1. **Set trong lập trình Java là gì**

Tập hợp Set được sử dụng để lưu trữ các phần tử duy nhất không bị trùng lặp nhau trong tập hợp. Đây chính là điều khác biệt của Set so với các tập hợp khác. Các phần tử trong Set là duy nhất, chúng ta có thể thêm, xóa, sửa các phần tử trong Set.

Ví dụ

import java.util.HashSet;

public class SetExample {

public static void main(String[] args) {

Set setA = new HashSet();

setA.add(element);

System.out.println( setA.contains(element) );

}

}

Trong ví dụ trên ta tạo ra một tập hợp Set với implementation (lớp thực thi) là HashSet. Ngoài HashSet thì chúng ta có những lớp implementation của Set như sau

* EnumSet
* HashSet
* LinkedHashSet
* TreeSet

Mỗi một implementation của Set đều chứa đựng sự khác biệt về cách sử dụng và tốc độ xử lý các phần tử trong tập hợp. Tuỳ vào yêu cầu bài toán mà ta có thể chọn các loại tập hợp Set tương ứng. Chi tiết mỗi tập hợp sẽ được nói trong từng bài cụ thể sau.

import java.util.HashSet;

import java.util.LinkedHashSet;

import java.util.Set;

import java.util.TreeSet;

public class SetExample {

public static void main(String[] args) {

Set setA = new HashSet();

Set setB = new LinkedHashSet();

Set setC = new TreeSet();

}

}

1. **Thêm phần tử vào Set**

Để thêm một phần tử vào Set ta sử dụng phương thức add như sau

import java.util.HashSet;

import java.util.LinkedHashSet;

import java.util.Set;

import java.util.TreeSet;

public class SetExample {

public static void main(String[] args) {

Set<String> setA = new HashSet<>();

setA.add("element 1");

setA.add("element 2");

setA.add("element 3");

}

}

1. **Duyệt qua các phần tử trong Set**

* Sử dụng Iterator để duyệt qua các phần tử. Các em có thể xem chi tiết iterator hoạt động tại đây

Set<String> setA = new HashSet<>();

setA.add("element 1");

setA.add("element 2");

setA.add("element 3");

Iterator<String> iterator = set.iterator();

while(iterator.hasNext(){

String element = iterator.next();

}

* Sử dụng For Each để duyệt qua các phần tử

Set set = new HashSet();

for(Object object : set) {

String element = (String) object;

}

1. **Xóa một phần tử trong Set**

Chúng ta sử dụng phương thức remove để xóa phần tử trong Set

|  |
| --- |
| set.remove("object-to-remove"); |

1. **Xóa tất cả các phần tử trong Set**

Để xóa tất cả các phần tử ta sử dụng phương thức clear

|  |
| --- |
| set.clear(); |

**Bài tập**

1. Viết chương trình Java để nối phần tử được chỉ định vào cuối tập băm.

2. Viết chương trình Java để lặp qua tất cả các phần tử trong danh sách băm.

3. Viết chương trình Java để lấy số phần tử trong tập băm.

4. Viết chương trình Java để làm trống một tập băm.

5. Viết một chương trình Java để kiểm tra một bộ băm có trống hay không.

6. Viết chương trình Java để sao chép một bộ băm sang một bộ băm khác.

7. Viết chương trình Java để chuyển một bộ băm thành một mảng.

8. Viết chương trình Java để chuyển một tập băm thành tập cây.

9. Viết chương trình Java để chuyển đổi một bộ băm thành Danh sách / ArrayList.

10. Viết chương trình Java để so sánh hai bộ băm.

11. Viết chương trình Java để so sánh hai tập hợp và giữ lại các phần tử giống nhau trên cả hai tập hợp.

12. Viết chương trình Java để loại bỏ tất cả các phần tử khỏi tập băm.

**Map**

**1. Map trong lập trình Java là gì**

Map lưu trữ các phần tử theo định dạng key và value. Anh ví dụ sinh viên thường có mã sinh viên và thông tin sinh viên. Như vậy key ở đây chính là mã sinh viên còn thông tin sinh viên chính là giá trị. Dựa vào key mà chúng ta có thể lấy ra sinh viên tương ứng. Key là giá trị duy nhất không trùng lặp. Như các em thấy mã sinh viên thì là duy nhất không có bạn nào trùng.

**2. Các class thực thi của Map**

Chúng ta có những tập hợp sau cài đặt interface Map đó là

* HashMap
* Hashtable
* EnumMap
* IdendityHashMap
* LinkedHashMap
* Properties
* TreeMap
* WeakHashMap

Trong các tập hợp trên thì HashMap và TreeMap thường hay được sử dụng nhất. HashMap lưu trữ theo giá trị key và value nhưng các phần tử trong nó không được sắp xếp, ngược lại thì các phần tử trong TreeMap đều được sắp xếp.

**3. Tạo một map**

|  |
| --- |
| Map mapA = new HashMap();  Map mapB = new TreeMap(); |

Tạo một Hashmap

|  |
| --- |
| import java.util.HashMap; // import the HashMap class  HashMap<String, String> capitalCities = new HashMap<String, String>(); |

**4. Thêm một phần tử vào HashMap**

Chúng ta sử dụng phương thức put để thêm phần tử vào Map

public class MyClass {

public static void main(String[] args) {

// Create a HashMap object called capitalCities

HashMap<String, String> capitalCities = new HashMap<String, String>();

// Add keys and values (Country, City)

capitalCities.put("England", "London");

capitalCities.put("Germany", "Berlin");

capitalCities.put("Norway", "Oslo");

capitalCities.put("USA", "Washington DC");

System.out.println(capitalCities);

}

}

**Lấy một phần tử trong HashMap**

Để lấy một phần tử trong Map ta sử dụng phương thức get với tham số là key

|  |
| --- |
| Map map = new HashMap();  map.put("key1", "value 1");  String element1 = (String) map.get("key1"); |

**Thêm các phần tử từ một HashMap khác**

Chúng ta sử dụng phương thức putAll để thêm các phần tử từ một tập hợp Map khác

Map<String, String> mapA = new HashMap<>();

mapA.put("key1", "value1");

mapA.put("key2", "value2");

Map<String, String> mapB = new HashMap<>();

mapB.putAll(mapA);

**Xóa một phần tử trong Map**

Chúng ta sử dụng phương thức remove để xóa một phần tử trong Map. Tham số truyền vào là key

|  |
| --- |
| capitalCities.remove("England"); |

Xóa hết các phần tử ta dùng phương thức clear

|  |
| --- |
| capitalCities.clear(); |

**Bài tập**

1. Viết chương trình Java để liên kết giá trị được chỉ định với khóa được chỉ định trong HashMap.

2. Viết chương trình Java để đếm số lượng ánh xạ khóa-giá trị (kích thước) trong một bản đồ.

3. Viết chương trình Java để sao chép tất cả các ánh xạ từ bản đồ được chỉ định sang bản đồ khác.

4. Viết một chương trình Java để loại bỏ tất cả các ánh xạ khỏi một bản đồ.

5. Viết chương trình Java để kiểm tra xem một bản đồ có chứa các ánh xạ khóa-giá trị (trống) hay không.

6. Viết một chương trình Java để lấy một bản sao nông của một cá thể HashMap.

7. Viết một chương trình Java để kiểm tra xem một bản đồ có chứa một ánh xạ cho khóa được chỉ định hay không.

8. Viết một chương trình Java để kiểm tra xem một bản đồ có chứa một ánh xạ cho giá trị được chỉ định hay không.

9. Viết một chương trình Java để tạo một khung nhìn tập hợp của các ánh xạ có trong một bản đồ.

10. Viết chương trình Java để lấy giá trị của một khóa được chỉ định trong bản đồ.

11. Viết một chương trình Java để có được một khung nhìn tập hợp các khóa có trong bản đồ này.

12. Viết một chương trình Java để có được một chế độ xem tập hợp các giá trị có trong bản đồ này.