**HashSet trong Java**

**1. Đặc điểm**

HashSet là Class triển khai phổ biến nhất của Set Interface nên nó sẽ có một vài đặc điểm và phương thức tương đồng với Set. 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 HashSet không dựa theo thứ tự lúc thêm vào và giá trị của các phần tử này là duy nhất.

**2. Initial capacity và Load factor trong HashSet**

Trong HashSet, chúng ta có 2 yếu tố ảnh hưởng trực tiếp tới hiệu suất của nó là **Initial capacity** (kích thước khởi tạo) và **Load factor** (tạm gọi là hệ số tải).

* Kích thước của 1 HashSet là số lượng bucket có trong HashSet (bucket ở đây là nơi mà chúng ta lưu trữ các value của HashSet). Kích thước khởi tạo của HashSet là kích thước ngay tại thời điểm nó được khởi tạo và mặc định là 24 = 16 và nó sẽ tăng lên gấp đôi khi kích thước của HashSet đạt đến ngưỡng, ví dụ kích thước của HashSet sẽ tăng lên 25 = 32, 26 = 64, 27 = 128 ứng với từng ngưỡng mà nó đạt được.
* Hệ số tải là chỉ số để đo xem đến ngưỡng nào thì kích thước (capacity) của HashSet sẽ tự động tăng lên. Giá trị mặc định của hệ số tải là 0.75f.

Vậy làm thế nào để tính ngưỡng của HashSet? Để tính ngưỡng của 1 HashSet chúng ta sẽ áp dụng theo công thức sau:

**Threshold = Current capacity \* Load factor**

, trong đó Threshold là ngưỡng mà chúng ta cần tính, Current capacity và Load lactor lần lượt là kích thước hiện tại và yếu tố tải của HashSet.

**Ví dụ:** Giả sử 1 HashSet được khởi tạo với kích thước ban đầu là 16 và yếu tố tải là 0.75 thì ngưỡng của HashSet này sẽ = 16 \* 0.75 = 12. Con số 12 này có nghĩa là kích thước của HashSet này sẽ tăng từ 16 lên 32 sau khi phần tử thứ 12 được thêm vào trong HashSet đó.

Ý nghĩa của Initial capacity và Load factor đối với hiệu suất của HashSet: Đây là 2 yếu tố chính ảnh hưởng tới hiệu suất hoạt động của HashSet, vì vậy việc lựa chọn kích thước ban đầu và hệ số tải phù hợp sẽ giúp cho hiệu suất hoạt động của HashSet đạt hiệu quả tốt trong những yêu cầu phức tạp mà chúng ta gặp phải khi lập trình với HashSet.

**3. Các thao tác cơ bản trên HashSet**

### **3.1. Tạo mới một HashSet**

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

**Cú pháp**

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

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

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public static void main(String[] args) {      // khai báo 1 HashSet có tên là hashSetInt      // có kiểu là Integer      HashSet<Integer> hashSetInt = new HashSet<>();        // khai báo 1 Hashset có kích thước khởi tạo = 32      HashSet<Character> hashSetChar = new HashSet<>(32);        // khai báo 1 HashSet có kích thước khởi tạo = 16      // và yếu tố tải = 0.75f (mặc định)      HashSet<String> hashSetString = new HashSet<>(16, 0.75f);        // khai báo 1 HashSet được tạo thành từ 1 Collection khác      HashSet<Float> hashSetFloat = new HashSet<>(new TreeSet<>());  } |

### **3.2. Hiển thị các phần tử có trong HashSet.**

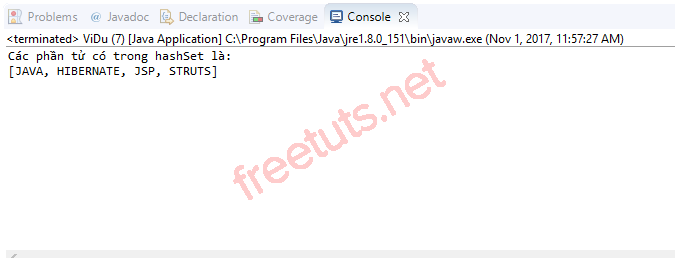
Để hiển thị các phần tử có trong HashSet, chúng ta có các cách như sau:

**Hiển thị theo tên HashSet.**

**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 HashSet có tên là hashSet      // có kiểu là String      HashSet<String> hashSet = new HashSet<>();        // thêm các phần tử vào hashSet sử dụng phương thức add()      hashSet.add("JAVA");      hashSet.add("JSP");      hashSet.add("STRUTS");      hashSet.add("HIBERNATE");      hashSet.add("JSP");      hashSet.add("JAVA");        // hiển thị các phần tử có trong hashSet      // trong hashSet có 2 phần tử là "JAVA"      // mà các phần tử trong 1 HashSet là không trùng nhau      // nên sẽ chỉ có 1 phần tử "JAVA" được hiển thị.      System.out.println("Các phần tử có trong hashSet là: ");      System.out.println(hashSet);  } |

Kết quả sau khi biên dịch chương trình:

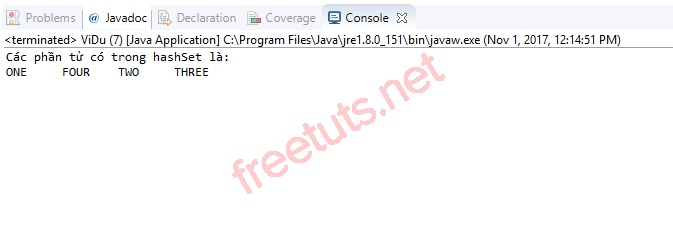


**Sử dụng vòng lặp**for**cải tiến duyệt theo đối tượng trong HashSet.**

**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 HashSet có tên là hashSet      // có kiểu là String      HashSet<String> hashSet = new HashSet<>();        // thêm các phần tử vào hashSet sử dụng phương thức add()      hashSet.add("ONE");      hashSet.add("TWO");      hashSet.add("THREE");      hashSet.add("FOUR");;        // hiển thị các phần tử có trong hashSet      // bằng cách sử dụng vòng lặp for duyệt theo đối tượng      // trong đó kiểu dữ liệu của biến str      // phải trùng với kiểu dữ liệu của hashSet      System.out.println("Các phần tử có trong hashSet là: ");      for (String str : hashSet) {          System.out.print(str + "\t");      }  } |

Kết quả sau khi biên dịch chương trình:



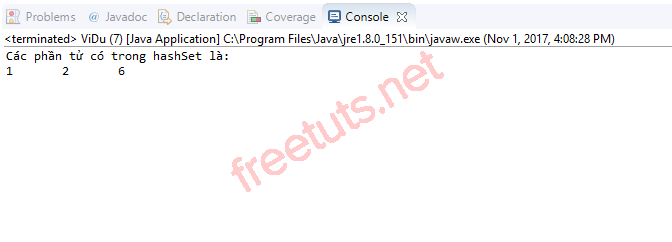
**Sử dụng Iterator.**

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 | public static void main(String[] args) {      // khai báo 1 HashSet có tên là hashSet      // có kiểu là Integer      HashSet<Integer> hashSet = new HashSet<>();        // thêm các phần tử vào hashSet sử dụng phương thức add()      hashSet.add(1);      hashSet.add(2);      hashSet.add(1);      hashSet.add(6);        // khai báo một Iterator có kiểu là Integer      Iterator<Integer> iterator = hashSet.iterator();        // hiển thị các phần tử có trong linkedList      // bằng cách sử dụng Iterator      System.out.println("Các phần tử có trong hashSet là: ");      while (iterator.hasNext()) {          System.out.print(iterator.next() + "\t");      }  } |

Kết quả sau khi biên dịch chương trình:



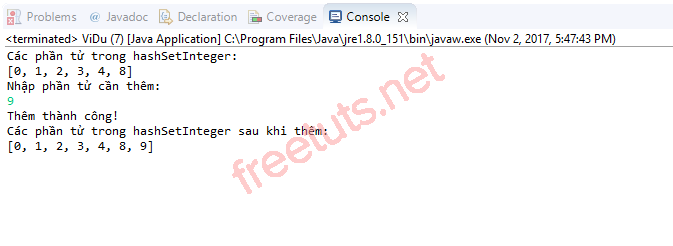
### **3.3. Thêm phần tử vào trong HashSet**

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

Kết quả sau khi biên dịch chương trình:



### **3.4. Xóa phần tử**

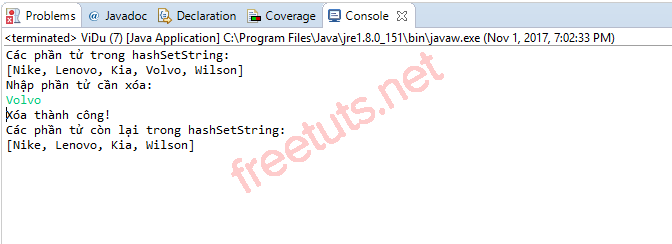
Để xóa một phần tử khỏi HashSet, chúng ta sẽ sử dụng phương thức remove(). Ví dụ dưới đây sẽ sử dụng phương thức remove() để xóa một phần tử bất kỳ trong hashSetString:

**Ví dụ**

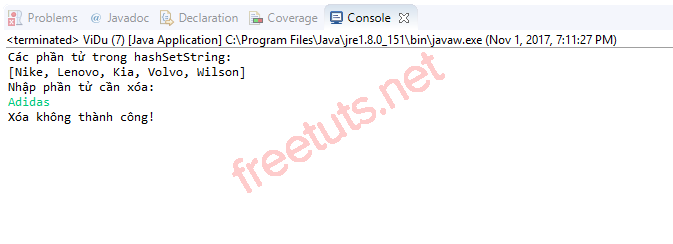
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | public static void main(String[] args) {      String name;      HashSet<String> hashSetString = new HashSet<>();      Scanner scanner = new Scanner(System.in);        // thêm các phần tử vào hashSetString      hashSetString.add("Wilson");      hashSetString.add("Nike");      hashSetString.add("Volvo");      hashSetString.add("Kia");      hashSetString.add("Lenovo");      hashSetString.add("Lenovo");        // hiển thị các phần tử trong hashSetString      System.out.println("Các phần tử trong hashSetString: ");      System.out.println(hashSetString);        System.out.println("Nhập phần tử cần xóa: ");      name = scanner.nextLine();        // nếu phần tử cần xóa      // có tồn tại hashSetString thì sẽ thông báo xóa thành công      // và hiển thị các phần tử còn lại      // ngược lại thông báo xóa không thành công      if (hashSetString.contains(name)) {          hashSetString.remove(name);          System.out.println("Xóa thành công!");          System.out.println("Các phần tử còn lại trong hashSetString:");          System.out.println(hashSetString);      } 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:

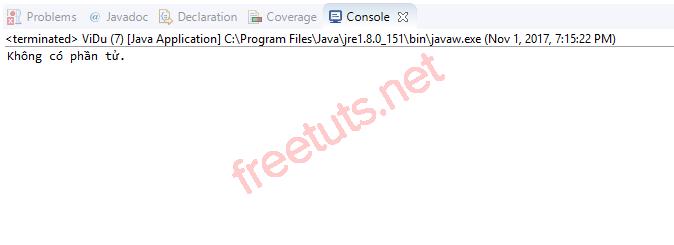


Để xóa tất cả các phần tử có trong HashSet, chúng ta sẽ sử dụng phương thức clear() có sẵn của Java. Ví dụ:

**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) {      float fNumber;      HashSet<Float> hashSetFloat = new HashSet<>();        // thêm các phần tử vào hashSetFloat      hashSetFloat.add(7.14f);      hashSetFloat.add(19.14f);      hashSetFloat.add(1.11f);      hashSetFloat.add(20.14f);        // xóa toàn bộ các phần tử trong hashSetFloat      // sử dụng phương thức clear()      hashSetFloat.clear();        // sau khi xóa thì trong hashSetFloat      // sẽ không có phần tử nào      // phương thức isEmpty() dưới đây sẽ kiểm tra      // nếu hashSetFloat không có giá trị      // thì sẽ hiển thị thông báo "Không có phần tử"      if (hashSetFloat.isEmpty()) {          System.out.println("Không có phần tử.");      }  } |

Kết quả sau khi biên dịch chương trình:



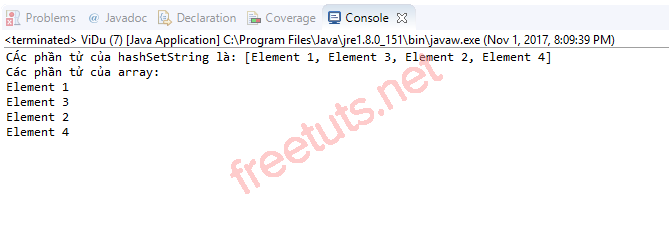
### **3.5. Chuyển đổi HashSet thành mảng (Array)**

Để chuyển đổi 1 HashSet thành mảng, chúng ta sẽ sử dụng phương thức toArray() có sẵn 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 | public static void main(String[] args) {      // Tạo 1 HashSet có tên là hashSetString      HashSet<String> hashSetString = new HashSet<String>();        hashSetString.add("Element 1");      hashSetString.add("Element 2");      hashSetString.add("Element 3");      hashSetString.add("Element 4");        System.out.println("CÁc phần tử của hashSetString là: " + hashSetString);        // Tạo 1 mảng có tên là array và có cùng kiểu dữ liệu với hashSetString      // số phần tử của hashSetString là số phần tử của array      String[] array = new String[hashSetString.size()];        // chuyển hashSetString thành mảng sử dụng toArray()      hashSetString.toArray(array);        // Hiển thị các phần tử của array      System.out.println("Các phần tử của array: ");      for(String temp : array){          System.out.println(temp);      }  } |

Kết quả sau khi biên dịch chương trình:



**3. Ví dụ tổng hợp**

 Viết chương trình thực hiện các công việc sau:

* Khai báo 1 HashSet có kiểu dữ liệu là String. Sau đó thêm vào HashSet này tên của các loại trái cây được nhập bất kỳ từ bàn phím.
* Hiển thị số phần tử có trong HashSet vừa tạo.
* Nhập vào tên 1 loại trái cây và kiểm tra loại trái cây đó có tồn tại trong HashSet này hay không. Nếu có thì thông báo có tìm thấy, ngược lại thì thông báo không tìm thấy.
* Xóa 1 loại trái cây bất kỳ trong HashSet đó và hiển thị các phần tử còn lại.
* Tạo 1 List mới có cùng kiểu dữ liệu với HashSet và thêm các phần tử của List này. Sau đó thêm các phần tử của List này vào trong HashSet ban đầu và hiển thị lại HashSet này sử dụng Iterator.
* Xóa các phần tử của List có trong HashSet và hiển thị lại HashSet.

**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  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66 | public static void main(String[] args) {      String fruitName;      int n;      Scanner scanner = new Scanner(System.in);        // Tạo 1 HashSet có tên là hashSetFruits      HashSet<String> hashSetFruits = new HashSet<String>();        System.out.println("Nhập vào số phần tử của hashSetFruits: ");      n = Integer.parseInt(scanner.nextLine());   // hạn chế hiện tượng trôi lệnh        System.out.println("Nhập vào tên các loại trái cây: ");      for (int i = 0; i < n; i++) {          System.out.print("Nhập tên loại trái cây thứ " + i + ": ");          fruitName = scanner.nextLine();          hashSetFruits.add(fruitName);      }        // hiển thị số phần tử của hashSetFruits      // sử dụng phương thức size()      System.out.println("Số phần tử của hashSetFruits = " + hashSetFruits.size());        // tìm loại trái cây      System.out.println("Nhập tên loại trái cây cần tìm: ");      String fruitSearch = scanner.nextLine();      if (hashSetFruits.contains(fruitSearch)) {          System.out.println("Có trái cây " + fruitSearch + " trong hashSetFruits!");      } else {          System.out.println("Không tìm thấy " + fruitSearch);      }        // Xóa 1 loại trái cây bất kỳ trong HashSet đó và hiển thị các phần tử còn lại.      System.out.println("Nhập vào tên loại trái cây cần xóa: ");      String fruitDelete = scanner.nextLine();        if (hashSetFruits.contains(fruitDelete)) {          hashSetFruits.remove(fruitSearch);          System.out.println("Xóa thành công!");          System.out.println("Các phần tử còn lại trong hashSetFruits: " + hashSetFruits);      } else {          System.out.println("Xóa không thành công!");      }        // Tạo 1 List mới có cùng kiểu dữ liệu với HashSet      // và thêm các phần tử của List này.      List<String> listFruits = new ArrayList<>();      listFruits.add("Apple");      listFruits.add("Banana");      listFruits.add("Mango");      listFruits.add("Apple");        // thêm các phần tử của List này vào trong HashSet ban đầu      // và hiển thị lại HashSet này sử dụng Iterator.      // sử dụng phương thức addAll()      hashSetFruits.addAll(listFruits);      System.out.println("Các phần tử có trong hashSetFruits sau khi thêm: ");      Iterator<String> iterator = hashSetFruits.iterator();      while (iterator.hasNext()) {          System.out.print(iterator.next() + "\t");      }        // Xóa các phần tử của List có trong HashSet và hiển thị lại HashSet      // sử dụng phương thức removeAll()      hashSetFruits.removeAll(listFruits);      System.out.println("\nCác phần tử có trong hashSetFruits sau khi xóa: " + hashSetFruits);  } |

Kết quả sau khi biên dịch chương trình:

