**TreeSet trong Java**

**1. Đặc điểm**

TreeSet là Class triển khai phổ biến nhất của SortedSet nên nó sẽ có một vài đặc điểm và phương thức tương đồng với SortedSet. 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), thứ tự các phần tử trong TreeSet mặc định được sắp xếp tăng dần và giá trị của các phần tử này là duy nhất.

**2. Các phương thức phổ biến**

### **2.1. Tạo mới một TreeSet**

Để khai báo một TreeSet, chúng ta cần phải import gói thư viện java.util.TreeSet của Java. Cú pháp import như sau:

**Cú pháp**

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

Sau đây là ví dụ cách tạo mới một TreeSet trong Java:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public static void main(String[] args) {      // khai báo 1 TreeSet có tên là treeSetInteger      // có kiểu là Integer      TreeSet<Integer> treeSetInteger = new TreeSet<>();        // khai báo 1 TreeSet có tên là treeSetCharacter      // được tạo thành từ 1 Class Collection khác      TreeSet<Character> treeSetCharacter = new TreeSet<>(new HashSet<>());  } |

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

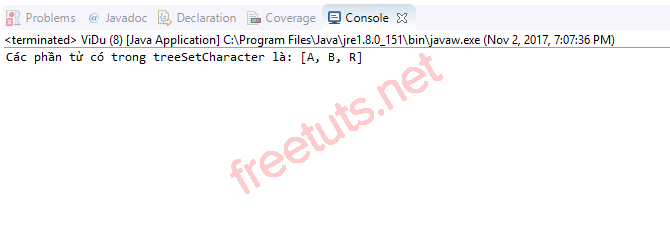
Để hiển thị các phần tử có trong TreeSet, chúng ta có các cách như sau:

**Hiển thị theo tên TreeSet.**

**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 1 TreeSet có tên là treeSetCharacter      // được tạo thành từ 1 Class Collection khác      TreeSet<Character> treeSetCharacter = new TreeSet<>(new HashSet<>());        // Thêm các phần tử vào trong treeSetCharacter      // sử dụng phương thức add()      treeSetCharacter.add('R');      treeSetCharacter.add('A');      treeSetCharacter.add('B');      treeSetCharacter.add('R');        // hiển thị các phần tử có trong treeSetCharacter      // trong treeSetCharacter có 2 phần tử là 'R'      // mà các phần tử trong treeSetCharacter là không trùng nhau      // nên sẽ chỉ có 1 phần tử 'R' được hiển thị      // các phần tử trong treeSetCharacter      // mặc định được sắp xếp theo thứ tự tăng dần.      System.out.println("Các phần tử có trong treeSetCharacter là: " + treeSetCharacter);  } |

Kết quả sau khi biên dịch chương trình:

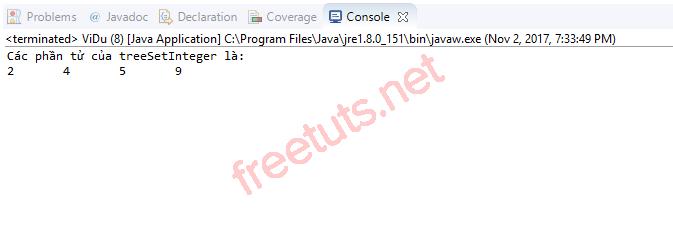


**Sử dụng vòng lặp**for**cải tiến duyệt theo đối tượng trong TreeSet.**

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public static void main(String[] args) {      TreeSet<Integer> treeSetInteger = new TreeSet<>();        treeSetInteger.add(5);      treeSetInteger.add(9);      treeSetInteger.add(4);      treeSetInteger.add(2);        // hiển thị các phần tử có trong treeSetInteger      // bằng cách sử dụng vòng lặp for duyệt theo đối tượng      // trong đó kiểu dữ liệu của biến number      // phải trùng với kiểu dữ liệu của treeSetInteger      System.out.println("Các phần tử của treeSetInteger là: ");      for (int number : treeSetInteger) {          System.out.print(number + "\t");      }  } |

Kết quả sau khi biên dịch chương trình:



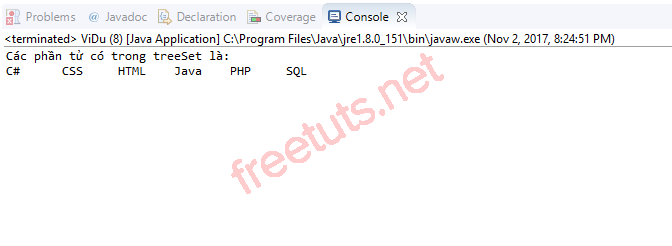
**Sử dụng Iterator.**

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 TreeSet có kiểu là String      TreeSet<String> treeSet = new TreeSet<>();        // thêm vào phần tử cho treeSet      treeSet.add("C#");      treeSet.add("Java");      treeSet.add("PHP");      treeSet.add("SQL");      treeSet.add("HTML");      treeSet.add("CSS");        // khai báo một Iterator có kiểu là String      Iterator<String> iterator = treeSet.iterator();        // hiển thị các phần tử có trong treeSet      // bằng cách sử dụng Iterator      System.out.println("Các phần tử có trong treeSet là: ");      while (iterator.hasNext()) {          System.out.print(iterator.next() + "\t");      }  } |

Kết quả sau khi biên dịch chương trình:



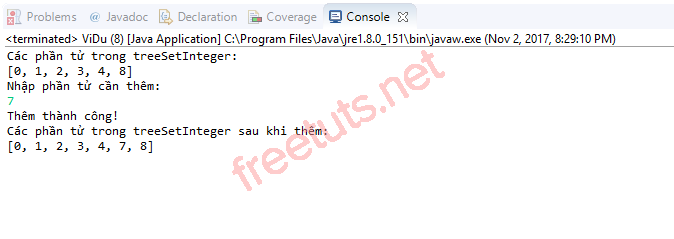
### **2.3. Thêm phần tử vào trong TreeSet**

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

**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 | public static void main(String[] args) {      int number;      TreeSet<Integer> treeSetInteger = new TreeSet<>();      Scanner scanner = new Scanner(System.in);        // thêm các phần tử vào treeSetInteger      treeSetInteger.add(0);      treeSetInteger.add(3);      treeSetInteger.add(1);      treeSetInteger.add(4);      treeSetInteger.add(2);      treeSetInteger.add(8);        // hiển thị các phần tử trong treeSetInteger      System.out.println("Các phần tử trong treeSetInteger: ");      System.out.println(treeSetInteger);        System.out.println("Nhập phần tử cần thêm: ");      number = scanner.nextInt();        // thêm một phần tử mới vào treeSetInteger từ bàn phím      // nếu phần tử đó đã tồn tại thì thông báo đã tồn tại      // ngược lại thì thêm vào      if (!treeSetInteger.contains(number)) {          treeSetInteger.add(number);          System.out.println("Thêm thành công!");          System.out.println("Các phần tử trong treeSetInteger sau khi thêm: ");          System.out.println(treeSetInteger);      } else {          System.out.println("Phần tử " + number + " đã tồn tại!");      }  } |

Kết quả sau khi biên dịch chương trình:



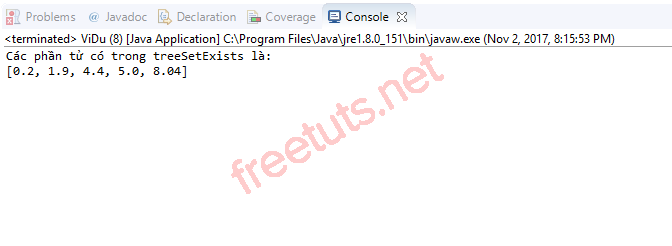
**Thêm phần tử sử dụng phương thức addAll().**

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

**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 2 TreeSet có kiểu là Float      TreeSet<Float> treeSet = new TreeSet<>();      TreeSet<Float> treeSetExists = new TreeSet<>();        // thêm vào phần tử cho treeSet và treeSetExists      treeSet.add(5f);      treeSet.add(8.04f);      treeSet.add(0.2f);        treeSetExists.add(4.4f);      treeSetExists.add(1.9f);        // thêm các phần tử của treeSet      // vào treeSetExists      treeSetExists.addAll(treeSet);        // hiển thị treeSetExists sau khi thêm      // các phần tử được sắp xếp tăng dần      System.out.println("Các phần tử có trong treeSetExists là: ");      System.out.print(treeSetExists + "\t");  } |

Kết quả sau khi biên dịch chương trình:



### **2.4. Xóa phần tử**

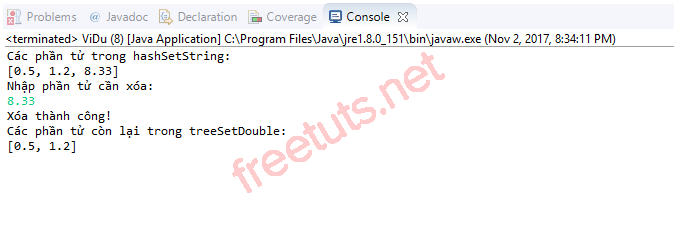
Để xóa một phần tử khỏi TreeSet, chúng ta sẽ sử dụng phương thức remove(). Ví dụ dưới đây sẽ sử dụng phương thức remove() để xóa một phần tử bất kỳ trong treeSetDouble:

**Ví dụ**

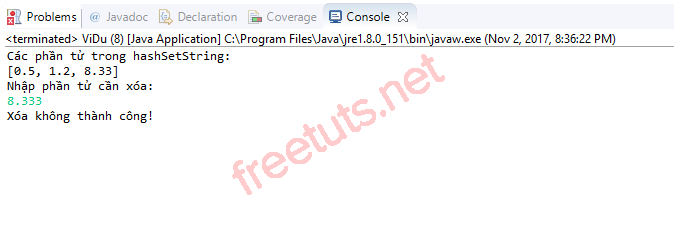
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | public static void main(String[] args) {      double number;      TreeSet<Double> treeSetDouble = new TreeSet<>();      Scanner scanner = new Scanner(System.in);        // thêm các phần tử vào treeSetDouble      treeSetDouble.add(0.5d);      treeSetDouble.add(8.33d);      treeSetDouble.add(1.2d);        // hiển thị các phần tử trong treeSetDouble      System.out.println("Các phần tử trong treeSetDouble: ");      System.out.println(treeSetDouble);        System.out.println("Nhập phần tử cần xóa: ");      number = scanner.nextDouble();        // nếu phần tử cần xóa      // có tồn tại treeSetDouble thì sẽ thông báo xóa thành công      // và hiển thị các phần tử còn lại      // ngược lại thông báo xóa không thành công      if (treeSetDouble.contains(number)) {          treeSetDouble.remove(number);          System.out.println("Xóa thành công!");          System.out.println("Các phần tử còn lại trong treeSetDouble:");          System.out.println(treeSetDouble);      } else {          System.out.println("Xóa không thành công!");      }  } |

Kết quả sau khi biên dịch chương trình:

**Trường hợp 1:** Xóa phần tử thành công:



**Trường hợp 2:** Xóa phần tử thất bại:

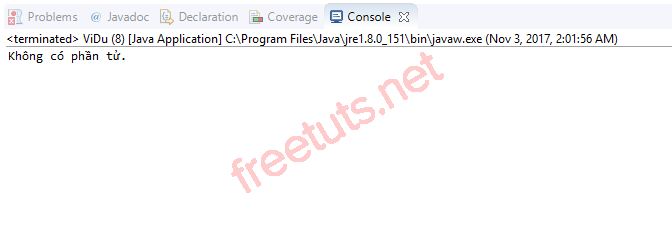


Để xóa tất cả các phần tử có trong TreeSet, chúng ta sẽ sử dụng phương thức clear() có sẵn của Java. Ví dụ:

**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 | public static void main(String[] args) {      int Number;      TreeSet<Integer> treeSetInteger = new TreeSet<>();      Scanner scanner = new Scanner(System.in);        // thêm các phần tử vào treeSetInteger      treeSetInteger.add(7);      treeSetInteger.add(19);      treeSetInteger.add(1);      treeSetInteger.add(20);        // xóa toàn bộ các phần tử trong treeSetInteger      // sử dụng phương thức clear()      treeSetInteger.clear();        // sau khi xóa thì trong treeSetInteger      // sẽ không có phần tử nào      // phương thức isEmpty() dưới đây sẽ kiểm tra      // nếu treeSetInteger không có giá trị      // thì sẽ hiển thị thông báo "Không có phần tử"      if (treeSetInteger.isEmpty()) {          System.out.println("Không có phần tử.");      }  } |

Kết quả sau khi biên dịch chương trình:



### **2.5. Trích xuất một phần trong TreeSet**

Đối với TreeSet, Java cung cấp cho chúng ta các phương thức để trích xuất các phần tử trong TreeSet 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**

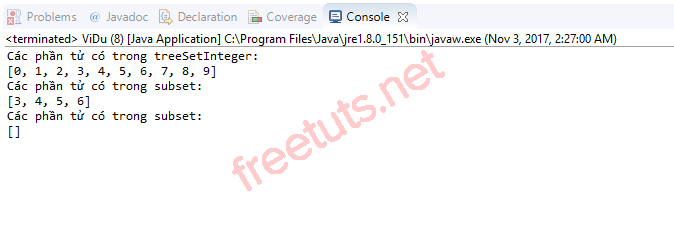
|  |  |
| --- | --- |
| 1 | SortedSet subSet(E fromElement, E toElement) |

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 TreeSet 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 | public static void main(String[] args) {      TreeSet<Integer> treeSetInteger = new TreeSet<>();        // thêm các phần tử vào trong treeSetInteger      treeSetInteger.add(2);      treeSetInteger.add(1);      treeSetInteger.add(4);      treeSetInteger.add(3);      treeSetInteger.add(6);      treeSetInteger.add(5);      treeSetInteger.add(8);      treeSetInteger.add(7);      treeSetInteger.add(0);      treeSetInteger.add(9);        System.out.println("Các phần tử có trong treeSetInteger: ");      System.out.println(treeSetInteger);        // khai báo 1 TreeSet có tên là subset      // có các phần tử được trích xuất      // trong đoạn [3,10) của treeSetInteger      SortedSet<Integer> subset = treeSetInteger.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 = treeSetInteger.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**

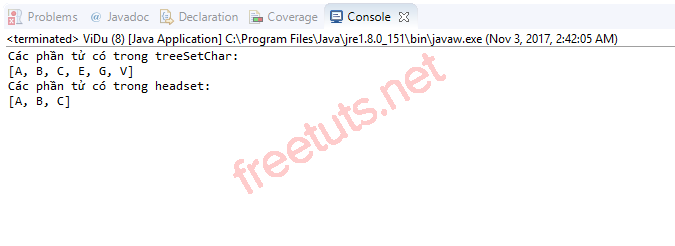
|  |  |
| --- | --- |
| 1 | SortedSet headSet(E toElement) |

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 TreeSet 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 | public static void main(String[] args) {      TreeSet<Character> treeSetChar = new TreeSet<>();        // thêm các phần tử vào trong treeSetChar      treeSetChar.add('V');      treeSetChar.add('A');      treeSetChar.add('B');      treeSetChar.add('C');      treeSetChar.add('G');      treeSetChar.add('E');        System.out.println("Các phần tử có trong treeSetChar: ");      System.out.println(treeSetChar);        // 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 'E' trong treeSetChar      SortedSet<Character> headset = treeSetChar.headSet('E');      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**

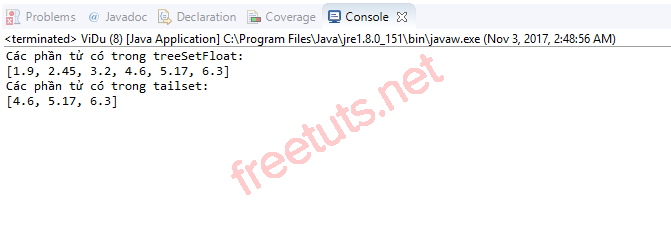
|  |  |
| --- | --- |
| 1 | SortedSet tailSet(E fromElement) |

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 TreeSet 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 | public static void main(String[] args) {      TreeSet<Float> treeSetFloat = new TreeSet<>();        // thêm các phần tử vào trong TreeSet      treeSetFloat.add(2.45f);      treeSetFloat.add(1.9f);      treeSetFloat.add(4.6f);      treeSetFloat.add(3.20f);      treeSetFloat.add(6.3f);      treeSetFloat.add(5.17f);        System.out.println("Các phần tử có trong treeSetFloat: ");      System.out.println(treeSetFloat);        // khai báo 1 SortedSet có tên là tailset      // có các phần tử được trích xuất      // từ phần tử lớn hơn hoặc bằng      // phần tử 4.6f đến phần tử cuối cùng của treeSetFloat      SortedSet<Float> tailset = treeSetFloat.tailSet(4.6f);      System.out.println("Các phần tử có trong tailset: ");      System.out.println(tailset);  } |

Kết quả sau khi biên dịch chương trình:



### **2.6. 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 | public static void main(String[] args) {      TreeSet<String> treeSetString = new TreeSet<>();        // thêm các phần tử vào trong treeSetString      treeSetString.add("JAVA");      treeSetString.add("PHP");      treeSetString.add("Spring");      treeSetString.add("Struts");      treeSetString.add("Python");      treeSetString.add("HTML");        System.out.println("Các phần tử có trong treeSetString: ");      System.out.println(treeSetString);        // tìm phần tử lớn nhất (phần tử cuối cùng)      // và phần tử nhỏ nhất (phần tử đầu tiên) trong treeSetString      String phanTuLonNhat = treeSetString.last();      String phanTuNhoNhat = treeSetString.first();      System.out.println("Phần tử lớn nhất và nhỏ nhất trong"          + " treeSetString là " + phanTuLonNhat + " và " + phanTuNhoNhat);  } |

Kết quả sau khi biên dịch chương trình:

