



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



DISIM
Dipartimento di Ingegneria
e Scienze dell'Informazione
e Matematica



Laboratorio di Algoritmi e Strutture Dati a.a. 2023/2024

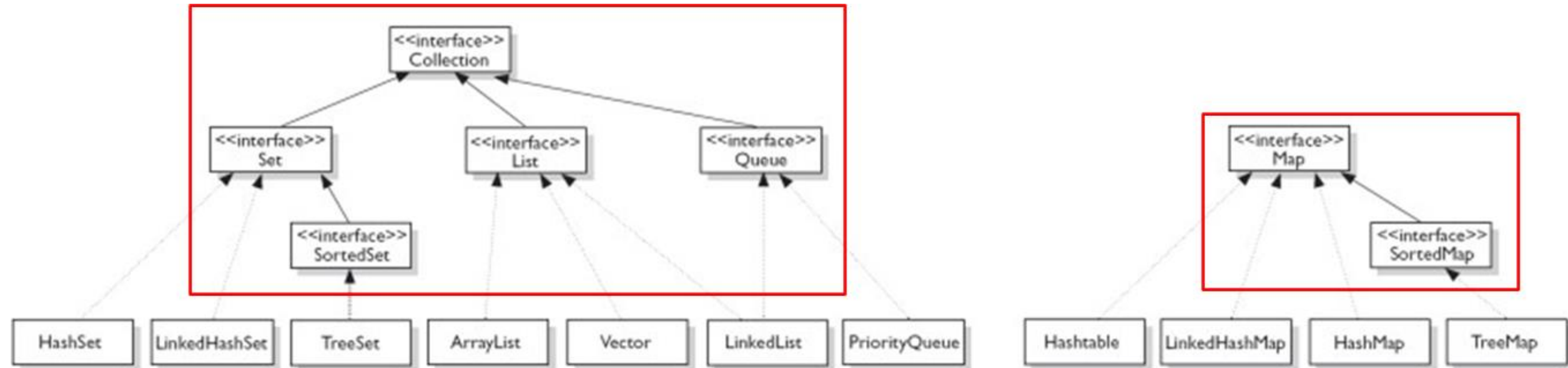
Le classi

$\text{HashMap}\langle K, V \rangle$ e $\text{TreeMap}\langle K, V \rangle$

Giovanna Melideo

Università degli Studi dell'Aquila
DISIM

JCF: Interfacce fondamentali (richiami)



ADT Mappa

- Una **mappa** è una raccolta in cui ogni elemento è una coppia (**chiave di ricerca, valore**)
 - dove la chiave (di ricerca) ha lo scopo di individuare univocamente il valore associato.
- Un “dizionario” (vocabolario) è un esempio di mappa:
 - la chiave (univoca) è la parola che viene definita
 - il valore è costituito dalla sua definizione e dall’etimologia
 - talvolta i termini **dizionario** e **mappa** vengono usati come sinonimi

ADT Mappa: esempi

- Studenti - coppie (ID-matricola, voto medio)
- Anagrafe – coppie (codice fiscale, nome)
- Rubrica telefonica - coppie (nominativo univoco, elenco dei numeri telefonici)
- Funzione discreta f - coppie (x,y) t.c. $f(x) = y$
- ...

ADT Mappa: operazioni

Le principali operazioni sulle mappe sono le seguenti:

- **inserimento** di una nuova coppia (chiave, valore)
- **aggiornamento** del valore associato ad una chiave
- **eliminazione** di una coppia (chiave, valore)
- **ricerca** del valore associato ad una chiave

L'interfaccia `Map<K, V>`

- Il JCF contiene un'interfaccia `java.util.Map<K, V>` che definisce le intestazioni dei metodi per il dato astratto "Mappa"
- L'interfaccia `java.util.Map<K, V>` specifica il tipo Mappa con chiavi di tipo `K` e valori associati di tipo `V`
- L'interfaccia `Map` non estende l'interfaccia `Collection`

Method Summary

Methods

Modifier and Type	Method and Description
void	clear () Removes all of the mappings from this map (optional operation).
boolean	containsKey (Object key) Returns true if this map contains a mapping for the specified key.
boolean	containsValue (Object value) Returns true if this map maps one or more keys to the specified value.
Set <Map.Entry<K, V>>	entrySet () Returns a Set view of the mappings contained in this map.
boolean	equals (Object o) Compares the specified object with this map for equality.
V	get (Object key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
int	hashCode () Returns the hash code value for this map.
boolean	isEmpty () Returns true if this map contains no key-value mappings.
Set <K>	keySet () Returns a Set view of the keys contained in this map.
V	put (K key, V value) Associates the specified value with the specified key in this map (optional operation).
void	putAll (Map<? extends K, ? extends V> m) Copies all of the mappings from the specified map to this map (optional operation).
V	remove (Object key) Removes the mapping for a key from this map if it is present (optional operation).
int	size () Returns the number of key-value mappings in this map.
Collection <V>	values () Returns a Collection view of the values contained in this map.

L'interfaccia Map.Entry<K,V> (cenno)

- Il metodo `Set<Map.Entry<K,V>> entrySet()` restituisce un insieme di `Map.Entry`

```
public static interface Map.Entry<K,V>
```

- «A map entry (**key-value pair**). The `Map.entrySet` method returns a collection-view of the map, whose elements are of this class. The only way to obtain a reference to a map entry is from the iterator of this collection-view. These `Map.Entry` objects are valid only for the duration of the iteration; more formally, the behavior of a map entry is undefined if the backing map has been modified after the entry was returned by the iterator, except through the `setValue` operation on the map entry.»

L'interfaccia Map.Entry<K,V>: metodi

Interface Map.Entry<K,V>

Enclosing interface: [Map<K,V>](#)

```
public static interface Map.Entry<K,V>
```

Method Summary

Methods

Modifier and Type	Method and Description
boolean	equals (Object o) Compares the specified object with this entry for equality.
K	getKey () Returns the key corresponding to this entry.
V	getValue () Returns the value corresponding to this entry.
int	hashCode () Returns the hash code value for this map entry.
V	setValue (V value) Replaces the value corresponding to this entry with the specified value (optional operation).

La classe `HashMap<K, V>`

- Implementa l'interfaccia `Map<K, V>`

Constructor Summary

Constructors

Constructor and Description

`HashMap()`

Constructs an empty `HashMap` with the default initial capacity (16) and the default load factor (0.75).

`HashMap(int initialCapacity)`

Constructs an empty `HashMap` with the specified initial capacity and the default load factor (0.75).

`HashMap(int initialCapacity, float loadFactor)`

Constructs an empty `HashMap` with the specified initial capacity and load factor.

`HashMap(Map<? extends K, ? extends V> m)`

Constructs a new `HashMap` with the same mappings as the specified `Map`.

L'interfaccia `SortedMap<K, V>`

- `SortedMap` estende e specializza `Map` analogamente a quanto `SortedSet` fa con `Set`
- L'ordinamento è quello naturale degli elementi (espresso dalla loro `compareTo`) o quello fornito da un apposito `Comparator` all'atto della creazione del `SortedSet`
- L'interfaccia di accesso aggiunge metodi che sfruttano l'esistenza di un ordinamento totale fra gli elementi:
 - **`firstKey`** e **`lastKey`** restituiscono la prima/ultima chiave nell'ordine
 - **`headMap`**, **`subMap`** e **`tailMap`** restituiscono le sottotabelle con le sole entry le cui chiavi sono minori/comprese/maggiori di quella data

L'interfaccia `SortedMap<K, V>`: metodi

Method Summary

Methods

Modifier and Type	Method and Description
<code>Comparator<? super K></code>	<code>comparator()</code> Returns the comparator used to order the keys in this map, or <code>null</code> if this map uses the natural ordering of its keys.
<code>Set<Map.Entry<K, V>></code>	<code>entrySet()</code> Returns a Set view of the mappings contained in this map.
<code>K</code>	<code>firstKey()</code> Returns the first (lowest) key currently in this map.
<code>SortedMap<K, V></code>	<code>headMap(K toKey)</code> Returns a view of the portion of this map whose keys are strictly less than <code>toKey</code> .
<code>Set<K></code>	<code>keySet()</code> Returns a Set view of the keys contained in this map.
<code>K</code>	<code>lastKey()</code> Returns the last (highest) key currently in this map.
<code>SortedMap<K, V></code>	<code>subMap(K fromKey, K toKey)</code> Returns a view of the portion of this map whose keys range from <code>fromKey</code> , inclusive, to <code>toKey</code> , exclusive.
<code>SortedMap<K, V></code>	<code>tailMap(K fromKey)</code> Returns a view of the portion of this map whose keys are greater than or equal to <code>fromKey</code> .
<code>Collection<V></code>	<code>values()</code> Returns a Collection view of the values contained in this map.

Methods inherited from interface `java.util.Map`

`clear`, `containsKey`, `containsValue`, `equals`, `get`, `hashCode`, `isEmpty`, `put`, `putAll`, `remove`, `size`

La classe `TreeMap<K, V>`

- La classe alloca una mappa in un albero red-black, ordinato in base alle chiavi
- Implementa l'interfaccia `SortedMap<K, V>`

Constructors

Constructor and Description

`TreeMap()`

Constructs a new, empty tree map, using the natural ordering of its keys.

`TreeMap(Comparator<? super K> comparator)`

Constructs a new, empty tree map, ordered according to the given comparator.

`TreeMap(Map<? extends K, ? extends V> m)`

Constructs a new tree map containing the same mappings as the given map, ordered according to the *natural ordering* of its keys.

`TreeMap(SortedMap<K, ? extends V> m)`

Constructs a new tree map containing the same mappings and using the same ordering as the specified sorted map.

Esempio

- Uso di HashMap e TreeMap
 - Rif. Province.java

```
public class Capoluoghi {  
    private static Map<String, String> province = ;  
    ...  
    province.put("L'Aquila", "Abruzzo");  
  
}
```

Il metodo `Map.entrySet()` : osservazioni

- Una mappa non è iterabile
 - A partire dalla versione 8 di java l'interfaccia `java.util.Map` prevede un metodo `forEach()` che consente di specificare una azione da eseguire su tutti gli elementi della mappa.
- Il metodo `Map.entrySet()` consente di **iterare le coppie di una mappa** nell'ambito del set di entries associato all'oggetto `HashMap/TreeMap`
- Schema generale (data un'istanza "map" di tipo `Map<K,V>`):
 - ```
for (Map.Entry<K, V> entry: map.entrySet()) { ... }
```
- Se si modifica la mappa mentre è in corso un'iterazione sul corrispondente set, l'esito dell'iterazione è indefinito (tranne quando le modifiche avvengono mediante l'operazione `setValue` su una map entry restituita dall'iteratore).

# Esercizio 1

- Suppose we are given the name and division number for each employee in a company. There are no duplicate names. We would like to store this information alphabetically, by name.
  - How should this be done?
  - TreeMap? TreeSet? Comparable? Comparator?



# Esercizio 1: Esempio

|                   |    |
|-------------------|----|
| Misino,John       | 8  |
| Nguyen,Viet       | 14 |
| Panchenko,Eric    | 6  |
| Dunn,Michael      | 6  |
| Deusenbery,Amanda | 14 |
| Taoubina,Xenia    | 6  |

We want these elements stored in the following order:

|                   |    |
|-------------------|----|
| Deusenbery,Amanda | 14 |
| Dunn,Michael      | 6  |
| Misino,John       | 8  |
| Nguyen,Viet       | 14 |
| Panchenko,Eric    | 6  |
| Taoubina,Xenia    | 6  |

# Esercizio 1 (continua)

- We can use a **TreeMap** object.
- The keys will be names, of type String, and the values will be division numbers of type Integer.
- The String class implements the Comparable interface, so the elements will be stored in alphabetical order of keys.
  - **Rif. CompanyMap.java**
- Remark: If we use a HashMap object, the elements will not be stored in alphabetical order of keys

# Esercizio 2

- Re-do Programming Exercise 1, but now the ordering should be by increasing division numbers, and within each division number, by alphabetical order of names.

# Esercizio 2: Esempio

|                   |    |
|-------------------|----|
| Misino,John       | 8  |
| Nguyen,Viet       | 14 |
| Panchenko,Eric    | 6  |
| Dunn,Michael      | 6  |
| Deusenbery,Amanda | 14 |
| Taubina,Xenia     | 6  |

We want these elements stored in the following order:

|                   |    |
|-------------------|----|
| Dunn,Michael      | 6  |
| Panchenko,Eric    | 6  |
| Taubina,Xenia     | 6  |
| Misino,John       | 8  |
| Deusenbery,Amanda | 14 |
| Nguyen,Viet       | 14 |

## Esercizio 2 (continua)

- We can use a `TreeSet` object.
- The elements will be objects in an `Employee` class, which implements the `Comparable` interface.
  - **Rif. `Employee.java`**



UNIVERSITÀ  
DEGLI STUDI  
DELL'AQUILA



DISIM  
Dipartimento di Ingegneria  
e Scienze dell'Informazione  
e Matematica



# Domande?

**Giovanna Melideo**  
Università degli Studi dell'Aquila  
DISIM