

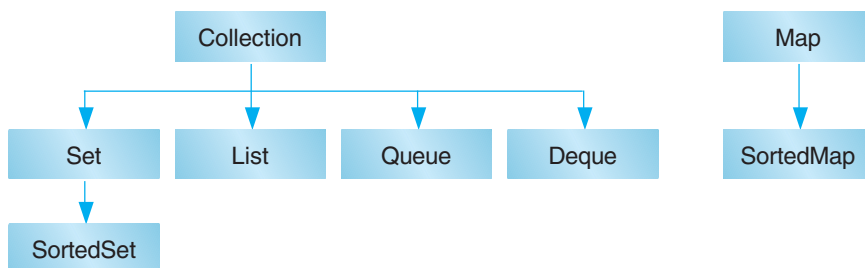
Il Java Collections Framework per gestire gli insiemi di dati

Il **Collections Framework** è tecnicamente una gerarchia di classi, implementata all'interno delle librerie di Java, che offre ai programmatori gli strumenti per memorizzare e manipolare gli insiemi di dati.

Con il termine **Collection** si fa riferimento ad un singolo oggetto, descritto da una classe, che al suo interno raggruppa un insieme di elementi. Oltre a raggruppare i dati, la *Collection* definisce un insieme di operazioni per l'aggiunta, la cancellazione e la ricerca sui dati stessi.

Con il termine **Framework** si vuole evidenziare che le *Collections* sono organizzate in una struttura che, tramite un insieme di classi e metodi, facilita l'elaborazione delle collezioni di dati da parte dei programmatori.

Le classi del *Collections Framework* sono organizzate nella seguente gerarchia di classi:



Oltre all'insieme generico *Collection*, il framework definisce i seguenti elementi:

- **Set**: un insieme che non contiene valori duplicati;
- **List**: un insieme ordinato di elementi, come per esempio la classe *Vector*;
- **Map**: un insieme in cui gli elementi sono memorizzati come coppie chiave/valore;
- **Queue/Deque**: un insieme di elementi in attesa di essere processati.

Indipendentemente da quale tipo di *Collection* si sta utilizzando, si possono richiamare i seguenti metodi condivisi.

Metodo	Descrizione
size()	Restituisce il numero di elementi dell'insieme.
isEmpty()	Restituisce il valore <i>true</i> se l'insieme è vuoto, altrimenti <i>false</i> .
contains(oggetto)	Restituisce il valore <i>true</i> se l'oggetto è presente nell'insieme, altrimenti <i>false</i> .
add(oggetto)	Aggiunge un elemento all'insieme.
remove(oggetto)	Toglie un elemento dall'insieme.
iterator()	Restituisce un <i>iterator</i> per eseguire un ciclo sugli elementi dell'insieme.
addAll(collection)	Aggiunge all'insieme tutti gli oggetti di un'altra collection.
removeAll(collection)	Rimuove dall'insieme tutti gli oggetti di un'altra collection.
clear()	Svuota l'insieme.

In aggiunta ai metodi precedenti, ogni *Collection* specifica può definire ulteriori metodi. Nel seguito vengono analizzate le tre *Collection* principali: *Set*, *List* e *Map*.

La collection Set

I *Set* consentono di raggruppare un insieme di elementi con la caratteristica che l'insieme non può contenere valori duplicati. I metodi utilizzabili su un *Set* sono tutti quelli precedentemente descritti e condivisi con le *Collection*.

Le classi concrete, sottoclassi della classe *Set*, che possono essere utilizzate per creare una *Collection* di tipo *Set* sono: **HashSet**, **TreeSet** e **LinkedHashSet**.

Per esempio, la dichiarazione di un insieme di colori che non può contenere duplicati viene fatta con la seguente istruzione:

```
Set colori = new HashSet();
```

La stessa dichiarazione può essere migliorata con l'indicazione del tipo specifico degli elementi contenuti nell'insieme, scrivendo:

```
Set<String> colori= new HashSet<String>();
```

Si noti che l'oggetto *colori* è stato dichiarato con il nome della sopraclasse *Set* e la sua istanza è stata creata con la sottoclasse *HashSet*.

Le seguenti righe di codice dovrebbero aggiungere tre colori all'insieme, ma in realtà ne aggiungono solo due, in quanto l'insieme non accetta i valori duplicati:

```
colori.add("rosso");  
colori.add("azzurro");  
colori.add("rosso");
```

La terza istruzione non genera errore e non modifica il numero di elementi dell'insieme. Se si controlla il numero di elementi con l'istruzione

```
colori.size();
```

si ottiene come risultato 2.

La collection List

Le *List* consentono di raggruppare un insieme di elementi in cui ogni elemento è caratterizzato da una sua posizione. Oltre ai metodi ereditati dalle *Collection*, le *List* definiscono un insieme di metodi in cui l'accesso agli elementi dell'insieme è gestito da un **indice**, che indica la posizione dell'elemento.

I metodi principali di *List* sono:

Metodo	Descrizione
get(indice)	Restituisce l'oggetto che si trova nella posizione indicata dall'indice.
set(indice, oggetto)	Inserisce un oggetto nella posizione indicata dall'indice, sovrascrivendo l'elemento precedente.
remove(indice)	Rimuove l'oggetto che si trova nella posizione indicata dall'indice.
indexOf(oggetto)	Restituisce l'indice a cui si trova l'oggetto.
subList(inizio, fine)	Restituisce una lista che contiene gli elementi compresi tra l'indice <i>inizio</i> e l'indice <i>fine</i> .

Le classi concrete, sottoclassi della classe *List*, che possono essere utilizzate per creare una *Collection* di tipo *List* sono: **Vector**, **ArrayList** e **LinkedList**.

La collection Map

Le *Map* consentono di raggruppare un insieme di coppie di elementi, in cui il primo elemento è la **chiave** (*key*) e il secondo il **valore** (*value*). All'interno della *Map* non possono esserci due coppie che condividono la stessa *key*, infatti alla stessa chiave può corrispondere soltanto un *value*. Le *Map* realizzano il concetto di **mapping**, in cui ad un certo oggetto (chiave) se ne fa corrispondere un altro (valore).

Sebbene la chiave e il valore possono riferirsi a oggetti di qualsiasi classe, solitamente le chiavi sono identificate da stringhe, oggetti di classe *String*, mentre i valori sono gli oggetti che si vogliono collegare alle chiavi. Per esempio, volendo gestire una rubrica telefonica con una *Map*, la chiave potrebbe essere la stringa del nome mentre il valore potrebbe essere l'oggetto contenente tutte le informazioni sul contatto.

I metodi principali di *Map* sono:

Metodo	Descrizione
put(chiave, valore)	Inserisce nella <i>Collection</i> la nuova coppia chiave/valore.
get(chiave)	Restituisce l'oggetto dell'insieme che è associato alla chiave.
remove(chiave)	Rimuove la coppia associata alla chiave.
containsKey(chiave)	Restituisce il valore <i>true</i> se la chiave è presente nella <i>Collection</i> , altrimenti <i>false</i> .
containsValue(valore)	Restituisce il valore <i>true</i> se il valore è presente nella <i>Collection</i> , altrimenti <i>false</i> .

Le classi concrete, sottoclassi della classe *Map*, che possono essere utilizzate per creare una *Collection* di tipo *Map* sono: **HashTable**, **HashMap**, **TreeMap** e **LinkedHashMap**.

Un esempio di utilizzo della classe *HashTable* è riportato nel *Materiale on line* "Struttura di dati dinamica per gestire le Hash Table".