Sesión 03

JAVA COLLECTION FRAMEWORK – JCF

Curso Virtual

Fundamentos de POO, Funcional y Reactivo

Ing. Aristedes Novoa

anovoa@galaxy.edu.pe

GALAXY TRAINING

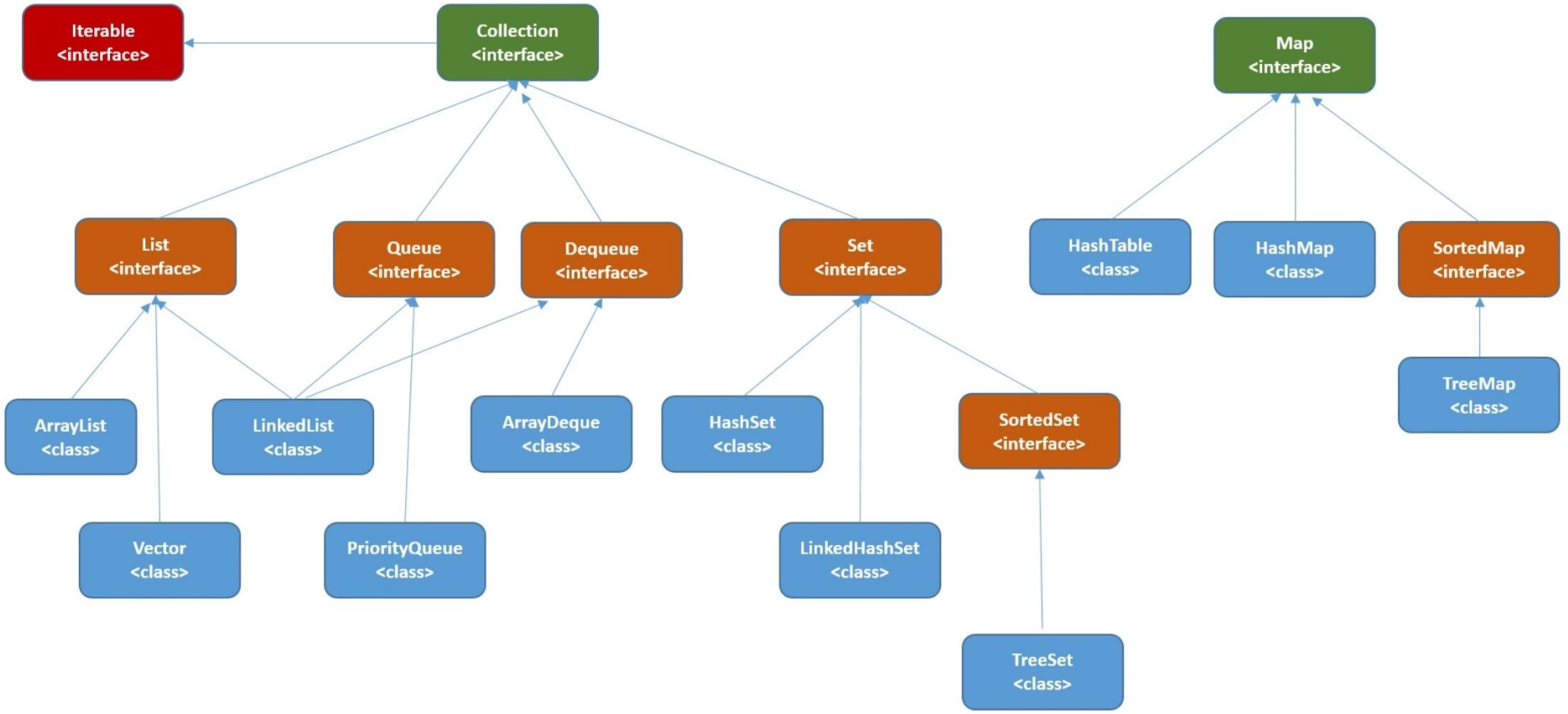
ÍNDICE


00:00 MIN
55:00 MIN
00:00 MIN
00:00 MIN
6KM 20M
designed by freepik.com


G-28

Análisis

Una colección representa un grupo de objetos, denominados elementos. Existen diversos tipos según si sus elementos están ordenados o si se permite duplicados o no. Se define con la interfaz Collection, de la cual cada tipo la hereda.



Colecciones y mapas

List: Define una sucesión de elementos que permite duplicados. **SubTipos (Clases)**: ArrayList, LinkedList y Vector.

Set: Define una colección que no puede contener elementos duplicados y para verificarlo usa los métodos **equals** y **hashCode** de la clase. **Subtipos (Clases)**: HashSet, TreeSet y LinkedHashSet.

Map: Define una colección donde asocia claves a valores, las claves no pueden ser duplicadas y solo tiene un valor asociado. **SubTipos (Clases)**: HashMap, TreeMap y LinkedHashMap.

Principales Interfaces

Interface	Duplicates Allowed?	Null Values Allowed?	Insertion order preserved?	Iterator	Data Structure
List	Yes	Yes, Multiple null values are allowed	Yes and can retrieve using index	Iterator, ListIterator	Array
Set	No	Yes but only once	No	Iterator	Underlying Map implementation
Map	Not for keys	Yes but only once for keys, can have multiple null values	No	Through keyset, value and entry set	Hashing techniques

Análisis comparativo

Interface	Hash Table	Resizable Array	Balanced Tree	Linked List	Hash Table + Linked List
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Deque		ArrayDeque		LinkedList	
Map	HashMap		TreeMap		LinkedHashMap

Interfaces e implementaciones

Es una colección que no puede contener elementos duplicados y para verificarlo usa los métodos **equals** y **hashCode** de la clase.

Principales Implementaciones (Clases)

HashSet: Es la implementación que tiene el mejor rendimiento pero no garantiza ningún orden a la hora de las iteraciones. Es importante definir el tamaño inicial ya que esto marcara el rendimiento.

TreeSet: Ordena los elementos en base a sus valores, es más lento que el HashSet.

LinkedHashSet: Ordena los elementos en orden de inserción es más lento que el HashSet.

Estas implementaciones no trabajan de manera segura (no thread-safe).

Implementaciones List

Interface List

```
List<String> arrayList = new ArrayList<>();
arrayList.add(“Elemento 1”);

List<String> linkedList = new LinkedList<>();
linkedList.add(“Elemento 1”);

List<String> vector = new Vector<>();
vector.add(“Elemento 1”);
```

Implementaciones List - Ejemplo

Es una colección que no puede contener elementos duplicados y para verificarlo usa los métodos **equals** y **hashCode** de la clase.

Principales Implementaciones (Clases)

HashSet: Es la implementación que tiene el mejor rendimiento pero no garantiza ningún orden a la hora de las iteraciones. Es importante definir el tamaño inicial ya que esto marcara el rendimiento.

TreeSet: Ordena los elementos en base a sus valores, es más lento que el HashSet.

LinkedHashSet: Ordena los elementos en orden de inserción es más lento que el HashSet.

Estas implementaciones no trabajan de manera segura (no thread-safe).

Implementaciones Set

Interface Set

```
Set<String> hashSet = new HashSet<String>();
hashSet.add(“Elemento 1”);

Set<String> treeSet = new TreeSet<>();
treeSet.add(“Elemento 1”);

Set<String> linkedHashSet = new LinkedHashSet<>();
linkedHashSet.add(“Elemento 1”);
```

Implementaciones Set - Ejemplo

Es una colección que no puede contener elementos duplicados y para verificarlo usa los métodos **equals** y **hashCode** de la clase.

Principales Implementaciones (Clases)

HashSet: Es la implementación que tiene el mejor rendimiento pero no garantiza ningún orden a la hora de las iteraciones. Es importante definir el tamaño inicial ya que esto marcara el rendimiento.

TreeSet: Ordena los elementos en base a sus valores, es más lento que el HashSet.

LinkedHashSet: Ordena los elementos en orden de inserción es más lento que el HashSet.

Estas implementaciones no trabajan de manera segura (no thread-safe).

Implementaciones Map

Interface Map

```
Map<String, String> hashMap = new HashMap<>();
hashMap.put("llave1", "Valor1");

Map<String, String> treeMap = new TreeMap<>();
treeMap.put("llave1", "Valor1");

Map<String, String> linkedHashMap = new LinkedHashMap<>();
linkedHashMap.put("llave1", "Valor1");
```

Implementaciones Map - Ejemplo

```java
Set<Integer> hashSet = new HashSet<Integer>(1_000_000);
Long startHashSetTime = System.currentTimeMillis();
for (int i = 0; i < 1_000_000; i++) {
    hashSet.add(i);
}
Long endHashSetTime = System.currentTimeMillis();
System.out.println("HashSet: " + (endHashSetTime - startHashSetTime));

Set<Integer> treeSet = new TreeSet<Integer>();
Long startTreeSetTime = System.currentTimeMillis();
for (int i = 0; i < 1_000_000; i++) {
    treeSet.add(i);
}
Long endTreeSetTime = System.currentTimeMillis();
System.out.println("TreeSet: " + (endTreeSetTime - startTreeSetTime));

Set<Integer> linkedHashSet = new LinkedHashSet<Integer>(1_000_000);
Long startLinkedHashSetTime = System.currentTimeMillis();
for (int i = 0; i < 1_000_000; i++) {
    linkedHashSet.add(i);
}
Long endLinkedHashSetTime = System.currentTimeMillis();
System.out.println("LinkedHashSet: " + (endLinkedHashSetTime - startLinkedHashSetTime));
```

Análisis de rendimiento - Set

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.TreeMap;
import java.util.TreeSet;
```

```
Set hashSet = Collections.synchronizedSet(new HashSet());
SortedSet treeSet = Collections.synchronizedSortedSet(new TreeSet());
Set linkedHashSet = Collections.synchronizedSet(new LinkedHashSet());

List arrayList = Collections.synchronizedList(new ArrayList());
List linkedList = Collections.synchronizedList(new LinkedList());

Map hashMap = Collections.synchronizedMap(new HashMap());
SortedMap treeMap = Collections.synchronizedSortedMap(new TreeMap());
Map linkedHashMap = Collections.synchronizedMap(new LinkedHashMap());
```

Colecciones – Sincronización


GALAXY
TRAINING