**Set Interface trong Java**

**1. Đặc điểm**

Set Interface là một loại Interface Collection. Khác với List, các phần tử trong List có thể giống nhau, còn đối với Set, các phần tử trong Set là duy nhất (nghĩa là giá trị của các phần tử này không được giống nhau).

Vậy Set được sử dụng trong trường hợp nào? Chúng ta sẽ sử dụng Set khi chúng ta muốn lưu trữ một danh sách các phần tử không có sự trùng lặp hoặc khi chúng ta không quan tâm đến thứ tự của các phần tử trong danh sách đó.

**2. Các phương thức phổ biến**

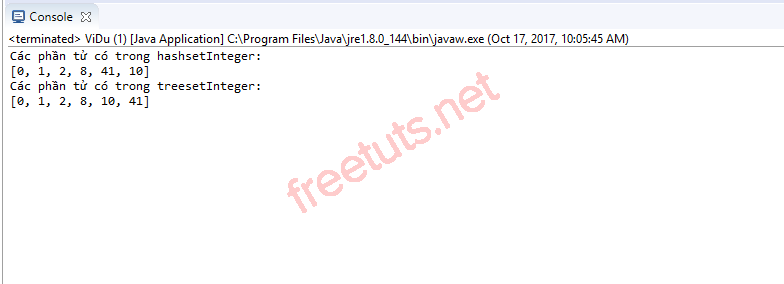
### **2.1. Tạo mới một Set Interface**

Trong bài [Tổng quan](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), 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 Set 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 HashSet và TreeSet. Đối với Set Interface có Class triển khai là HashSet thì các phần tử không được sắp xếp theo bất kỳ thứ tự nào, còn đối với Set Interface có Class triển khai là TreeSet thì thứ tự các phần tử trong Set được sắp xếp tăng dần. Ví dụ dưới đây sẽ cho các bạn thấy sự khác nhau khi sử dụng HashSet và TreeSet để khai báo Set trong Java:

**Cú pháp**

|  |  |
| --- | --- |
| 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) {      // khai báo Set Interface tên hashsetInteger      // và sử dụng Class là HashSet để triển khai      // HashSet là 1 Class Collection      // các phần tử trong hashsetInteger cũng có kiểu là Integer      Set<Integer> hashsetInteger = new HashSet<>();      hashsetInteger.add(41);      hashsetInteger.add(1);      hashsetInteger.add(0);      hashsetInteger.add(8);      hashsetInteger.add(1);      hashsetInteger.add(2);      hashsetInteger.add(10);        // khai báo Set Interface tên treesetInteger      // và sử dụng Class là TreeSet để triển khai      // TreeSet là 1 Class Collection      // các phần tử trong treesetInteger cũng có kiểu là Integer      Set<Integer> treesetInteger = new TreeSet<>();      treesetInteger.add(41);      treesetInteger.add(1);      treesetInteger.add(0);      treesetInteger.add(8);      treesetInteger.add(1);      treesetInteger.add(2);      treesetInteger.add(10);        System.out.println("Các phần tử có trong hashsetInteger: ");      System.out.println(hashsetInteger);      System.out.println("Các phần tử có trong treesetInteger: ");      System.out.println(treesetInteger);  } |

Kết quả sau khi biên dịch chương trình:

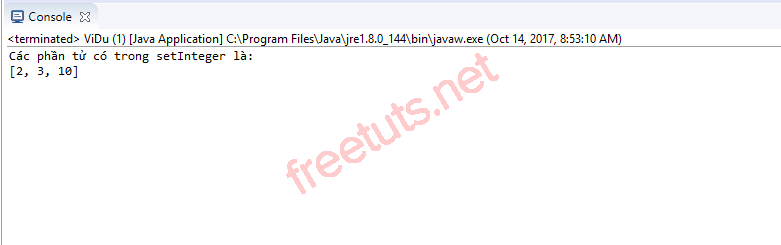


Chúng ta có thể tạo mới một Set thông qua một Collection đã tồn tại. Sau đây tôi sẽ đưa ra một ví dụ tạo mới 1 Set từ 1 List đã tồn tại:

**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) {      List<Integer> listInteger = new ArrayList<>();      listInteger.add(3);      listInteger.add(10);      listInteger.add(2);      listInteger.add(10);        // khai báo 1 Set Interface có kiểu là Integer      // có các phần tử là các phần tử của listInteger      Set<Integer> setInteger = new TreeSet<>(listInteger);        // hiển thị setInteger ở dạng mảng      // trong listInteger có 2 phần tử 10      // nhưng vì các phần tử trong Set không được giống nhau      // nên chỉ hiển thị 1 phần tử 10      System.out.println(setInteger);  } |

Kết quả sau khi biên dịch chương trình:

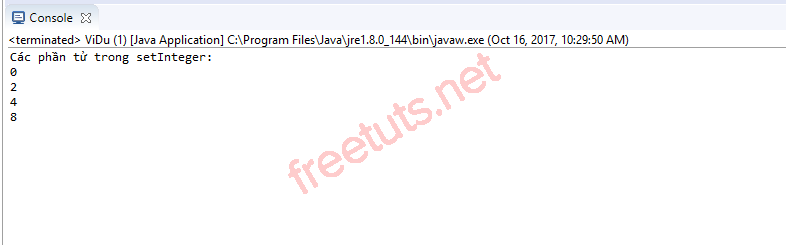


Ngoài ra, chúng ta có thể khởi tạo các phần tử cho Set bằng cách lọc các phần tử trong một Collection đã tồn tại theo một điều kiện cho trước. Ví dụ dưới đây sẽ tạo một Set có tên là setInteger, trong đó các phần tử của Set này bao gồm các phần tử là số chẵn trong listInteger đã tồn tại:

**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) {      List<Integer> listInteger = new ArrayList<>();      Set<Integer> setInteger = new TreeSet<>();        // thêm các phần tử vào listInteger      listInteger.add(0);      listInteger.add(3);      listInteger.add(1);      listInteger.add(4);      listInteger.add(2);      listInteger.add(8);        // lọc các phần tử là số chẵn trong listInteger      // và thêm vào trong setInteger      setInteger = listInteger.stream().filter(number -> number % 2 == 0)          .collect(Collectors.toSet());        // hiển thị các phần tử trong setInteger      System.out.println("Các phần tử trong setInteger: ");      for (int numbers : setInteger) {          System.out.println(numbers);      }  } |

Kết quả sau khi biên dịch chương trình:



Thông thường, đối với HashSet thì khả năng lưu trữ các phần tử mặc định là 16 phần tử. Vì vậy, trong trường hợp các bạn cần lưu trữ một Set có Class triển khai là HashSet và có nhiều hơn 16 phần tử thì các bạn nên khai báo số phần tử khi khởi tạo Set đó. Ví dụ dưới đây sẽ khai báo một Set có tên là setFloat, kiểu là Float và có 1000 phần tử:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4 | public static void main(String[] args) {      Set<Float> setFloat = new HashSet<>(1000);  } |

**Lưu ý:** Để khai báo Set chúng ta cần phải import gói thư viện java.util.Set, đối với HashSet thì import gói thư viện java.util.HashSet và với TreeSet thì import gói thư viện java.util.TreeSet. 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 Set  // thì import gói thư viện java.util.Set  import java.util.Set;  public class TênClass {      // ...  }    // Khai báo HashSet  // thì import gói thư viện java.util.HashSet  import java.util.HashSet;  public class TênClass {      // ...  }    // Khai báo TreeSet  // thì import gói thư viện java.util.TreeSet  import java.util.TreeSet;  public class TênClass {      // ...  } |

### **2.2. Hiển thị các phần tử có [trong](https://freetuts.net/set-interface-trong-java-1080.html) Set**

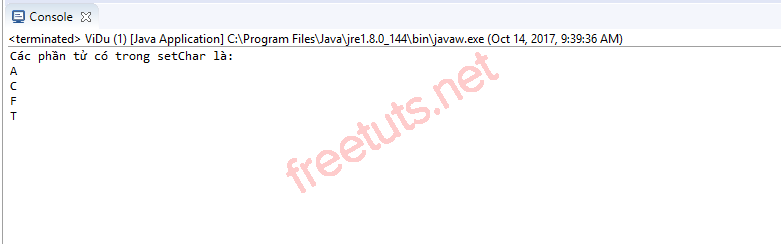
Để hiển thị các phần tử có trong Set, chúng ta có các cách như sau:

**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 | public static void main(String[] args) {      // khai báo List Interface có tên là setChar      // kiểu dữ liệu là Character (Wrapper class)      Set<Character> setChar = new TreeSet<Character>();        // thêm các phần tử      setChar.add('A');      setChar.add('C');      setChar.add('T');      setChar.add('F');        // hiển thị các phần tử có trong setChar      // bằng cách sử dụng vòng lặp for duyệt theo đối tượng      // trong đó kiểu dữ liệu của biến ch      // phải trùng với kiểu dữ liệu của setChar      System.out.println("Các phần tử có trong setChar là: ");      for (char ch : setChar) {          System.out.println(ch);      }  } |

Kết quả sau khi biên dịch chương trình:



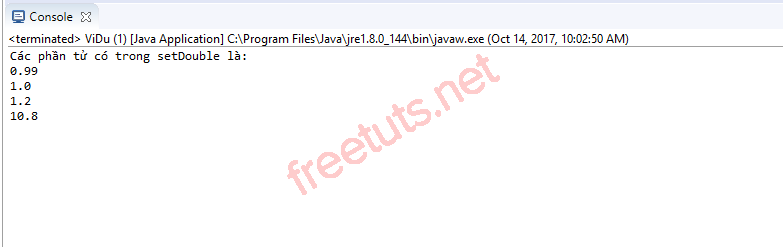
**Sử dụng Iterator.**

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  23  24  25  26  27  28  29  30 | public static void main(String[] args) {      // khai báo Set Interface có tên là setDouble      // kiểu dữ liệu là Double      Set<Double> setDouble = new TreeSet<Double>();        // khai báo một Iterator      Iterator<Double> iterator = null;        // thêm các phần tử      setDouble.add(10.8d);      setDouble.add(1.2d);      setDouble.add(1d);      setDouble.add(0.99d);        System.out.println("Các phần tử có trong setDouble là: ");        // Lấy ra đối tượng iterator để truy cập vào các phần tử của tập hợp.      // Đối tượng iterator này chỉ chứa các số Double.      // Lúc này iterator sẽ trỏ vào      // chỉ số trước chỉ số của phần tử đầu tiên trong setDouble      iterator = setDouble.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:



### **2.3. Thêm phần tử vào trong Set Interface**

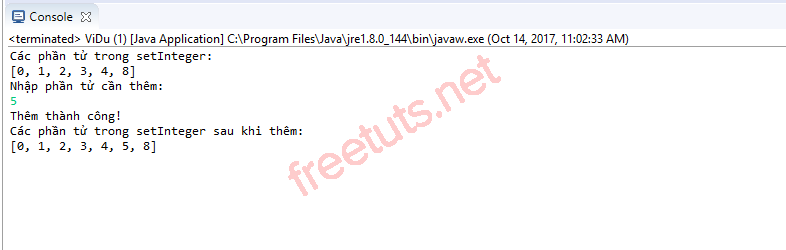
**Thêm phần tử sử dụng phương thức add().**

**Ví dụ**

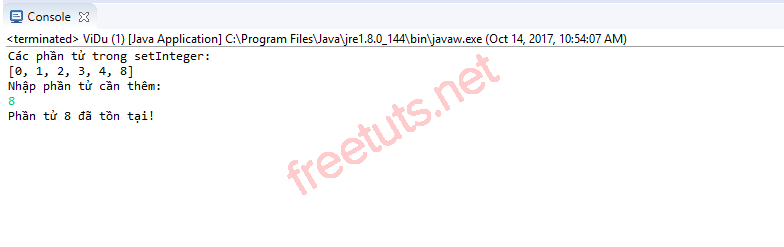
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | public static void main(String[] args) {      int number;      Set<Integer> setInteger = new TreeSet<>();      Scanner scanner = new Scanner(System.in);        // thêm các phần tử vào setInteger      setInteger.add(0);      setInteger.add(3);      setInteger.add(1);      setInteger.add(4);      setInteger.add(2);      setInteger.add(8);        // hiển thị các phần tử trong setInteger      System.out.println("Các phần tử trong setInteger: ");      System.out.println(setInteger);        System.out.println("Nhập phần tử cần thêm: ");      number = scanner.nextInt();        // thêm một phần tử mới vào setInteger từ bàn phím      // nếu phần tử đó đã tồn tại thì thông báo đã tồn tại      // ngược lại thì thêm vào      if (!setInteger.contains(number)) {          setInteger.add(number);          System.out.println("Thêm thành công!");          System.out.println("Các phần tử trong setInteger sau khi thêm: ");          System.out.println(setInteger);      } else {          System.out.println("Phần tử " + number + " đã tồn tại!");      }  } |

Kết quả sau khi biên dịch chương trình:

**Trường hợp 1:** Phần tử thêm vào chưa có trong setInteger:



**Trường hợp 2:** Phần tử thêm vào đã tồn tại trong setInteger:



### **2.4. Xóa phần tử**

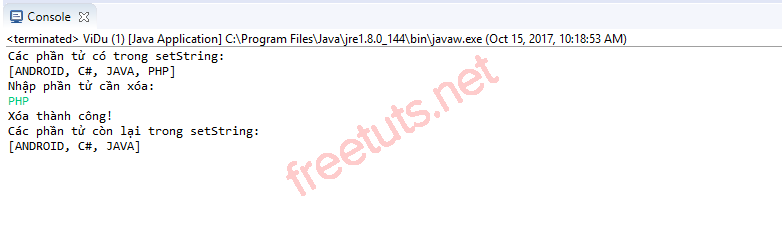
Để xóa một phần tử khỏi Set, chúng ta sẽ sử dụng phương thức remove(). Ví dụ dưới đây sẽ sử dụng phương thức remove() để xóa một phần tử bất kỳ trong setString:

**Ví dụ**

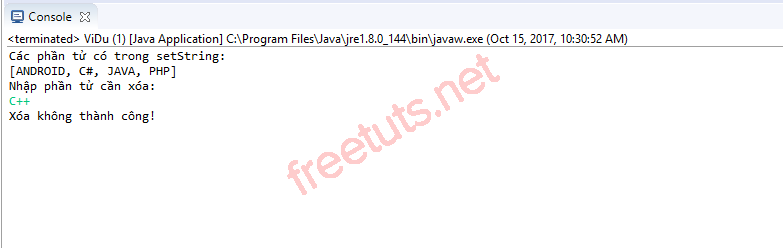
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | public static void main(String[] args) {      String str;      Set<String> setString = new TreeSet<>();      Scanner scanner = new Scanner(System.in);        // thêm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

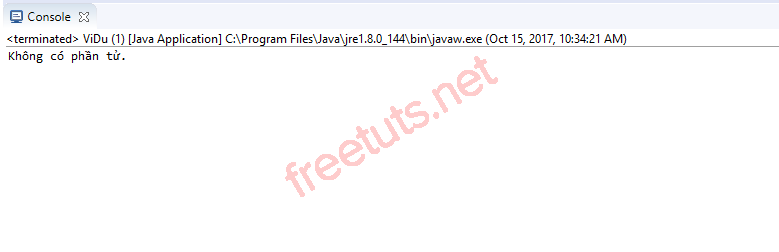


Để xóa tất cả các phần tử có trong Set, chúng ta sẽ sử dụng phương thức clear() có sẵn của Java. Ví dụ:

**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 | public static void main(String[] args) {      String str;      Set<String> setString = new TreeSet<>();      Scanner scanner = new Scanner(System.in);        // thêm các phần tử vào setInteger      setString.add("JAVA");      setString.add("ANDROID");      setString.add("PHP");      setString.add("C#");        // xóa toàn bộ các phần tử trong setString      // sử dụng phương thức clear()      setString.clear();        // sau khi xóa thì trong setString      // sẽ không có phần tử nào      // phương thức isEmpty() dưới đây sẽ kiểm tra      // nếu setString không có giá trị      // thì sẽ hiển thị thông báo "Không có phần tử"      if (setString.isEmpty()) {          System.out.println("Không có phần tử.");      }  } |

Kết quả sau khi biên dịch chương trình:



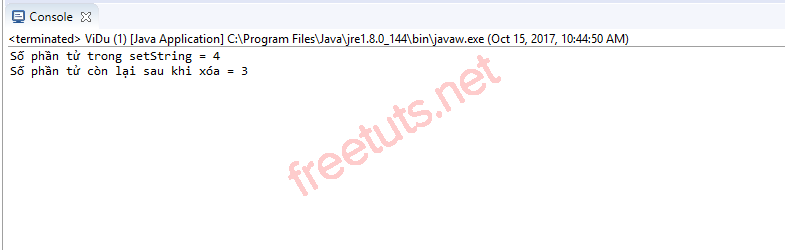
### **2.5. Đếm số phần tử có trong Set**

Ví dụ dưới đây sẽ sử dụng phương thức size() để đếm số phần tử có trong setString:

**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) {      String str;      Set<String> setString = new TreeSet<>();      Scanner scanner = new Scanner(System.in);        // thêm các phần tử vào setInteger      setString.add("JAVA");      setString.add("ANDROID");      setString.add("PHP");      setString.add("C#");        // hiển thị số phần tử có trong setString      System.out.print("Số phần tử trong setString = " + setString.size());        // xóa 1 phần tử trong setString      setString.remove("PHP");        // hiển thị số phần tử còn lại sau khi xóa      System.out.print("\nSố phần tử còn lại sau khi xóa = " + setString.size());  } |

Kết quả sau khi biên dịch chương trình:



### **2.6. Các toán tử giữa 2 Set trong Java**

Giả sử chúng ta có 2 Set có tên là setInteger1 và setInteger2 có kiểu là Integer. Giữa 2 Set này chúng ta có các toán tử tương tác như sau:

**Toán tử tập hợp con.**

Java cung cấp cho chúng ta phương thức containsAll() để xác định 1 Set có phải là tập hợp con của 1 Set khác hay không.

**Cú pháp**

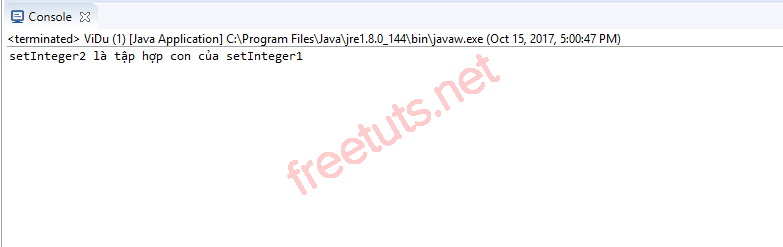
|  |  |
| --- | --- |
| 1 | set1.containsAll(set2); |

Phương thức này sẽ trả về true nếu set2 là tập hợp con của set1, ngược lại sẽ trả về false.

**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 | public static void main(String[] args) {      Integer[] arraySet1 = {2, 10, 4, 8, 5};      Integer[] arraySet2 = {10, 5};        // chuyển mảng arraySet1 và arraySet2      // sang 1 danh sách có cùng kiểu dữ liệu      // sử dụng phương thức Arrays.asList()      List<Integer> list1 = Arrays.asList(arraySet1);      List<Integer> list2 = Arrays.asList(arraySet2);        // chuyển List thành Set      Set<Integer> setInteger1 = new HashSet<>(list1);      Set<Integer> setInteger2 = new HashSet<>(list2);        // nếu kết quả của biểu thức kiểm tra trong if      // trả về kết quả đúng      // thì thực hiện lệnh bên trong if      // ngược lại thì thực hiện lệnh bên trong else      if (setInteger1.containsAll(setInteger2)) {          System.out.println("setInteger2 là tập hợp con của setInteger1");      } else {          System.out.println("setInteger2 không là tập hợp con của setInteger1");      }  } |

Kết quả sau khi biên dịch chương trình:



**Toán tử hợp**

**Cú pháp**

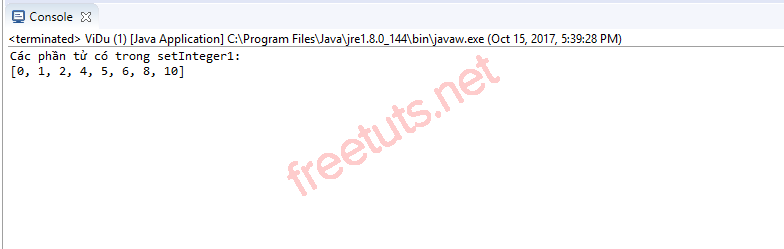
|  |  |
| --- | --- |
| 1 | set1.addAll(set2); |

Phương thức addAll() sẽ thực hiện phép toán hợp giữa set1 và set2 (tức là lấy các phần tử có trong set2 thêm vào trong set1).

**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) {      Integer[] arraySet1 = {2, 10, 4, 8, 5};      Integer[] arraySet2 = {1, 6, 0};        // chuyển mảng arraySet1 và arraySet2      // sang 1 danh sách có cùng kiểu dữ liệu      // sử dụng phương thức Arrays.asList()      List<Integer> list1 = Arrays.asList(arraySet1);      List<Integer> list2 = Arrays.asList(arraySet2);        // chuyển List thành Set      Set<Integer> setInteger1 = new HashSet<>(list1);      Set<Integer> setInteger2 = new HashSet<>(list2);        // thêm tất cả các phần tử của setInteger2      // vào trong setInteger1      setInteger1.addAll(setInteger2);        System.out.println("Các phần tử có trong setInteger1: ");      System.out.println(setInteger1);  } |

Kết quả sau khi biên dịch chương trình:



**Toán tử giao**

**Cú pháp**

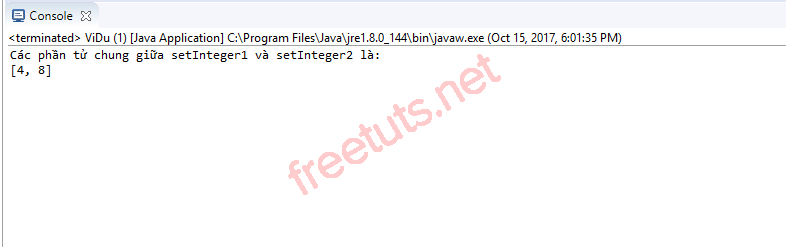
|  |  |
| --- | --- |
| 1 | set1.retainAll(set2); |

Phương thức retainAll() sẽ loại bỏ các phần tử có trong set1 nhưng không có trong set2 (tức là tìm ra các phần tử chung giữa set1 và set2).

**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) {      Integer[] arraySet1 = {2, 10, 4, 8, 5};      Integer[] arraySet2 = {8, 12, 4};        // chuyển mảng arraySet1 và arraySet2      // sang 1 danh sách có cùng kiểu dữ liệu      // sử dụng phương thức Arrays.asList()      List<Integer> list1 = Arrays.asList(arraySet1);      List<Integer> list2 = Arrays.asList(arraySet2);        // chuyển List thành Set      Set<Integer> setInteger1 = new HashSet<>(list1);      Set<Integer> setInteger2 = new HashSet<>(list2);        // loại bỏ các phần tử có trong set1 nhưng không có trong set2      // các bạn thấy trong ví dụ này      // setInteger1 có 5 phần tử là 2, 10, 4, 8 và 5      // setInteger2 có 3 phần tử là 8, 12 và 4      // nên kết quả của ví dụ này sẽ trả về setInteger1      // bao gồm 2 phần tử là 4 và 8      setInteger1.retainAll(setInteger2);        System.out.println("Các phần tử chung giữa setInteger1 và setInteger2 là: ");      System.out.println(setInteger1);  } |

Kết quả sau khi biên dịch chương trình:



**Toán tử hiệu**

**Cú pháp**

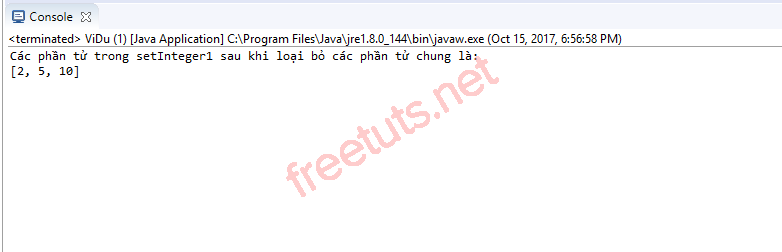
|  |  |
| --- | --- |
| 1 | set1.removeAll(set2); |

Phương thức removeAll() sẽ loại bỏ các phần tử có [trong](https://freetuts.net/set-interface-trong-java-1080.html) set1 và cũng có trong set2 (tức là loại bỏ các phần tử chung giữa set1 và set2).

**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) {      Integer[] arraySet1 = {2, 10, 4, 8, 5};      Integer[] arraySet2 = {8, 12, 4};        // chuyển mảng arraySet1 và arraySet2      // sang 1 danh sách có cùng kiểu dữ liệu      // sử dụng phương thức Arrays.asList()      List<Integer> list1 = Arrays.asList(arraySet1);      List<Integer> list2 = Arrays.asList(arraySet2);        // chuyển List thành Set      Set<Integer> setInteger1 = new HashSet<>(list1);      Set<Integer> setInteger2 = new HashSet<>(list2);        // loại bỏ các phần tử có trong set1 và cũng có trong set2      // các bạn thấy trong ví dụ này      // setInteger1 có 5 phần tử là 2, 10, 4, 8 và 5      // setInteger2 có 3 phần tử là 8, 12 và 4      // nên kết quả của ví dụ này sẽ trả về setInteger1      // bao gồm 2 phần tử là 2, 5 và 10      setInteger1.removeAll(setInteger2);        System.out.println("Các phần tử trong setInteger1 sau khi"          + " loại bỏ các phần tử chung là:  ");      System.out.println(setInteger1);  } |

Kết quả sau khi biên dịch chương trình:



**3. Ví dụ tổng hợp**

Viết chương trình thực hiện các yêu cầu sau:

* Khai báo 1 Set có Class triển khai là HashSet, kiểu dữ liệu là String. Sau đó thêm vào phần tử là tên của các khoa của một trường đại học cho Set này (giá trị của các phần tử được nhập từ bàn phím).
* Hiển thị các phần tử vừa nhập có trong Set vừa nhập sử dụng Iterator.
* Thêm vào một khoa mới vào trong Set, nếu tên khoa đó đã tồn tại thì thông báo cho người dùng biết tên khoa đó đã có, còn ngược lại thêm bình thường và thông báo "Thêm thành công!".
* Xóa một khoa bất kỳ ra khỏi Set. Kiểm tra nếu khoa cần xóa có tồn tại trong Set thì mới xóa và thông báo "Xóa thành công!", ngược lại thông báo "Xóa không thành công!".

**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 | public static void main(String[] args) {      String tenKhoa;      Set<String> khoa = new HashSet<>();      Scanner scanner = new Scanner(System.in);        // thêm phần tử      khoa.add("Khoa Công nghệ thông tin");      khoa.add("Khoa Kinh tế");      khoa.add("Khoa Sư phạm");        // hiển thị phần tử sử dụng Iterator      Iterator<String> iterator = khoa.iterator();        System.out.println("Các phần tử có trong khoa là: ");      while (iterator.hasNext()) {          System.out.println(iterator.next());      }        // thêm khoa mới      // nếu tên khoa đó đã tồn tại thì thông báo tên khoa đó đã có      // còn ngược lại thêm bình thường và thông báo "Thêm thành công!".      System.out.println("Nhập tên khoa cần thêm: ");      tenKhoa = scanner.nextLine();      if (khoa.contains(tenKhoa)) {          System.out.println("Khoa" + tenKhoa + " đã tồn tại!");      } else {          khoa.add(tenKhoa);          iterator = khoa.iterator();          System.out.println("Các phần tử có trong khoa sau khi thêm là: ");          while (iterator.hasNext()) {              System.out.println(iterator.next());          }      }        // xóa khoa      // nếu khoa cần xóa có tồn tại trong Set thì mới xóa và thông báo "Xóa thành công!"      // ngược lại thông báo "Xóa không thành công!".      System.out.println("Nhập tên khoa cần xóa: ");      tenKhoa = scanner.nextLine();      if (khoa.contains(tenKhoa)) {          khoa.remove(tenKhoa);          System.out.println("Xóa thành công!");          System.out.println("Các phần tử có trong khoa sau khi xóa là: ");          iterator = khoa.iterator();          while (iterator.hasNext()) {              System.out.println(iterator.next());          }      } else {          System.out.println("Xóa không thành công!");      }  } |

Kết quả sau khi biên dịch chương trình:

