Programmazione Orientata agli Oggetti

Collezioni:

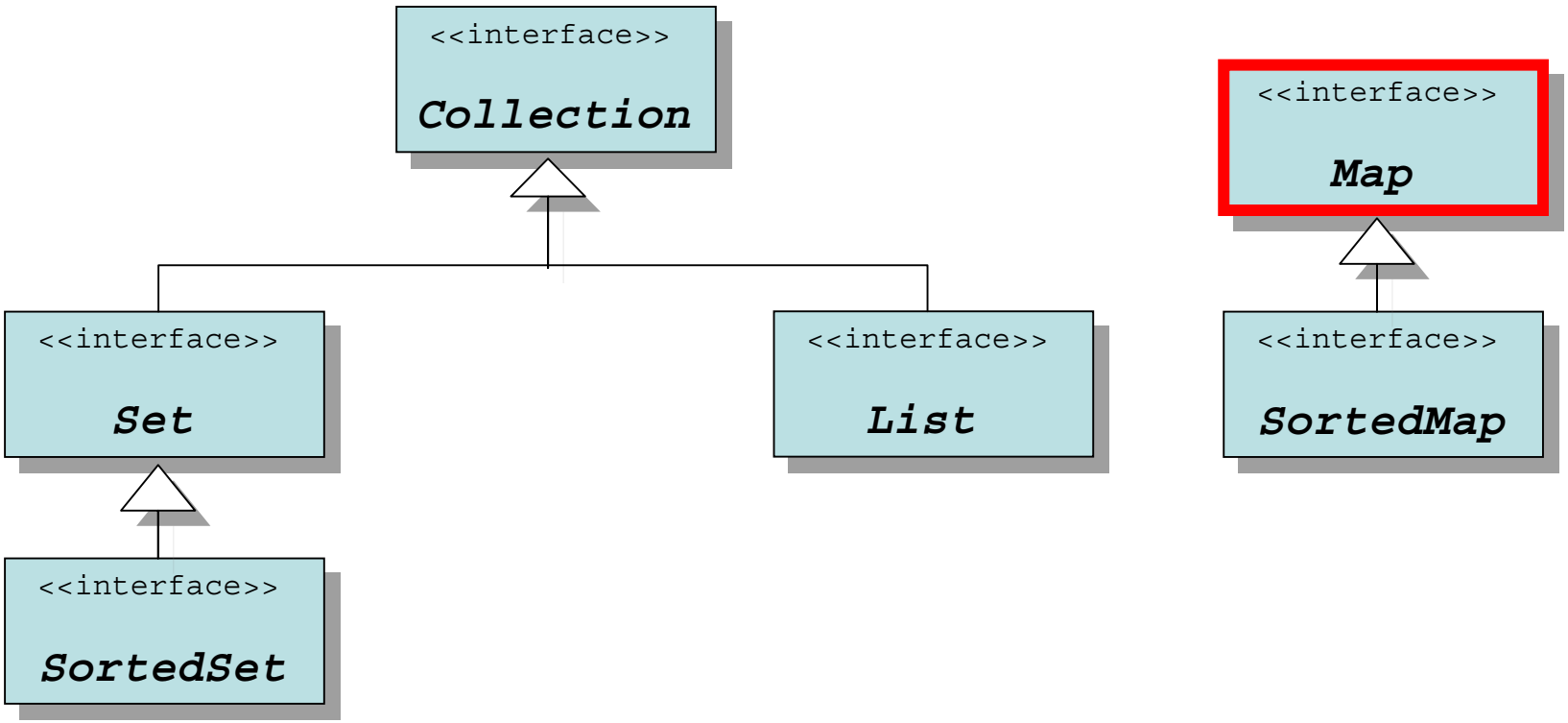
Mappe generiche

Concetti introdotti

- Mappe
- Interface **`Map<K,V>`**
- Operazioni tipiche sulle mappe
- Implementazioni di **`Map<K,V>`**

Collezioni: Interface

- Le principali interface del package **`java.util`**



- Per ognuna di queste interface il package offre diverse implementazioni

Mappe

- Una mappa (o dizionario, o array associativo) è una collezione di coppie chiave-valore (entry)
- Le chiavi, in quanto tali, sono uniche
  - In una mappa non possono esistere due chiavi eguali
- Ad ogni chiave può essere associato un solo valore
  - Ma il valore può essere una collezione

- Esempi:
  - rubrica telefonica: a ciascuna persona (chiave) è associato un numero di telefono (valore)
  - Indice analitico di un libro: ad ogni voce (chiave) è associato un insieme di numeri di pagine (valore)

Mappe

- Operazioni tipiche su una mappa sono:
  - Inserimento di una coppia chiave-valore
    - Inserisci nella rubrica: "Paolo Rossi" -> 0288821212
  - Richiesta del valore associato ad una chiave
    - Dammi il numero di telefono di "Elena Rondi"
  - Aggiornamento di un valore
    - Cambia il numero di telefono di "Paolo Rossi" in 063232122
  - Rimozione di una coppia chiave-valore
    - Rimuovi "Fabio Capello" dalla mia rubrica
  - Verifica di esistenza di una chiave
    - C'è "Carla Bruni" nella mia rubrica?
  - Richiesta dell'insieme delle chiavi
    - Dammi i nomi di tutte le persone della mia rubrica

I metodi base di `Map<K,V>`

- **`V put(K chiave, V valore);`**
  inserisce la coppia chiave-valore nella mappa;
  - se la chiave esiste già, allora il valore viene aggiornato e il metodo ritorna il valore vecchio
  - se la chiave non esiste viene inserita una nuova coppia, e il metodo ritorna **`null`**
- **`V get(Object chiave);`**
  restituisce il valore associato alla chiave, **`null`** se la chiave non esiste nella mappa
- **`V remove(Object chiave);`**
  rimuove il valore associato alla chiave e lo ritorna
- **`boolean containsKey(Object chiave);`**
  verifica se la chiave è presente nella mappa

perché non `K` ?

Metodi bulk e viste di `Map<K,V>`

- **`void putAll(Map<K,V> mappa)`**
  inserisce nella mappa tutte le coppie della mappa passata come parametro
- **`void clear()`**

  Elimina tutte le coppie dalla mappa
- **`Set<K> keySet()`**
  Restituisce in un insieme tutte le chiavi
- **`Collection<V> values()`**

  Restituisce in una collezione tutti i valori

Mappe: implementazioni

- Nelle librerie del Collection Framework abbiamo diverse implementazioni di **`Map<K,V>`**:
  - **`HashMap<K,V>`**
  - **`TreeMap<K,V>`**
- Nelle due implementazioni l'unicità delle chiavi viene garantita da meccanismi analoghi a quelli delle implementazioni di **`Set`**. In particolare:
  - l'unicità delle chiavi in **`HashMap<K,V>`** è garantita attraverso i metodi **`hashCode()`** e **`equals()`** delle chiavi
  - l'unicità delle chiavi in **`TreeMap<K,V>`** è garantita attraverso il metodo **`compareTo()`** delle chiavi (e le chiavi devono implementare **`Comparable<K>`**) oppure tramite il metodo **`compare()`** di un comparatore esterno **`Comparator<K>`**

Esercizio

- Scrivere una batteria di metodi di test per comprendere a fondo la semantica di tutti i metodi citati nelle slide precedenti con riferimento alle due implementazioni di **`Map<K,V>`**
  - **`HashMap<K,V>`**
  - **`TreeMap<K,V>`**

Esempio d'uso

```
import java.util.*;
public class Rubrica {
    private Map<String,Integer> rubrica = new HashMap<String,Integer>();
   public void inserisci(String nome, Integer numero) {
        this.rubrica.put(nome, numero);
   }
   public void rimuovi(String nome) {
        this.rubrica.remove(nome);
   }
   public Set<String> nomiInRubrica() {
        return this.rubrica.keySet();
   }
   public Integer dammiIlNumeroDi(String nome) {
        return this.rubrica.get(nome);
   }
   public Integer aggiornaNumero(String nome, Integer numero) {
        return this.rubrica.put(nome, numero);
   }
   public void stampa(){
        System.out.println("----------------------------");
        System.out.println("Rubrica");
        Set<String> nomi = nomiInRubrica();
        for(String s : nomi)
                System.out.println(s+": "+this.rubrica.get(s));
        System.out.println("----------------------------");
   }…
```

Esempio d'uso (cont)

```
…  public static void main(String[] args) {
        Rubrica r = new Rubrica();
        String s1 = new String("Paolo"), s2 = new String("Fabio");
        String s3 = new String("Anna"),  s4 = new String("Carla");

        r.inserisci(s1, 390112461); // inserisco Paolo->390112461 in rubrica
        r.inserisci(s2, 390108361); // inserisco Fabio->390108361 in rubrica
        r.inserisci(s3, 39062888); // inserisco Anna->39062888 in rubrica
        r.stampa(); // stampo la rubrica
        // verifico se ho il numero di Carla
        Integer numeroCercato = r.dammiIlNumeroDi(s4);
        if (numeroCercato == null)
                System.out.println("Il numero di "+s4+" non esiste");
        else
                System.out.println("Il numero di "+s4+" è "+numeroCercato);

        r.rimuovi(s2);  // tolgo i dati relativi a Fabio dalla rubrica

        // cambio il numero di Paolo (e mi faccio stampare il numero vecchio)
        Integer nuovoNumero = 39066777;
        Integer vecchioNumero = r.aggiornaNumero(s1,nuovoNumero);
        System.out.println("Aggiornato il numero di "+s1+
                           " da "+vecchioNumero+" a "+nuovoNumero);
        r.stampa();  // stampo la rubrica
    }
}
```

Esercizio

- Con riferimento al codice dell'esercizio **`Precipitazioni`**, nella classe **`VisualizzatoreIstogrammaGrafico`**, scrivere il codice del metodo **`Map costruisciIstogramma().`**
  - Questo metodo analizza la matrice delle precipitazioni e costruisce, restituendola, una mappa, che ha per chiavi gli interi che compaiono nella matrice, e per le occorrenze delle chiavi nella matrice (cioè, il numero di volte che ciascun valore della chiave compare nella matrice).
  - Ad esempio, data la matrice:

3	3	1	2	7
3	2	2	4	6
4	4	2	1	3
3	4	1	6	1
2	3	2	500	8

Verrebbe costruita una mappa con i seguenti valori (l'ordine delle chiavi è irrilevante): `8→1, 500→1, 4→4, 6→2, 7→1, 2→6, 3→6, 1→4`

Infatti: il numero `8` compare `1` volta nella matrice, il numero `500` compare `1` volta nella matrice, il numero `4` compare `4` volte nella matrice, il numero `6` compare `2` volte nella matrice, etc.