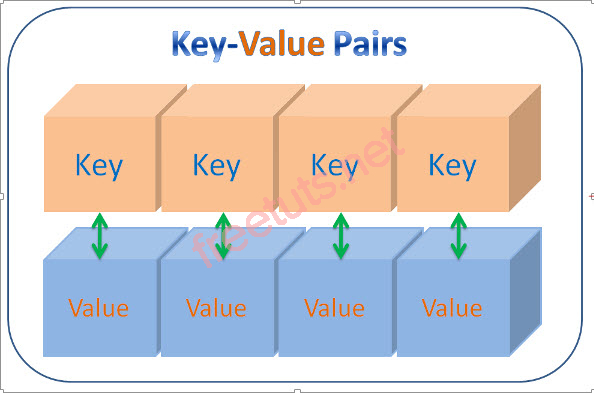
**Map Interface trong Java**

**1. Đặc điểm**

**Map là một tập các cặp khóa - giá trị (key - value)**. Giá trị của các phần tử trong Map có thể giống nhau, nhưng **khóa thì không được giống nhau** (vì thế chúng ta có thể tạo ra 1 Set có các phần tử là khóa của Map). Dựa vào khóa, chúng ta có thể xác định được các giá trị value tương ứng với khóa đó. Dưới đây là hình ảnh minh họa mối quan hệ giữa key và value trong Map:



Map được sử dụng trong trường hợp chúng ta muốn truy xuất, cập nhật hoặc tìm kiếm phần tử thông qua khóa của phần tử đó. Ví dụ:

* Một Map bao gồm thông tin của người quản lý và nhân viên trong một công ty. Mỗi một người quản lý (key) sẽ liên kết với danh sách các nhân viên mà người đó quản lý (value).
* Một Map bao gồm thông tin của một lớp học và các sinh viên có trong lớp đó. Mỗi một lớp (key) sẽ liên kết với danh sách các sinh viên của lớp đó (value).

**2. Các phương thức phổ biến**

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

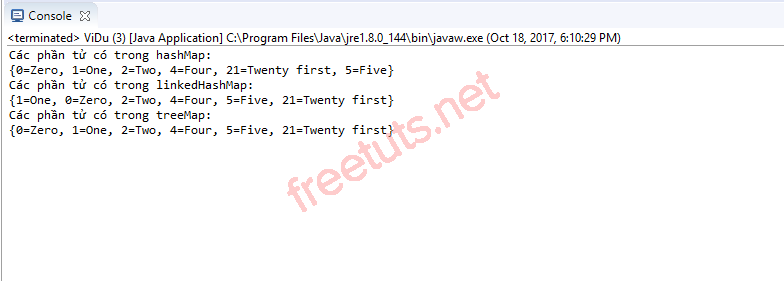
Trong bài [Tổng quan](https://freetuts.net/tong-quan-ve-collections-trong-java-1070.html" \o "tong quan ve collections trong java 1070 html" \t "https://freetuts.net/_blank), tôi có trình bày những thành phần của Collections Framework, trong đó tôi có đề cập đến **Implementations**là sự triển khai các Interface (ví dụ như các Class), **vì vậy để khai báo một Map chúng ta cần phải dùng đến các Class để triển khai nó**, trong phần này chúng ta sẽ sử dụng 3 loại phổ biến nhất là HashMap, LinkedHashMap và TreeMap. Đối với Map Interface có Class triển khai là HashMap thì thứ tự các phần tử không dựa theo thứ tự lúc thêm vào, đối với Map Interface có Class triển khai là LinkedHashMap thì thứ tự các phần tử dựa theo thứ tự lúc thêm vào, còn đối với Map Interface có Class triển khai là TreeMap thì thứ tự các phần tử được sắp xếp theo chiều tăng dần của khóa. Ví dụ dưới đây sẽ cho các bạn thấy sự khác nhau khi sử dụng HashMap, LinkedHashMap và TreeMap để khai báo Map trong Java:

*Bài viết này được đăng tại [free tuts .net]*

**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 | public static void main(String[] args) {      // khai báo Map Interface tên hashMap      // và sử dụng Class là HashMap để triển khai      // HashMap là 1 Class Collection      // mỗi phần tử trong hashMap bao gồm 2 phần      // key (Integer) và value (String)      Map<Integer, String> hashMap = new HashMap<>();        // Thêm value vào trong hashMap với key tương ứng      // sử dụng phương thức put()      // đối số thứ nhất trong put là key có kiểu là Integer      // và đối số thứ hai là value có kiểu là String      hashMap.put(1, "One");      hashMap.put(0, "Zero");      hashMap.put(2, "Two");      hashMap.put(4, "Four");      hashMap.put(21, "Twenty first");      hashMap.put(5, "Five");        // khai báo Map Interface tên linkedHashMap      // và sử dụng Class là LinkedHashMap để triển khai      // LinkedHashMap là 1 Class Collection      // mỗi phần tử trong linkedHashMap bao gồm 2 phần      // key (Integer) và value (String)      Map<Integer, String> linkedHashMap = new LinkedHashMap<>();        // Thêm value vào trong linkedHashMap với key tương ứng      linkedHashMap.put(1, "One");      linkedHashMap.put(0, "Zero");      linkedHashMap.put(2, "Two");      linkedHashMap.put(4, "Four");      linkedHashMap.put(5, "Five");      linkedHashMap.put(21, "Twenty first");        // khai báo Map Interface tên treeMap      // và sử dụng Class là TreeMap để triển khai      // TreeMap là 1 Class Collection      // mỗi phần tử trong treeMap bao gồm 2 phần      // key (Integer) và value (String)      Map<Integer, String> treeMap = new TreeMap<>();        // Thêm value vào trong treeMap với key tương ứng      treeMap.put(1, "One");      treeMap.put(0, "Zero");      treeMap.put(2, "Two");      treeMap.put(4, "Four");      treeMap.put(21, "Twenty first");      treeMap.put(5, "Five");        System.out.println("Các phần tử có trong hashMap: ");      System.out.println(hashMap);      System.out.println("Các phần tử có trong linkedHashMap: ");      System.out.println(linkedHashMap);      System.out.println("Các phần tử có trong treeMap: ");      System.out.println(treeMap);  } |

Kết quả sau khi biên dịch chương trình:



**Lưu ý:** Để khai báo Map chúng ta cần phải import gói thư viện java.util.Map, đối với HashMap thì import gói thư viện java.util.HashMap, đối với LinkedHashMap thì import gói thư viện java.util.LinkedHashMap và với TreeMap thì import gói thư viện java.util.TreeMap. 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  21  22  23  24  25  26  27 | // Khai báo Map  // thì import gói thư viện java.util.Map  import java.util.Map;  public class TênClass {      // ...  }    // Khai báo HashMap  // thì import gói thư viện java.util.HashMap  import java.util.HashMap;  public class TênClass {      // ...  }    // Khai báo LinkedHashMap  // thì import gói thư viện java.util.LinkedHashMap  import java.util.LinkedHashMap;  public class TênClass {      // ...  }    // Khai báo TreeMap  // thì import gói thư viện java.util.TreeMap  import java.util.TreeMap;  public class TênClass {      // ...  } |

### **2.2. Các cách lấy giá trị của Map**

**Sử dụng vòng lặp for cải tiến.**

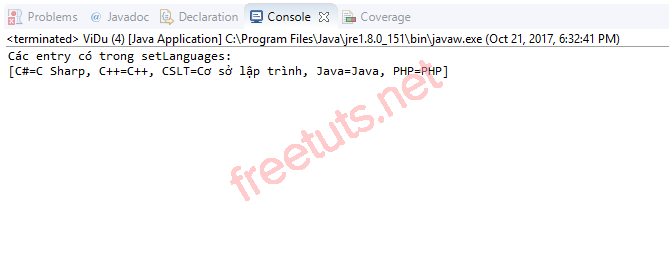
**Lấy**[toàn](https://freetuts.net/map-interface-trong-java-1087.html)**bộ các entry của Map.**

Để lấy toàn bộ các entry (1 entry sẽ bao gồm key và value tương ứng với key đó) của Map, Java cung cấp cho chúng ta phương thức entrySet(). Phương thức này sẽ trả về 1 Set bao gồm các entry có [trong](https://freetuts.net/map-interface-trong-java-1087.html) Map. Ví dụ dưới đây sẽ minh họa cách sử dụng phương thức này.

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public static void main(String[] args) {      Map<String, String> mapLanguages = new TreeMap<>();      mapLanguages.put("CSLT", "Cơ sở lập trình");      mapLanguages.put("C++", "C++");      mapLanguages.put("C#", "C Sharp");      mapLanguages.put("PHP", "PHP");      mapLanguages.put("Java", "Java");        // tạo 1 Set có tên là setLanguages      // chứa toàn bộ các entry (vừa key vừa value)      // của mapLanguages      Set<Map.Entry<String, String>> setLanguages = mapLanguages.entrySet();        System.out.println("Các entry có trong setLanguages:");      System.out.println(setLanguages);  } |

Kết quả sau khi biên dịch chương trình:

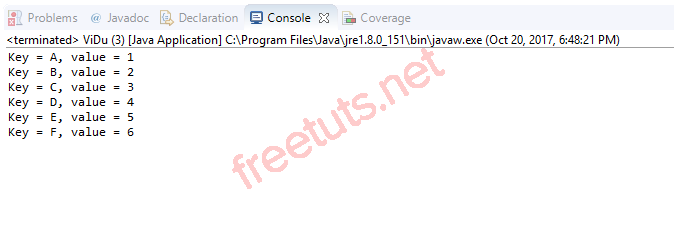


Ngoài ra, kể từ Java 8 trở đi chúng ta có thể lấy toàn bộ các entry trong Map bằng cách sử dụng forEach() như sau:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public static void main(String[] args) {      Map<Character, Integer> mapChar = new TreeMap<>();      mapChar.put('A', 1);      mapChar.put('B', 2);      mapChar.put('C', 3);      mapChar.put('D', 4);      mapChar.put('E', 5);      mapChar.put('F', 6);        // Cách duyệt Map với forEach() trong Java 8      // đối số thứ nhất bên trong forEach là key      // đối số thứ hai bên trong forEach là value      mapChar.forEach((keyChar, valueInt) -> System.out.println(          "Key = " + keyChar + ", value = " + valueInt));  } |

Kết quả sau khi biên dịch chương trình:



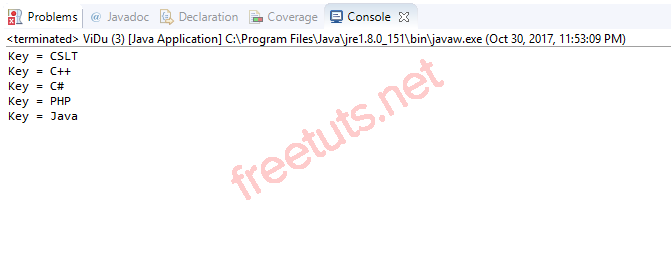
**Lấy toàn bộ key của Map.**

Để lấy toàn bộ key của Map, Java cung cấp cho chúng ta phương thức keySet(). Phương thức này sẽ trả về 1 Set bao gồm các key có trong Map. Ví dụ dưới đây sẽ minh họa cách sử dụng phương thức này.

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public static void main(String[] args) {      Map<String, String> mapLanguages = new LinkedHashMap<>();      mapLanguages.put("CSLT", "Cơ sở lập trình");      mapLanguages.put("C++", "C++");      mapLanguages.put("C#", "C Sharp");      mapLanguages.put("PHP", "PHP");      mapLanguages.put("Java", "Java");        // phương thức keySet()      // sẽ trả về 1 Set chứa key có trong Map      for (String key : mapLanguages.keySet()) {          System.out.println("Key = " + key);      }  } |

Kết quả sau khi biên dịch chương trình:



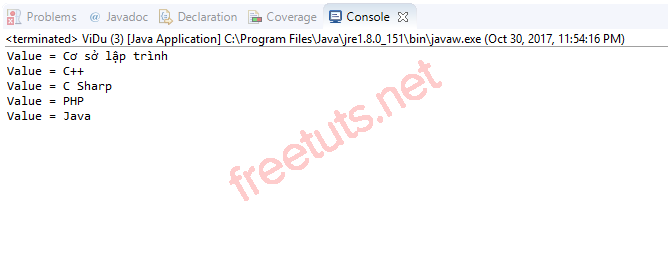
**Lấy toàn bộ value của Map.**

Để lấy toàn bộ value của Map, Java cung cấp cho chúng ta phương thức values(). Phương thức này sẽ trả về 1 tập hợp bao gồm các value có trong Map. Ví dụ dưới đây sẽ minh họa cách sử dụng phương thức này.

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public static void main(String[] args) {      Map<String, String> mapLanguages = new LinkedHashMap<>();      mapLanguages.put("CSLT", "Cơ sở lập trình");      mapLanguages.put("C++", "C++");      mapLanguages.put("C#", "C Sharp");      mapLanguages.put("PHP", "PHP");      mapLanguages.put("Java", "Java");        // phương thức values() sẽ trả về      // một tập hợp gồm các values có trong Map      for (String value: mapLanguages.values()) {          System.out.println("Value = " + value);      }  } |

Kết quả sau khi biên dịch chương trình:



**Sử dụng Iterator.**

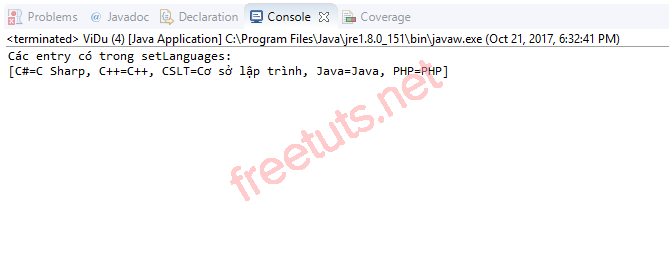
Để sử dụng được Iterator chúng ta cần phải import gói thư viện java.util.Iterator của Java.

Lấy toàn bộ các entry của Map.

**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) {      Map<String, String> mapLanguages = new TreeMap<>();      mapLanguages.put("CSLT", "Cơ sở lập trình");      mapLanguages.put("C++", "C++");      mapLanguages.put("C#", "C Sharp");      mapLanguages.put("PHP", "PHP");      mapLanguages.put("Java", "Java");        // sử dụng Iterator để lấy toàn bộ entry của Map      // vì 1 entry bao gồm key và value      // nên kiểu dữ liệu của Iterator sẽ bao gồm      // kiểu dữ liệu của cả key và value      Iterator<Map.Entry<String, String>> iterator = mapLanguages.entrySet().iterator();        System.out.println("Các entry có trong mapLanguages là: ");      while (iterator.hasNext()) {          System.out.println(iterator.next());      }  } |

Kết quả sau khi biên dịch chương trình:

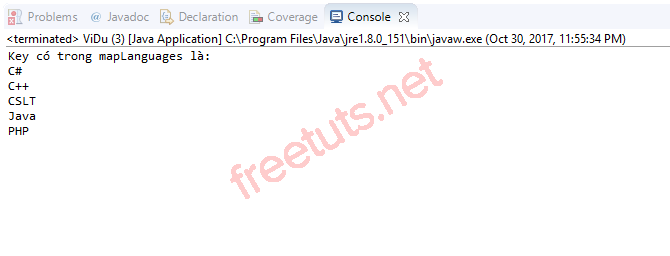


**Lấy toàn bộ key của Map.**

**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) {      Map<String, String> mapLanguages = new TreeMap<>();      mapLanguages.put("CSLT", "Cơ sở lập trình");      mapLanguages.put("C++", "C++");      mapLanguages.put("C#", "C Sharp");      mapLanguages.put("PHP", "PHP");      mapLanguages.put("Java", "Java");        // sử dụng Iterator để lấy toàn bộ key của Map      // thông qua phương thức keySet()      // vì các key có kiểu là String      // nên iterator cũng có kiểu là String      Iterator<String> iterator = mapLanguages.keySet().iterator();        System.out.println("Key có trong mapLanguages là: ");      while (iterator.hasNext()) {          System.out.println(iterator.next());      }  } |

Kết quả sau khi biên dịch chương trình:

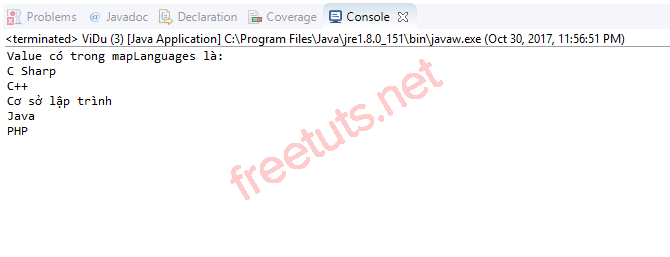


**Lấy toàn bộ value của Map.**

**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) {      Map<String, String> mapLanguages = new TreeMap<>();      mapLanguages.put("CSLT", "Cơ sở lập trình");      mapLanguages.put("C++", "C++");      mapLanguages.put("C#", "C Sharp");      mapLanguages.put("PHP", "PHP");      mapLanguages.put("Java", "Java");        // sử dụng Iterator để lấy toàn bộ value của Map      // thông qua phương thức values()      // vì các value có kiểu là String      // nên iterator cũng có kiểu là String      Iterator<String> iterator = mapLanguages.values().iterator();        System.out.println("Value có trong mapLanguages là: ");      while (iterator.hasNext()) {          System.out.println(iterator.next());      }  } |

Kết quả sau khi biên dịch chương trình:



### **2.3. Thêm dữ liệu vào trong Map**

Để thêm dữ liệu vào trong Map, Java cung cấp cho chúng ta phương thức put().

**Cú pháp**

|  |  |
| --- | --- |
| 1 | put(K key, V value); |

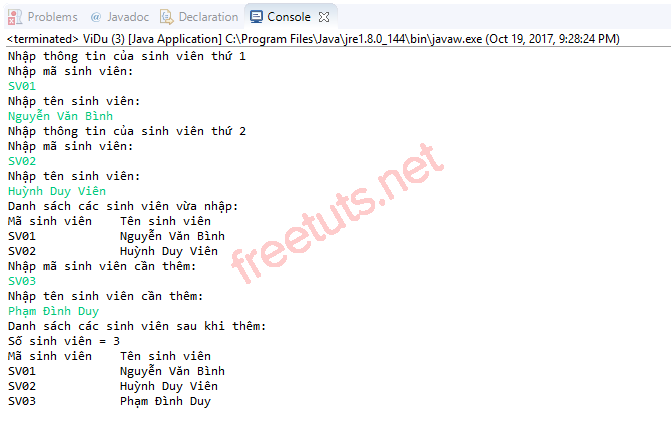
, trong đó: key là khóa, value là giá trị. Mỗi key sẽ tương ứng với một value cụ thể.

**Ví dụ**

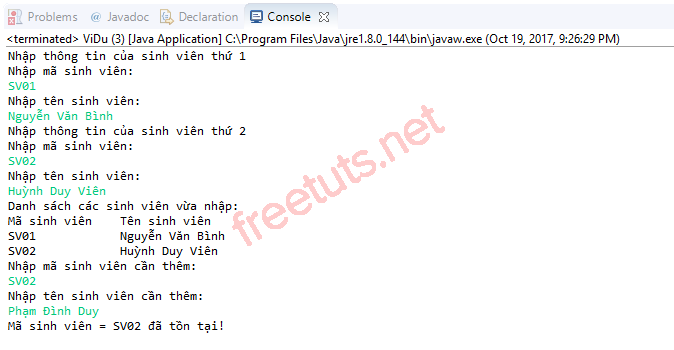
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55 | public static void main(String[] args) {      int soSinhVien = 2;      Map<String, String> mapStudents = new TreeMap<>();      Scanner scanner = new Scanner(System.in);      String maSinhVien, tenSinhVien;        // thêm thông tin của 2 sinh viên      // vào trong mapStudents      // trong đó key là mã sinh viên, còn value là tên của sinh viên đó      for (int i = 1; i <= soSinhVien; i++) {          System.out.println("Nhập thông tin của sinh viên thứ " + i);          System.out.println("Nhập mã sinh viên: ");          maSinhVien = scanner.nextLine();          System.out.println("Nhập tên sinh viên: ");          tenSinhVien = scanner.nextLine();          mapStudents.put(maSinhVien, tenSinhVien);      }        // hiển thị danh sách sinh viên sử dụng Iterator      System.out.println("Danh sách các sinh viên vừa nhập: ");      System.out.println("Mã sinh viên\tTên sinh viên");      Iterator<Map.Entry<String, String>> iterator = mapStudents.entrySet().iterator();      while (iterator.hasNext()) {          // tạo 1 entry          Map.Entry<String, String> entry = iterator.next();          System.out.println(entry.getKey() + "\t\t" + entry.getValue());      }        // thêm 1 sinh viên mới vào trong mapStudents      // nếu mã sinh viên đó đã tồn tại thì thông báo mã đã tồn tại      // ngược lại thêm vào bình thường và thông báo "Thêm thành công"      // sau đó tăng số sinh viên lên 1      System.out.println("Nhập mã sinh viên cần thêm: ");      String maSinhVienMoi = scanner.nextLine();      System.out.println("Nhập tên sinh viên cần thêm: ");      String tenSinhVienMoi = scanner.nextLine();        // phương thức containsKey() sẽ kiểm tra mã sinh viên mới nhập vào      // có tồn tại trong mapStudents hay chưa      if (mapStudents.containsKey(maSinhVienMoi)) {          System.out.println("Mã sinh viên = " + maSinhVienMoi + " đã tồn tại!");      } else {          mapStudents.put(maSinhVienMoi, tenSinhVienMoi);          soSinhVien++;          System.out.println("Danh sách các sinh viên sau khi thêm: ");          System.out.println("Số sinh viên = " + soSinhVien);          System.out.println("Mã sinh viên\tTên sinh viên");          iterator = mapStudents.entrySet().iterator();          while (iterator.hasNext()) {              // tạo 1 entry              Map.Entry<String, String> entry = iterator.next();              System.out.println(entry.getKey() + "\t\t" + entry.getValue());          }      }  } |

**Kết quả sau khi biên dịch chương trình:**

Thêm dữ liệu thành công:



Thêm dữ liệu thất bại:



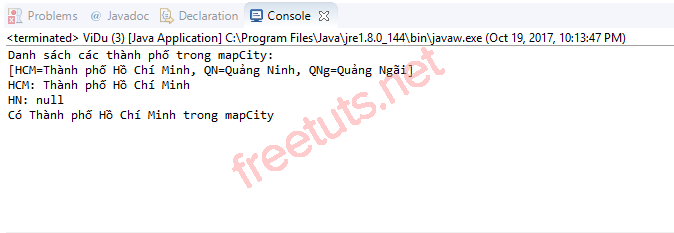
### **2.4. Lấy dữ liệu value [trong](https://freetuts.net/map-interface-trong-java-1087.html) Map khi biết được key**

Để lấy dữ liệu value trong Map khi biết được key liên kết với value đó, Java cung cấp cho chúng ta phương thức get(). Phương thức này sẽ trả về giá trị (value) tương ứng với key đó, nếu trong Map không có key đó thì phương thức này sẽ trả về giá trị null.

**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) {      Map<String, String> mapCity = new TreeMap<>();      mapCity.put("QNg", "Quảng Ngãi");      mapCity.put("QN", "Quảng Nam");      // trong trường hợp này ta thấy      // key của Quảng Nam và Quảng Ninh      // đều là QN nên chương trình sẽ thêm      // vào trong Map value đứng sau (tức là Quảng Ninh)      mapCity.put("QN", "Quảng Ninh");      mapCity.put("HCM", "Thành phố Hồ Chí Minh");        System.out.println("Danh sách các thành phố trong mapCity: ");      Set<Map.Entry<String, String>> setCity = mapCity.entrySet();      System.out.println(setCity);        // lấy thành phố có mã là HCM      // và hiển thị tên thành phố      System.out.println("HCM: " + mapCity.get("HCM"));        // lấy thành phố có mã là HN      // vì trong mapCity không có thành phố nào có mã là HN      // nên sẽ hiển thị giá trị null      System.out.println("HN: " + mapCity.get("HN"));        // Để kiểm tra xem 1 value có trong Map hay không      // chúng ta sẽ dùng phương thức containsValue()      if (mapCity.containsValue("Thành phố Hồ Chí Minh")) {          System.out.println("Có Thành phố Hồ Chí Minh trong mapCity");      }  } |

Kết quả sau khi biên dịch chương trình:



### **2.5. Xóa 1 entry trong Map**

Để xóa 1 entry trong Map, Java cung cấp cho chúng ta phương thức remove().

**Cú pháp**

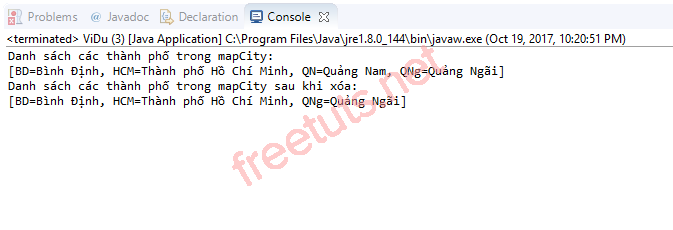
|  |  |
| --- | --- |
| 1 | remove(K key); |

, trong đó key là khóa của entry cần xóa.

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | public static void main(String[] args) {      Map<String, String> mapCity = new TreeMap<>();      mapCity.put("QNg", "Quảng Ngãi");      mapCity.put("QN", "Quảng Nam");      mapCity.put("BD", "Bình Định");      mapCity.put("HCM", "Thành phố Hồ Chí Minh");        System.out.println("Danh sách các thành phố trong mapCity: ");      Set<Map.Entry<String, String>> setCity = mapCity.entrySet();      System.out.println(setCity);        // xóa entry có khóa là QN ra khỏi mapCity      // sử dụng phương thức remove()      mapCity.remove("QN");        System.out.println("Danh sách các thành phố trong mapCity sau khi xóa: ");      System.out.println(setCity);  } |

Kết quả sau khi biên dịch chương trình:



### **2.6. Thay thế value của 1 entry trong Map**

Để thay thế value của 1 entry trong Map, Java cung cấp cho chúng ta 2 dạng của phương thức replace() như sau:.

**Cú pháp 1**

|  |  |
| --- | --- |
| 1 | replace(K key, V value); |

,trong đó key là khóa của entry cần thay thế, value là giá trị mới được thay thế.

**Cú pháp 2**

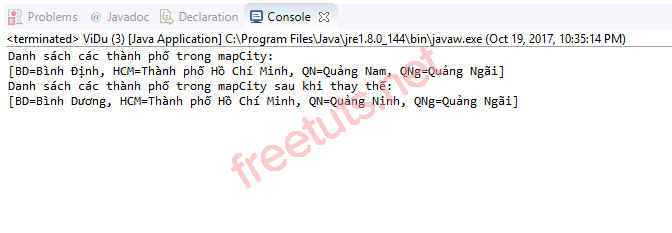
|  |  |
| --- | --- |
| 1 | replace(K key, V oldValue, V newValue); |

, trong đó key là khóa của entry cần thay thế, oldValue là giá trị cần thay thế, newValue là giá trị mới được 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 | public static void main(String[] args) {      Map<String, String> mapCity = new TreeMap<>();      mapCity.put("QNg", "Quảng Ngãi");      mapCity.put("QN", "Quảng Nam");      mapCity.put("BD", "Bình Định");      mapCity.put("HCM", "Thành phố Hồ Chí Minh");        System.out.println("Danh sách các thành phố trong mapCity: ");      Set<Map.Entry<String, String>> setCity = mapCity.entrySet();      System.out.println(setCity);        // thay thế value của entry có khóa là QN      // thành Quảng Ninh      mapCity.replace("QN", "Quảng Ninh");        // ngoài ra chúng ta có thế thay thế như sau      // câu lệnh bên dưới sẽ thay thế entry      // có key là BD, value là Bình Định thành Bình Dương      mapCity.replace("BD", "Bình Định", "Bình Dương");        System.out.println("Danh sách các thành phố trong mapCity sau khi thay thế: ");      System.out.println(setCity);  } |

Kết quả sau khi biên dịch chương trình:



### **2.7. Sao chép Map**

Để sao chép các entry có trong Map này vào trong 1 Map khác, Java cung cấp cho chúng ta phương thức putAll().

**Cú pháp**

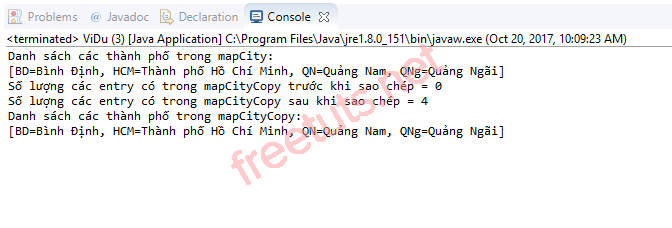
|  |  |
| --- | --- |
| 1 | putAll(Map m); |

, trong đó m là tên của Map được sao chép.

**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 | public static void main(String[] args) {      Map<String, String> mapCity = new TreeMap<>();      mapCity.put("QNg", "Quảng Ngãi");      mapCity.put("QN", "Quảng Nam");      mapCity.put("BD", "Bình Định");      mapCity.put("HCM", "Thành phố Hồ Chí Minh");        System.out.println("Danh sách các thành phố trong mapCity: ");      Set<Map.Entry<String, String>> setCity = mapCity.entrySet();      System.out.println(setCity);        // tạo 1 Map rỗng      Map<String, String> mapCityCopy = new TreeMap<>();        // phương thức size() sẽ trả về số lượng entry có trong Map      System.out.println("Số lượng các entry có trong mapCityCopy "          + "trước khi sao chép = " + (mapCityCopy.size()));        // sao chép các entry của mapCity      // vào trong mapCityCopy      mapCityCopy.putAll(mapCity);        System.out.println("Số lượng các entry có trong mapCityCopy "          + "sau khi sao chép = " + (mapCityCopy.size()));      System.out.println("Danh sách các thành phố trong mapCityCopy: ");      Set<Map.Entry<String, String>> setCityCopy = mapCityCopy.entrySet();      System.out.println(setCityCopy);  } |

Kết quả sau khi biên dịch chương trình:



**3. Ví dụ tổng hợp**

Viết chương trình thực hiện các yêu cầu sau: Nhập vào 1 chuỗi bất kỳ từ bàn phím, sau đó hiển thị độ dài chuỗi vừa nhập vào và các ký tự có trong chuỗi đó (một ký tự chỉ được hiển thị 1 lần).

**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) {      Map<Integer, Character> mapString = new TreeMap<>();      Scanner scanner = new Scanner(System.in);      String str;      Set<Character> setChar;   // lưu trữ các ký tự có trong chuỗi str        System.out.println("Nhập vào chuỗi bất kỳ:");      str = scanner.nextLine();        // chuyển đổi chuỗi str thành 1 mảng các ký tự      char[] charStr = str.toCharArray();        setChar = new TreeSet<Character>();      for (char ch : charStr) {          // thêm các ký tự có trong mảng charStr          // vào trong setChar          // lúc này những ký tự nào giống nhau          // thì chỉ được thêm vào 1 lần          setChar.add(ch);      }        // Hiển thị các ký tự duy nhất trong chuỗi      // và độ dài của chuỗi đó      System.out.println("Độ dài của chuỗi và các ký tự có trong chuỗi là: ");      for (Character ch : setChar) {          mapString.put(str.length(), ch);          System.out.print(mapString.keySet() + "=>" + mapString.values() + "\n");      }  } |

Kết quả sau khi biên dịch chương trình:

