VIETNAM AUSTRALIA Vocational College

Slide-1.3: Map, HashMap, LinkedHashMap, TreeMap

Giảng viên: Nguyễn Bá Minh Đạo

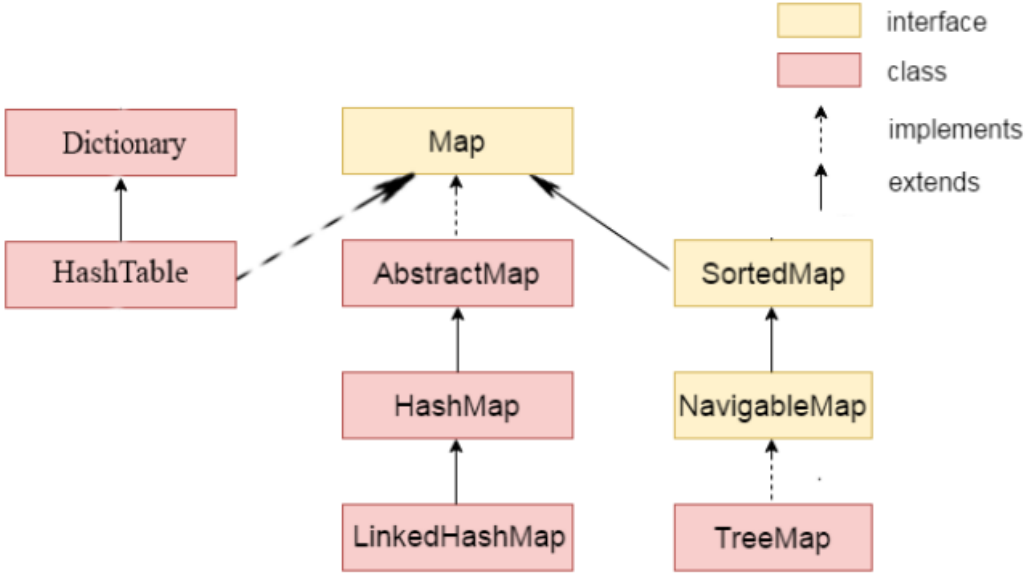
Nội dung

1. Map, HashMap, LinkedHashMap, TreeMap
2. So sánh HashMap và HashSet
3. Chuyển đổi từ HashMap sang ArrayList
4. Giới thiệu lớp Hashtable trong Java
5. So sánh HashMap và Hashtable

Map, HashMap, LinkedHashMap, TreeMap

Giới thiệu Map Interface

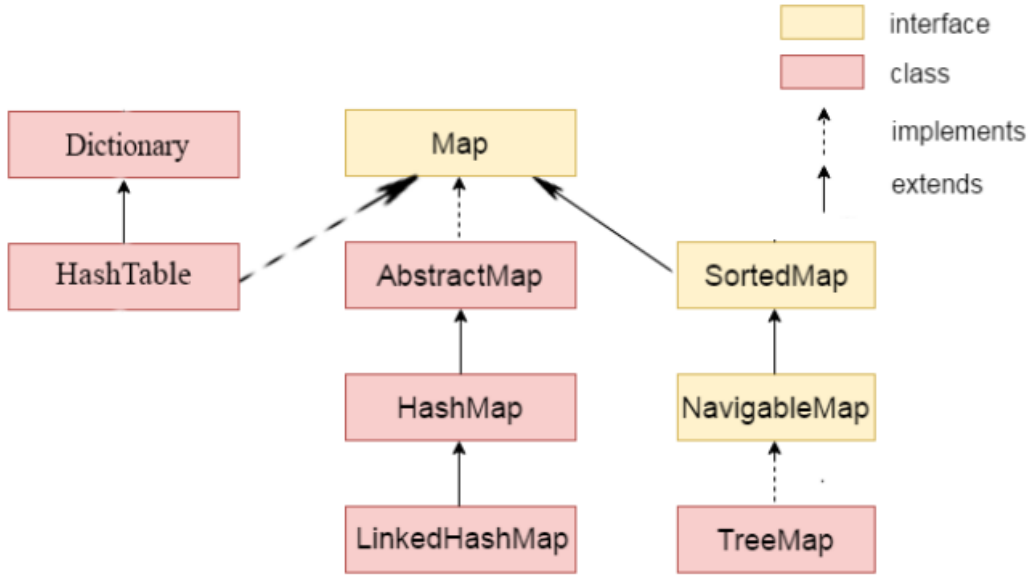
- **Map** (tập hợp) trong Java là một **Interface**, được sử dụng để lưu trữ và truy xuất dữ liệu theo cặp khóa (key) và giá trị (value). Mỗi cặp **key** và **value** được gọi là **entry**.
- **Map** chỉ chứa các giá trị **key** duy nhất, không chứa các **key** trùng lặp.
- Sức chứa (compacity) mặc định khi khởi tạo map là **2^4 = 16. Kích thước này** sẽ tự động tăng gấp đôi mỗi khi thêm phần tử vượt quá kích thước của nó.



Map, HashMap, LinkedHashMap, TreeMap

Giới thiệu Map Interface

- Các lớp cài đặt (implements) **Map** interface là HashMap, LinkedHashMap, TreeMap
  - **HashMap** không đảm bảo thứ tự các entry được thêm vào.
  - **LinkedHashMap** đảm bảo thứ tự các entry được thêm vào.
  - **TreeMap** duy trì thứ tự các phần tử dựa vào bộ so sánh **Comparator**.



Map, HashMap, LinkedHashMap, TreeMap

Các phương thức (method) của Map interface

Phương thức	Mô tả
void clear()	Gỡ bỏ tất cả cặp key/value từ Map đang gọi
boolean containsKey(Object key)	Trả về true nếu Map đang gọi chứa k như là một key. Nếu không là false
boolean containsValue(Object v)	Trả về true nếu Map đang gọi chứa v như là một value. Nếu không là false
boolean equals(Object obj)	Trả về true nếu obj là một Map và chứa cùng các Entry. Nếu không là false.
Object get(Object key)	Trả về value mà liên kết với key
int hashCode()	Trả về hash code cho Map đang gọi

Map, HashMap, LinkedHashMap, TreeMap

Các phương thức (method) của Map interface

Phương thức	Mô tả
boolean isEmpty()	Trả về true nếu Map đang gọi là trống, nếu không là false
Object put(Object key, Object value)	Đặt một entry vào Map đang gọi, ghi đè bất kỳ value trước mà liên kết với key. Với key và value tương ứng là k và v. Trả về null nếu key đã không tồn tại. Nếu không thì, value trước mà liên kết với key được trả về
void putAll(Map map)	Đặt tất cả entry từ m vào trong Map này
Object remove(Object key)	Gỡ bỏ entry mà có khóa là key được chỉ định.

Map, HashMap, LinkedHashMap, TreeMap

Các phương thức (method) của Map interface

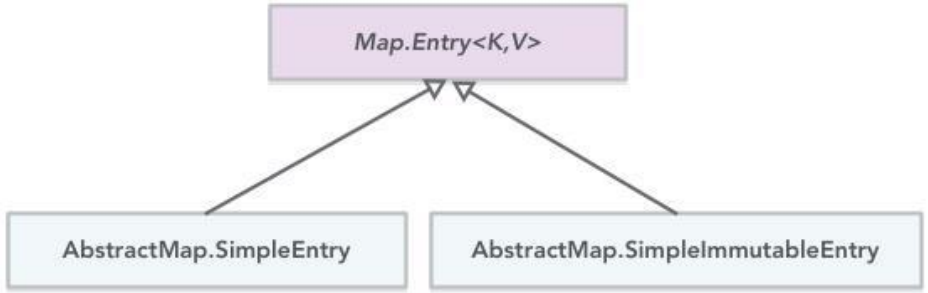
Phương thức	Mô tả
int size()	Trả về số các cặp key/value trong Map
Collection values( )	Trả về một tập hợp chứa các value trong Map. Phương thức này cung cấp một collection-view của các value trong Map
Set keySet()	Nó được sử dụng để trả đối tượng Set có chứa tất cả các keys.
Set entrySet()	Nó được sử dụng để trả lại đối tượng Set có chứa tất cả các keys và values.

Map, HashMap, LinkedHashMap, TreeMap

Giới thiệu Map.Entry Interface

- **Entry** là một interface con của **Map**. Vì vậy, có thể truy cập nó bằng tên **Map.Entry**
- Nó cung cấp các phương pháp để truy xuất các **key** và **value**
- Interface **java.util.Map.Entry** được định nghĩa như sau:

```
public interface Map<K,V> {
    interface Entry<K,V> {

    }
}
```



- Các phương thức của **Map.Entry** interface**:**

Phương thức	Mô tả
Object getKey()	Được dùng để lấy key.
Object getValue()	Được sử dụng để lấy value.

Map, HashMap, LinkedHashMap, TreeMap

Ví dụ sử dụng Map Interface, lớp HashMap, Map.Entry

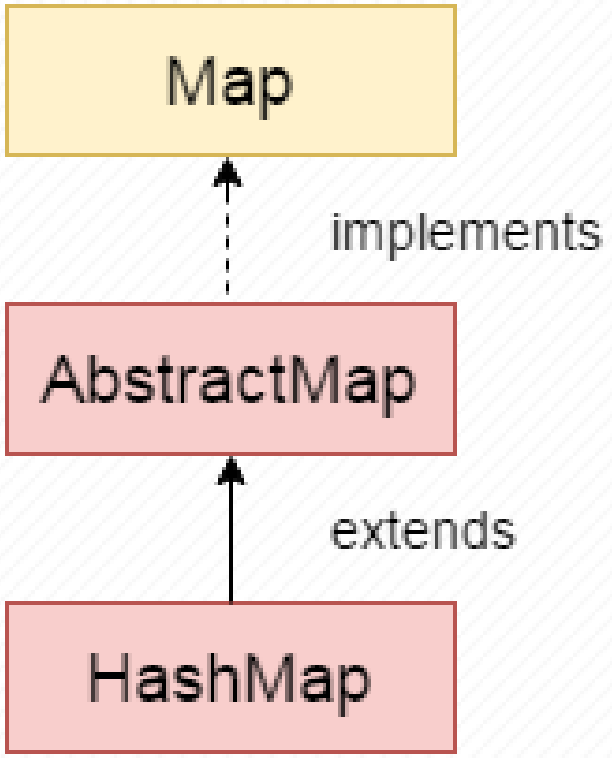
```java
public class MapExample01 {
    public static void main(String args[]) {
        // init map
        Map<Integer, String> map = new HashMap<Integer, String>();
        map.put(1, "Basic java");
        map.put(2, "OOP");
        map.put(3, "Collection");

        // show map using method keySet()
        for (Integer key : map.keySet()) {
            String value = map.get(key);
            System.out.println(key + " = " + value);
        }

        System.out.println("---");

        // show map using method keySet()
        for (Entry<Integer, String> entry : map.entrySet()) {
            Integer key = entry.getKey();
            String value = entry.getValue();
            System.out.println(key + " = " + value);
        }
    }
}
```

Map, HashMap, LinkedHashMap, TreeMap

Giới thiệu lớp HashMap

- **HashMap** lưu trữ dữ liệu dưới dạng cặp **key** và **value**.
- **HashMap** chỉ chứa các **key** duy nhất.
- **HashMap** có thể có 1 **key** là **null** và nhiều giá trị **null**.
- **HashMap** duy trì các phần tử **KHÔNG** theo thứ tự chèn.
- Lớp **java.util.HashMap** được định nghĩa như sau:

```
public class HashMap<K,V> extends AbstractMap<K,V>
        implements Map<K,V>, Cloneable, Serializable {
}
```



- Trong đó:
  - **K**: đây là kiểu **khóa** (key) để lưu trữ.
  - **V**: đây là kiểu **giá trị** (value) được ánh xạ.

Map, HashMap, LinkedHashMap, TreeMap

Các phương thức khởi tạo (constructor) của lớp HashMap

Phương thức	Mô tả
HashMap()	khởi tạo một danh sách map trống
HashMap(Map<? extends K, ? extends V> m)	khởi tạo một map với các phần tử của map m.

Các phương thức (method) của lớp HashMap

❑ Tương tự các phương thức đã được giới thiệu ở bài viết về **Map Interface**

Map, HashMap, LinkedHashMap, TreeMap

Ví dụ sử dụng HashMap với kiểu dữ liệu cơ bản (Wrapper)

```
public class HashMapExample01 {
    public static void main(String args[]) {
        // init map
        Map<Integer, String> map = new HashMap<Integer, String>();
        map.put(1, "Basic java");
        map.put(2, "OOP");
        map.put(3, "Collection");

        // show map using method keySet()
        for (Integer key : map.keySet()) {
            String value = map.get(key);
            System.out.println(key + " = " + value);
        }

        System.out.println("---");

        // show map using method keySet()
        for (Entry<Integer, String> entry : map.entrySet()) {
            Integer key = entry.getKey();
            String value = entry.getValue();
            System.out.println(key + " = " + value);
        }
    }
}
```

Map, HashMap, LinkedHashMap, TreeMap

Ví dụ sử dụng HashMap với key kiểu String, value kiểu Object (Student)

```java
public class Student {
    private int id;
    private String name;

    public Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", "
                + "name=" + name + "]";
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

}
```

```java
public class HashMapExample02 {
    public static void main(String args[]) {
        // Student's data
        Student student1 = new Student(1, "Student 1");
        Student student2 = new Student(2, "Student 2");
        Student student3 = new Student(3, "Student 3");

        // init map
        Map<Integer, Student> map = new HashMap<Integer, Student>();
        map.put(student1.getId(), student1);
        map.put(student2.getId(), student2);
        map.put(student3.getId(), student3);

        // show map using method keySet()
        for (Integer key : map.keySet()) {
            Student value = map.get(key);
            System.out.println(key + " = " + value);
        }

        System.out.println("---");

        // show map using method keySet()
        for (Entry<Integer, Student> entry : map.entrySet()) {
            Integer key = entry.getKey();
            Student value = entry.getValue();
            System.out.println(key + " = " + value);
        }
    }
}
```

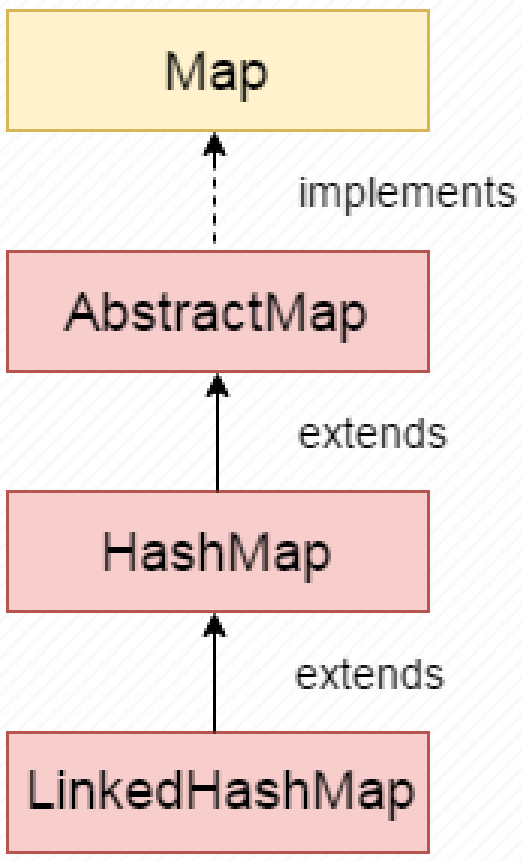
Map, HashMap, LinkedHashMap, TreeMap

Giới thiệu lớp LinkedHashMap

- **LinkedHashMap** lưu trữ dữ liệu dưới dạng cặp **key** và **value**.
- **LinkedHashMap** chỉ chứa các **key** duy nhất.
- **LinkedHashMap** có thể có 1 **key** là **null** và nhiều giá trị **null**.
- **LinkedHashMap** duy trì các phần tử theo thứ tự chèn.
- Lớp **java.util.LinkedHashMap** được định nghĩa như sau:

```
public class LinkedHashMap<K,V>
        extends HashMap<K,V> implements Map<K,V> {
}
```

- Trong đó:
  - **K**: đây là kiểu **khóa** (key) để lưu trữ.
  - **V**: đây là kiểu **giá trị** (value) được ánh xạ.



Map, HashMap, LinkedHashMap, TreeMap

Các phương thức khởi tạo (constructor) của lớp LinkedHashMap

Phương thức	Mô tả
LinkedHashMap()	khởi tạo một danh sách map trống
LinkedHashMap(Map<? extends K, ? extends V> m)	khởi tạo một map với các phần tử của map m.

Các phương thức (method) của lớp LinkedHashMap

❑ Tương tự các phương thức đã được giới thiệu ở bài viết về **Map Interface**

Map, HashMap, LinkedHashMap, TreeMap

Ví dụ sử dụng LinkedHashMap với kiểu dữ liệu cơ bản (Wrapper)

```
public class LinkedHashMapExample01 {
    public static void main(String args[]) {
        // init map
        Map<Integer, String> map = new LinkedHashMap<Integer, String>();
        map.put(1, "Basic java");
        map.put(2, "OOP");
        map.put(4, "Multi-Thread");
        map.put(3, "Collection");

        // show map using method keySet()
        for (Integer key : map.keySet()) {
            String value = map.get(key);
            System.out.println(key + " = " + value);
        }

        System.out.println("---");

        // show map using method keySet()
        for (Entry<Integer, String> entry : map.entrySet()) {
            Integer key = entry.getKey();
            String value = entry.getValue();
            System.out.println(key + " = " + value);
        }
    }
}
```

Map, HashMap, LinkedHashMap, TreeMap

Ví dụ dùng LinkedHashMap với key kiểu String, value kiểu Object (Student)

```java
public class Student {
    private int id;
    private String name;

    public Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", "
                + "name=" + name + "]";
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

}
```

```java
public class LinkedHashMapExample02 {
    public static void main(String args[]) {
        // Student's data
        Student student1 = new Student(1, "Student 1");
        Student student2 = new Student(2, "Student 2");
        Student student3 = new Student(3, "Student 3");
        Student student4 = new Student(4, "Student 4");

        // init map
        Map<Integer, Student> map = new LinkedHashMap<Integer, Student>();
        map.put(student1.getId(), student1);
        map.put(student2.getId(), student2);
        map.put(student4.getId(), student4);
        map.put(student3.getId(), student3);

        // show map using method keySet()
        for (Integer key : map.keySet()) {
            Student value = map.get(key);
            System.out.println(key + " = " + value);
        }

        System.out.println("---");

        // show map using method keySet()
        for (Entry<Integer, Student> entry : map.entrySet()) {
            Integer key = entry.getKey();
            Student value = entry.getValue();
            System.out.println(key + " = " + value);
        }
    }
}
```

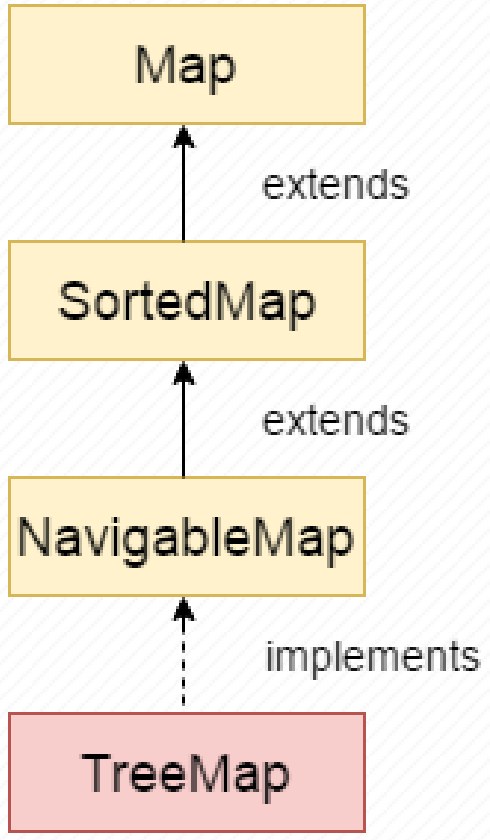
Map, HashMap, LinkedHashMap, TreeMap

Giới thiệu lớp TreeMap

- **TreeMap** lưu trữ dữ liệu dưới dạng cặp **key** và **value**.
- **TreeMap** chỉ chứa các **key** duy nhất.
- **TreeMap** không cho phép bất kỳ **key** nào là **null**, nhưng có thể có nhiều giá trị **null**.
- **TreeMap** duy trì các phần tử theo thứ tự **key** tăng dần.
- Lớp **java.util.TreeMap** được định nghĩa như sau:

```
public class TreeMap<K,V>
        extends AbstractMap<K,V>
  implements NavigableMap<K,V>, Cloneable, java.io.Serializable {
}
```

- Trong đó:
  - **K**: đây là kiểu **khóa** (key) để lưu trữ.
  - **V**: đây là kiểu **giá trị** (value) được ánh xạ.



Map, HashMap, LinkedHashMap, TreeMap

Các phương thức khởi tạo (constructor) của lớp TreeMap

Phương thức	Mô tả
TreeMap()	khởi tạo một danh sách map trống
TreeMap(Map<? extends K, ? extends V> m)	khởi tạo một map với các phần tử của map m.

Các phương thức (method) của lớp TreeMap

❑ Tương tự các phương thức đã được giới thiệu ở bài viết về **Map Interface**

Map, HashMap, LinkedHashMap, TreeMap

Ví dụ sử dụng TreeMap với kiểu dữ liệu cơ bản (Wrapper)

```
public class TreeMapExample01 {
    public static void main(String args[]) {
        // init map
        Map<Integer, String> map = new TreeMap<Integer, String>();
        map.put(1, "Basic java");
        map.put(2, "OOP");
        map.put(4, "Multi-Thread");
        map.put(3, "Collection");

        // show map using method keySet()
        for (Integer key : map.keySet()) {
            String value = map.get(key);
            System.out.println(key + " = " + value);
        }

        System.out.println("---");

        // show map using method keySet()
        for (Entry<Integer, String> entry : map.entrySet()) {
            Integer key = entry.getKey();
            String value = entry.getValue();
            System.out.println(key + " = " + value);
        }
    }
}
```

Map, HashMap, LinkedHashMap, TreeMap

Ví dụ sử dụng TreeMap với key kiểu String, value kiểu Object (Student)

```java
public class Student {
    private int id;
    private String name;

    public Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", "
                + "name=" + name + "]";
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

}
```

```java
public class TreeMapExample02 {
    public static void main(String args[]) {
        // Student's data
        Student student1 = new Student(1, "Student 1");
        Student student2 = new Student(2, "Student 2");
        Student student3 = new Student(3, "Student 3");
        Student student4 = new Student(4, "Student 4");

        // init map
        Map<Integer, Student> map = new TreeMap<Integer, Student>();
        map.put(student1.getId(), student1);
        map.put(student2.getId(), student2);
        map.put(student4.getId(), student4);
        map.put(student3.getId(), student3);

        // show map using method keySet()
        for (Integer key : map.keySet()) {
            Student value = map.get(key);
            System.out.println(key + " = " + value);
        }

        System.out.println("---");

        // show map using method keySet()
        for (Entry<Integer, Student> entry : map.entrySet()) {
            Integer key = entry.getKey();
            Student value = entry.getValue();
            System.out.println(key + " = " + value);
        }
    }
}
```

So sánh HashMap và HashSet

Giống nhau giữa HashMap và HashSet

- Cả hai cấu trúc dữ liệu không duy trì bất kỳ thứ tự cho các phần tử truyền vào.
- Đều sử dụng phương thức **hashCode()** và **equals()** để duy trì tính duy nhất của dữ liệu.
- Cả hai đều cung cấp cho hiệu suất thời gian là hằng số cho các thao tác chèn (**add/put**) và loại bỏ **(remove)**.
- Cả hai đều không đồng bộ (**non-synchronized**)

So sánh HashMap và HashSet

Khác nhau giữa HashMap và HashSet

HashSet	HashMap
HashSet cài đặt (implement) Set interface.	HashMap cài đặt (implement) Map interface.
HashSet lưu trữ dữ liệu dưới dạng các đối tượng (object).	HashMap lưu trữ dữ liệu dưới dạng cặp khóa-giá trị (key-value).
Bên trong HashSet sử dụng HashMap.	Bên trong HashMap sử dụng một mảng đối tượng Entry<K, V>.
HashSet không cho phép các phần tử trùng lặp.	HashMap không cho phép các khóa (key) trùng lặp, nhưng cho phép các giá trị (value) trùng lặp.

So sánh HashMap và HashSet

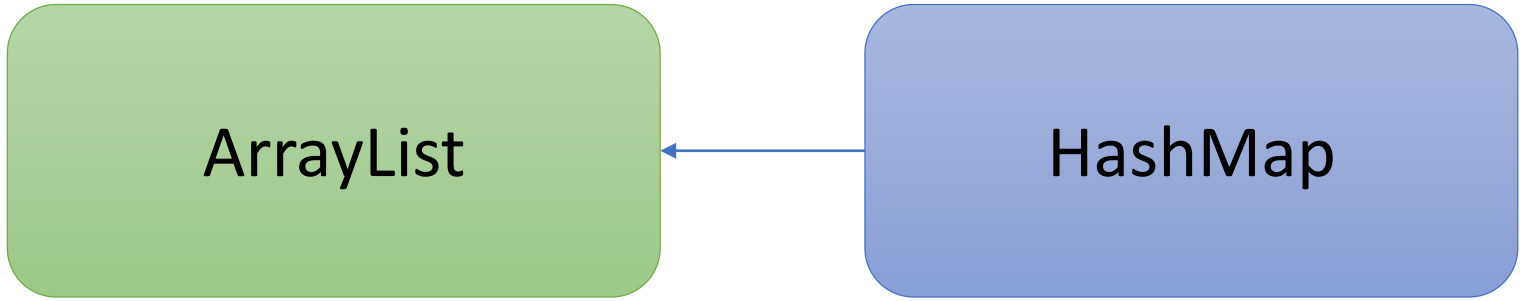
Khác nhau giữa HashMap và HashSet

HashSet	HashMap
HashSet chỉ cho phép một phần tử null.	HashMap cho phép một khóa (key) null và nhiều giá trị (value) null.
Thao tác chèn (insert/add) chỉ yêu cầu một đối tượng.	Thao tác chèn (put) yêu cầu hai đối tượng, khóa và giá trị (key-value).
HashSet hơi chậm hơn HashMap.	HashMap nhanh hơn một chút so với HashSet.
Sử dụng HashSet khi bạn cần duy nhất dữ liệu (object).	Sử dụng HashMap khi bạn cần duy nhất khóa (key).

Chuyển đổi từ HashMap sang ArrayList

Tại sao cần biết chuyển đổi từ HashMap sang ArrayList?

- **HashMap** và **ArrayList** là hai cấu trúc dữ liệu được sử dụng nhiều nhất trong java.
- Cả hai lớp **HashMap** và **ArrayList** đều kế thừa từ các hệ phân cấp khác nhau.
- **HashMap** được kế thừa từ **Map Interface** đại diện cho dữ liệu dạng các cặp **key-value**
- **ArrayList** được kế thừa từ **List Interface**, sắp xếp dữ liệu một cách tuần tự.
- Chuyển đổi **HashMap** sang **ArrayList** đã trở thành câu hỏi thường gặp trong các cuộc phỏng vấn Java vì không có phương pháp trực tiếp nào trong **HashMap** chuyển đổi **HashMap** thành **ArrayList**



Chuyển đổi từ HashMap sang ArrayList

Làm thế nào để chuyển đổi từ HashMap sang ArrayList?

- **HashMap** chứa cặp **key-value**, có 3 cách để chuyển đổi **HashMap** thành **ArrayList**:
  - Chuyển đổi các **khóa** (key) của **HashMap** thành **ArrayList**

```
// Creating a HashMap object
Map<String, String> map = new HashMap<String, String>();

// Getting Set of keys from HashMap
Set<String> keySet = map.keySet();

// Creating an ArrayList of keys by passing the keySet
List<String> listOfKeys = new ArrayList<String>(keySet);
```

Chuyển đổi từ HashMap sang ArrayList

Làm thế nào để chuyển đổi từ HashMap sang ArrayList?

- **HashMap** chứa cặp **key-value**, có 3 cách để chuyển đổi **HashMap** thành **ArrayList**:
  - Chuyển đổi các **giá trị** (value) của **HashMap** thành **ArrayList**

```
// Creating a HashMap object
Map<String, String> map = new HashMap<String, String>();

// Getting Collection of values from HashMap
Collection<String> values = map.values();

// Creating an ArrayList of values
List<String> listOfValues = new ArrayList<String>(values);
```

Chuyển đổi từ HashMap sang ArrayList

Làm thế nào để chuyển đổi từ HashMap sang ArrayList?

- **HashMap** chứa cặp **key-value**, có 3 cách để chuyển đổi **HashMap** thành **ArrayList**:
  - Chuyển đổi các cặp **khóa-giá trị** (key-value) của **HashMap** thành **ArrayList**

```
// Creating a HashMap object
Map<String, String> map = new HashMap<String, String>();

// Getting the Set of entries
Set<Entry<String, String>> entrySet = map.entrySet();

// Creating an ArrayList Of Entry objects
List<Entry<String, String>> listOfEntry = new
ArrayList<Entry<String,String>>(entrySet);
```

Chuyển đổi từ HashMap sang ArrayList

Ví dụ chuyển đổi từ HashMap sang ArrayList

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

public class HashMapToArrayList {
    public static void main(String[] args) {..
}
```

Chuyển đổi từ HashMap sang ArrayList

Ví dụ chuyển đổi từ HashMap sang ArrayList (trong hàm main)

```
// Creating a HashMap object
Map<String, String> studentPerformanceMap =
                            new HashMap<String, String>();

// Adding elements to HashMap
studentPerformanceMap.put("John Kevin", "Average");
studentPerformanceMap.put("Rakesh Sharma", "Good");
studentPerformanceMap.put("Prachi D", "Very Good");
studentPerformanceMap.put("Ivan Jose", "Very Bad");
studentPerformanceMap.put("Smith Jacob", "Very Good");
studentPerformanceMap.put("Anjali N", "Bad");

// Getting Set of keys
Set<String> keySet = studentPerformanceMap.keySet();

// Creating an ArrayList of keys
List<String> listOfKeys = new ArrayList<String>(keySet);

System.out.println("ArrayList Of Keys :");
for (String key : listOfKeys) {
    System.out.println(key);
}

System.out.println("-------------------------");
```

```
// Getting Collection of values
Collection<String> values = studentPerformanceMap.values();

// Creating an ArrayList of values
List<String> listOfValues = new ArrayList<String>(values);

System.out.println("ArrayList Of Values :");
for (String value : listOfValues) {
    System.out.println(value);
}

System.out.println("-------------------------");

// Getting the Set of entries
Set<Entry<String, String>> entrySet = studentPerformanceMap.entrySet();

// Creating an ArrayList Of Entry objects
List<Entry<String, String>> listOfEntry =
        new ArrayList<Entry<String, String>>(entrySet);
System.out.println("ArrayList of Key-Values :");
for (Entry<String, String> entry : listOfEntry) {
    System.out.println(entry.getKey() + " : " + entry.getValue());
}
```

Giới thiệu Hashtable trong Java

Giới thiệu về lớp Hashtable

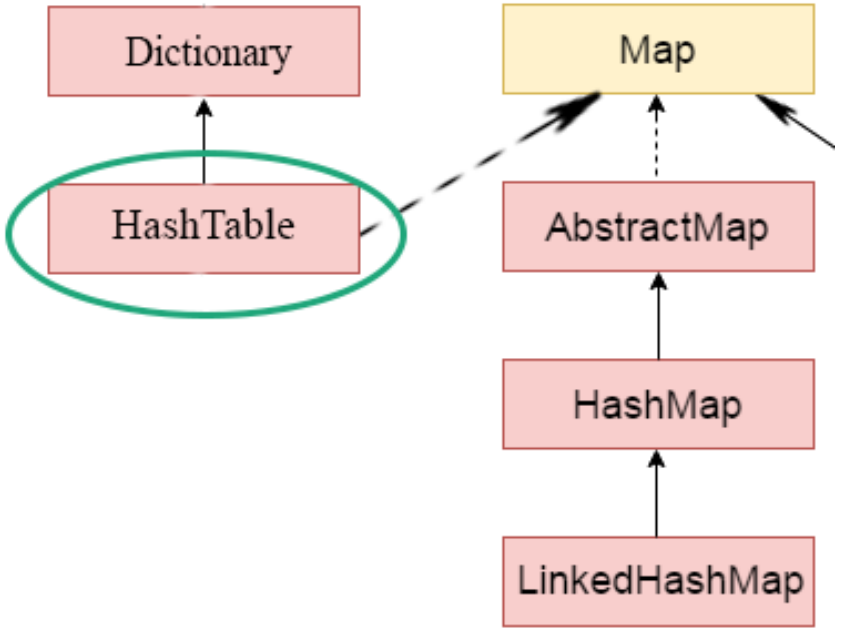
- Lớp Java **Hashtable** cài đặt (implement) một bảng **hashtable** để map **khóa** và **giá trị**.
- **Hashtable** kế thừa lớp **Dictionary** và cài đặt (implement) **Map Interface**.
- Các đặc điểm quan trọng về lớp **Hashtable** trong Java là:
  - **Hashtable** là một mảng của list. Mỗi list được biết đến như một **bucket** (vùng chứa) các phần tử. Ví trí của một **bucket** được xác định bằng việc gọi phương thức **hashcode()**. **Hashtable** cũng lưu trữ dữ liệu dưới dạng cặp **key** và **value**.
  - **Hashtable** chứa các key duy nhất.
  - **Hashtable** không thể có bất kỳ **key** hoặc giá trị nào là **null**.
  - **Hashtable** được đồng bộ (**synchronized**)

Giới thiệu Hashtable trong Java

Giới thiệu về lớp Hashtable

- Lớp **java.util.Hashtable** được định nghĩa như sau:

```
public class Hashtable<K,V>
    extends Dictionary<K,V>
  implements Map<K,V>, Cloneable, java.io.Serializable {

}
```



- Trong đó:
  - **K**: đây là kiểu **khóa** (key) để lưu trữ.
  - **V**: đây là kiểu **giá trị** (value) được ánh xạ.

Giới thiệu Hashtable trong Java

Các phương thức khởi tạo (constructor) của lớp Hashtable

Phương thức	Mô tả
Hashtable()	khởi tạo một hashtable trống, sức chứa (capacity) ban đầu mặc định là 11.
Hashtable(Map<? extends K, ? extends V> t)	khởi tạo một hashtable với các phần tử của map t.
Hashtable(int initialCapacity)	khởi tạo một hashtable trống, với sức chứa (capacity) ban đầu được xác định.

Các phương thức (method) của lớp Hashtable

❑ Tương tự các phương thức đã được giới thiệu ở bài viết về **Map Interface**

Giới thiệu Hashtable trong Java

Ví dụ sử dụng Hashtable với kiểu dữ liệu cơ bản (Wrapper)

```
public class HashTableExample01 {
    public static void main(String args[]) {
        // init Hashtable
        Hashtable<Integer, String> hashTable = new Hashtable<Integer, String>();
        hashTable.put(1, "Basic java");
        hashTable.put(2, "OOP");
        hashTable.put(3, "Collection");

        // show Hashtable using method keySet()
        for (Integer key : hashTable.keySet()) {
            String value = hashTable.get(key);
            System.out.println(key + " = " + value);
        }

        System.out.println("---");

        // show map using method keySet()
        for (Entry<Integer, String> entry : hashTable.entrySet()) {
            Integer key = entry.getKey();
            String value = entry.getValue();
            System.out.println(key + " = " + value);
        }
    }
}
```

Giới thiệu Hashtable trong Java

Ví dụ Hashtable với kiểu dữ liệu do người dùng tự định nghĩa (Wrapper)

```java
public class Student {
    private int id;
    private String name;

    public Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", "
                + "name=" + name + "]";
    }
}
```

```java
public class HashTableExample02 {
    public static void main(String args[]) {
        // Student's data
        Student student1 = new Student(1, "Student 1");
        Student student2 = new Student(2, "Student 2");
        Student student3 = new Student(3, "Student 3");

        // init Hashtable
        Hashtable<Integer, Student> hashTable = new Hashtable<Integer, Student>();
        hashTable.put(student1.getId(), student1);
        hashTable.put(student2.getId(), student2);
        hashTable.put(student3.getId(), student3);

        // show Hashtable using method keySet()
        for (Integer key : hashTable.keySet()) {
            Student value = hashTable.get(key);
            System.out.println(key + " = " + value);
        }

        System.out.println("---");

        // show Hashtable using method keySet()
        for (Entry<Integer, Student> entry : hashTable.entrySet()) {
            Integer key = entry.getKey();
            Student value = entry.getValue();
            System.out.println(key + " = " + value);
        }
    }
}
```

So sánh HashMap và Hashtable

Giống nhau giữa HashMap và Hashtable

- ❑ Cả **HashMap** và **Hashtable** đều cài đặt **Map Interface**.
- ❑ **HashMap** và **Hashtable** đều được sử dụng để lưu trữ dữ liệu ở dạng cặp **key** và **value**.
- ❑ **HashMap** và **Hashtable** đều đang sử dụng kỹ thuật băm để lưu trữ các khóa duy nhất.

So sánh HashMap và Hashtable

Khác nhau giữa HashMap và Hashtable

HashMap	Hashtable
HashMap **cho phép một key là null và nhiều giá trị null**.	Hashtable **không cho phép bất kỳ key hoặc giá trị null**.
HashMap **không đồng bộ**.	Hashtable là **đồng bộ**.
HashMap **nhanh hơn** Hashtable .	Hashtable **chậm hơn** HashMap.
HashMap được **duyệt bởi Iterator**.	Hashtable được **duyệt bởi Enumerator và Iterator**.
Iterator trong HashMap là **fail-fast**.	Enumerator trong Hashtable là **không fail-fast**.
HashMap kế thừa lớp **AbstractMap**.	Hashtable kế thừa lớp **Dictionary**.

So sánh HashMap và Hashtable

Khác nhau giữa HashMap và Hashtable

HashMap	Hashtable
Chúng ta có thể làm cho HashMap đồng bộ bằng cách gọi phương thức: **Map** m = **Collections.synchronizedMap**(hashMap);	**Hashtable** được đồng bộ nội bộ và không thể hủy đồng bộ hóa.
HashMap được ưa thích trong các ứng dụng đơn luồng (single-thread). Nếu bạn muốn sử dụng HashMap trong ứng dụng đa luồng (mulit-thread), có thể thực hiện bằng cách sử dụng phương thức **Collections.synchronizedMap()**.	Mặc dù **Hashtable** có để sử dụng trong các ứng dụng đa luồng (multi-thread), nhưng ngày nay nó **ít được sử dụng**. Bởi vì, **ConcurrentHashMap** là lựa chọn tốt hơn HashTable.

Tổng kết nội dung bài học

- Set, HashSet, LinkedHashSet, TreeSet
- So sánh HashSet, LinkedHashSet, TreeSet