInteger.add(6);      listInteger.add(5);      listInteger.add(8);      listInteger.add(7);      listInteger.add(0);      listInteger.add(9);        // khai báo 1 SortedSet Interface có kiểu là Integer      // có các phần tử là các phần tử của listInteger      SortedSet<Integer> sortedsetInteger = new TreeSet<>(listInteger);        System.out.println("Các phần tử có trong sortedsetInteger: ");      System.out.println(sortedsetInteger);        // khai báo 1 SortedSet có tên là subset      // có các phần tử được trích xuất      // trong đoạn [3,7) của sortedsetInteger      SortedSet<Integer> subset = sortedsetInteger.subSet(3, 7);      System.out.println("Các phần tử có trong subset: ");      System.out.println(subset);        // nếu phần tử đầu và phần tử cuối bằng nhau      // thì kết quả của phương thức subSet()      // sẽ trả về subset không có phần tử nào      subset = sortedsetInteger.subSet(3, 3);      System.out.println("Các phần tử có trong subset: ");      System.out.println(subset);  } |

Kết quả sau khi biên dịch chương trình:



**Phương thức headSet().**

**Cú pháp**

|  |  |
| --- | --- |
| 1 | SortedSet headSet(E toElement) |

Công dụng: Phương thức headSet() sẽ trả về một SortedSet được trích xuất từ phần tử đầu tiên đến phần tử đứng trước phần tử toElement của một SortedSet cho trước.

**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) {      List<Integer> listInteger = new ArrayList<>();        // thêm các phần tử vào trong listInteger      listInteger.add(2);      listInteger.add(1);      listInteger.add(4);      listInteger.add(3);      listInteger.add(6);      listInteger.add(5);      listInteger.add(8);      listInteger.add(7);      listInteger.add(0);      listInteger.add(9);        // khai báo 1 SortedSet Interface có kiểu là Integer      // có các phần tử là các phần tử của listInteger      SortedSet<Integer> sortedsetInteger = new TreeSet<>(listInteger);        System.out.println("Các phần tử có trong sortedsetInteger: ");      System.out.println(sortedsetInteger);        // khai báo 1 SortedSet có tên là headset      // có các phần tử được trích xuất      // từ phần tử đầu tiên đến      // phần tử đứng trước phần tử 5 trong sortedsetInteger      SortedSet<Integer> headset = sortedsetInteger.headSet(5);      System.out.println("Các phần tử có trong headset: ");      System.out.println(headset);  } |

Kết quả sau khi biên dịch chương trình:



**Phương thức tailSet().**

**Cú pháp**

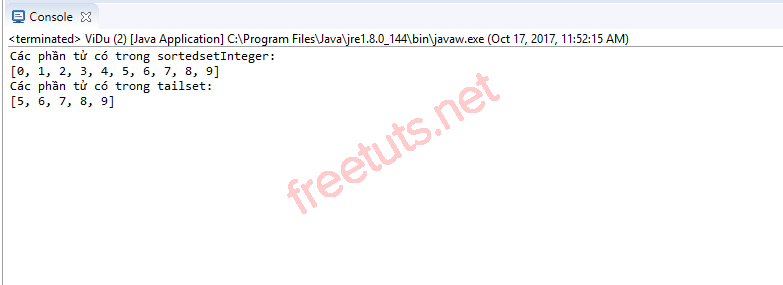
|  |  |
| --- | --- |
| 1 | SortedSet tailSet(E fromElement) |

Công dụng: Phương thức tailSet() sẽ trả về một SortedSet được trích xuất từ phần tử lớn hơn hoặc bằng phần tử fromElement đến phần tử cuối cùng của một SortedSet cho trước.

**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) {      List<Integer> listInteger = new ArrayList<>();        // thêm các phần tử vào trong listInteger      listInteger.add(2);      listInteger.add(1);      listInteger.add(4);      listInteger.add(3);      listInteger.add(6);      listInteger.add(5);      listInteger.add(8);      listInteger.add(7);      listInteger.add(0);      listInteger.add(9);        // khai báo 1 SortedSet Interface có kiểu là Integer      // có các phần tử là các phần tử của listInteger      SortedSet<Integer> sortedsetInteger = new TreeSet<>(listInteger);        System.out.println("Các phần tử có trong sortedsetInteger: ");      System.out.println(sortedsetInteger);        // khai báo 1 SortedSet có tên là tailset      // có các phần tử được trích xuất      // từ phần tử lớn hơn hoặc bằng      // phần tử fromElement đến phần tử cuối cùng của sortedsetInteger      SortedSet<Integer> tailset = sortedsetInteger.tailSet(5);      System.out.println("Các phần tử có trong tailset: ");      System.out.println(tailset);  } |

Kết quả sau khi biên dịch chương trình:



### **2.3. Tìm phần tử nhỏ nhất và lớn nhất trong SortedSet**

Để tìm phần tử nhỏ nhất và lớn nhất trong SortedSet, Java cung cấp cho chúng ta 2 phương thức đó là first() (tìm phần tử nhỏ nhất) và last() (tìm phần tử lớn nhất).

**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) {      List<Integer> listInteger = new ArrayList<>();        // thêm các phần tử vào trong listInteger      listInteger.add(2);      listInteger.add(1);      listInteger.add(4);      listInteger.add(3);      listInteger.add(6);      listInteger.add(5);      listInteger.add(8);      listInteger.add(7);      listInteger.add(0);      listInteger.add(9);        SortedSet<Integer> sortedsetInteger = new TreeSet<>(listInteger);        System.out.println("Các phần tử có trong sortedsetInteger: ");      System.out.println(sortedsetInteger);        // tìm phần tử lớn nhất và nhỏ nhất trong sortedsetInteger      int phanTuLonNhat = sortedsetInteger.last();      int phanTuNhoNhat = sortedsetInteger.first();      System.out.println("Phần tử lớn nhất và nhỏ nhất trong"          + " sortedsetInteger là " + phanTuLonNhat + " và " + phanTuNhoNhat);  } |

Kết quả sau khi biên dịch chương trình:

