Il framework Java Collections

Introduzione al Framework

a.a. 2022-2023
Programmazione II – ITPS – Università degli studi di Bari Aldo Moro
ultima modifica: 11 dicembre 2022

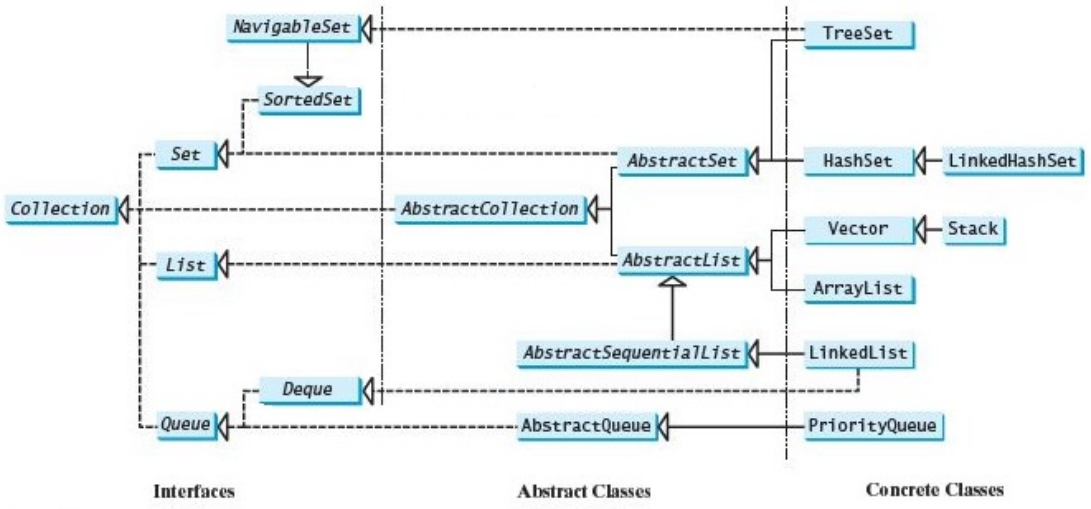
Introduzione: definizione di Collection

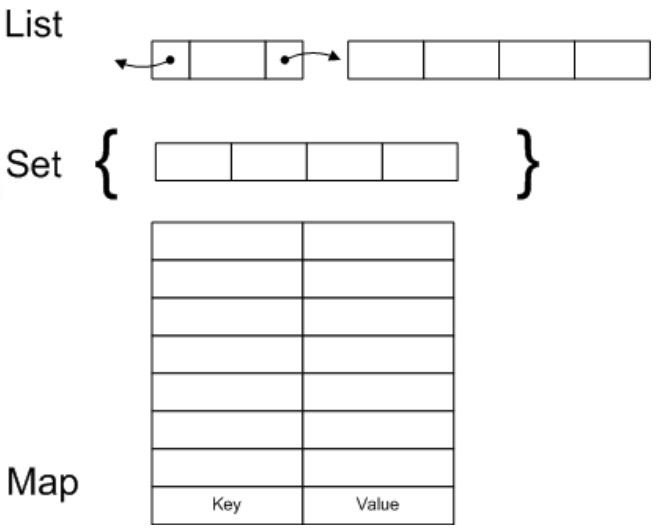
Una Collection (collezione, talvolta chiamata contenitore), è un oggetto che raggruppa più elementi in un'unica unità

- Le collezioni sono utilizzate per memorizzare, recuperare, manipolare e comunicare dati aggregati.
- In genere, rappresentano elementi di dati che formano un gruppo naturale, come:
  - una mano di poker (un insieme di carte)
  - una cartella di posta (un insieme di lettere)
  - un elenco telefonico (una mappatura di nomi e numeri di telefono)

La gerarchia del Framework Java Collections (FJC)

- Architettura unificata per rappresentare e manipolare le collezioni
  - formata da un insieme di interfacce e di classi che implementano tali interfacce
    - gli oggetti di una collezione sono chiamati elementi
  - supporta diversi tipi di collezioni: ***set, list, queue, deque, map***
    - Map, ulteriore collezione di dati seppur non facente parte del framework Collection.
  - Le interfacce e le classi della libreria si trovano nel package *java.util*





FJC: cosa comprende

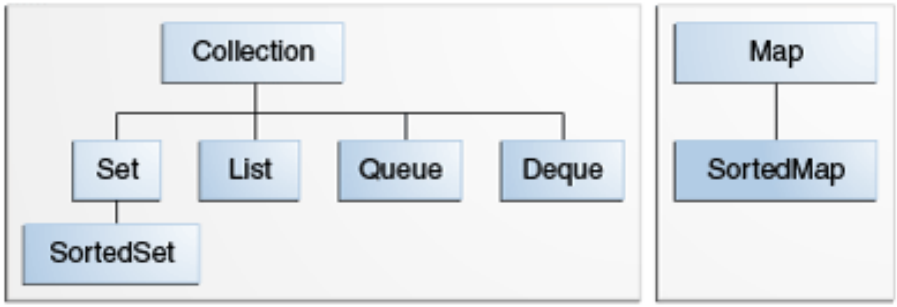
- Interfacce
  - rappresentano vari tipi di collezioni di uso comune.
- Implementazioni
  - sono classi concrete che implementano le interfacce basiche del framework Java Collections, utilizzando strutture dati efficienti.
- Algoritmi
  - funzioni che realizzano algoritmi di uso comune, quali algoritmi di ricerca e di ordinamento su oggetti che implementano le interfacce del framework Java Collections.

FJC: perché usarlo

- Generalità
  - permette di modificare l'implementazione di una collezione senza modificare le classi che si servono delle collezioni.
- Interoperabilità
  - permette di utilizzare (e farsi utilizzare da) codice realizzato indipendentemente dal proprio.
- Efficienza
  - le classi che realizzano le collezioni sono ottimizzate per avere prestazioni particolarmente buone.

Le interfacce nel FJC

- Le interfacce di base delle collezioni incapsulano diversi tipi di collezioni, mostrati nella figura a destra
- Queste interfacce consentono di manipolare le collezioni indipendentemente dai dettagli della loro rappresentazione.
- le interfacce di base sono organizzate in una gerarchia.



Le interfacce di base del FJC (1/2)

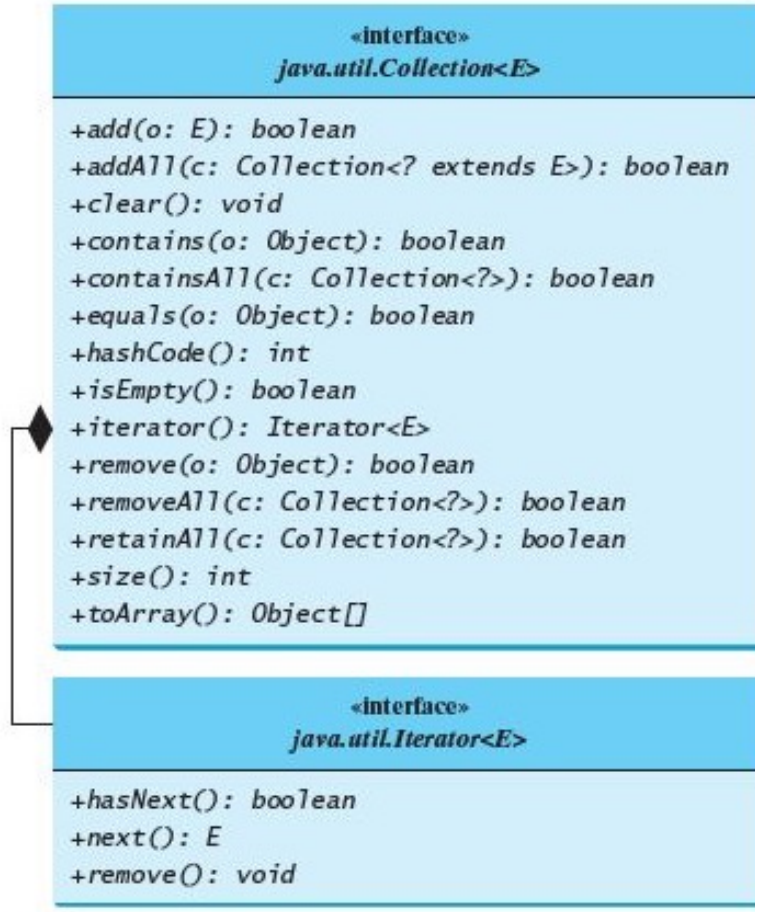
- Collection - la radice della gerarchia delle collezioni; rappresenta un gruppo di oggetti noti come elementi.
  - L'interfaccia Collection è il minimo comune denominatore che tutte le collezioni implementano e viene utilizzata per passare le collezioni e per manipolarle quando si desidera la massima generalità. Alcuni tipi di collezioni consentono la duplicazione degli elementi, altri no. Alcune sono ordinate e altre non ordinate. La piattaforma Java non fornisce implementazioni dirette di questa interfaccia, ma fornisce implementazioni di sottointerfacce più specifiche, come Set e List
- Set - una collezione che non può contenere elementi duplicati.
  - Modella l'astrazione matematica degli insiemi e viene utilizzata per rappresentare insiemi, come le carte che compongono una mano di poker, i corsi che compongono il programma di uno studente o i processi in esecuzione su una macchina
- List - un insieme ordinato (talvolta chiamato sequenza/lista) che può contenere duplicati
  - L'utilizzatore di una lista ha generalmente un controllo preciso sulla posizione di ciascun elemento nell'elenco e può accedere agli elementi in base al loro indice intero (posizione).

Le interfacce di base del FJC (2/2)

- Queue - una collezione utilizzata per contenere più elementi prima dell'elaborazione. Oltre alle operazioni di base delle Collection, una coda fornisce ulteriori operazioni di inserimento, estrazione e ispezione.
  - Le code di solito, ma non necessariamente, ordinano gli elementi in modo FIFO (first-in, first-out). Fanno eccezione le code di priorità, che ordinano gli elementi in base a un comparatore fornito o all'ordine naturale degli elementi. Qualunque sia l'ordine utilizzato, la testa della coda è l'elemento che verrebbe rimosso da una chiamata di rimozione o poll. In una coda FIFO, tutti i nuovi elementi vengono inseriti in coda alla coda. Altri tipi di coda possono utilizzare regole di posizionamento diverse. Ogni implementazione di coda deve specificare le sue proprietà di ordinamento.
- Deque - una coda usata sia come FIFO (first-in, first-out) che come LIFO (last-in, first-out).
  - tutti i nuovi elementi possono essere inseriti, recuperati e rimossi da entrambe le estremità.
  - Oltre alle operazioni di base, fornisce ulteriori operazioni di inserimento, estrazione e ispezione.
- Map - Un oggetto che associa ad una chiave un valore. Una Map non può contenere chiavi duplicate; ogni chiave può mappare al massimo un valore.

L'interfaccia Collection

Interfaccia di base per la manipolazione di una collezione di oggetti



Basic: Operazioni di base quali inserimento, cancellazione, ricerca di un elemento nella collezione

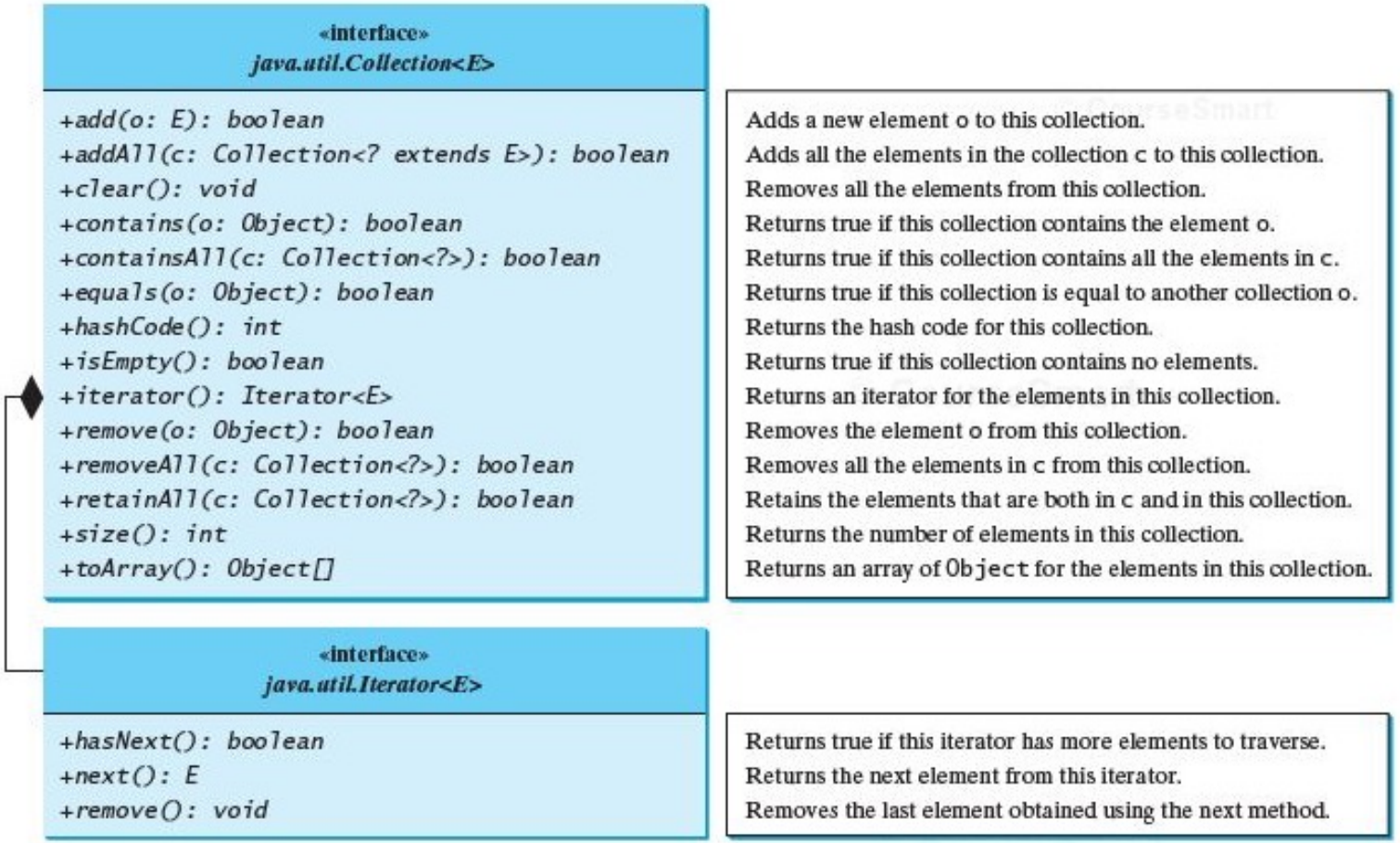
Bulk: Operazioni che lavorano su intere collezioni quali l'inserimento, la cancellazione, la ricerca di collezioni di elementi

Array: Operazioni per trasformare il contenuto della collezione in un array.

Optional: Operazioni che lanciano UnsupportedOperationException se non supportate da una data implementazione dell'interfaccia.

L'interfaccia Collection

Interfaccia di base per la manipolazione di una collezione di oggetti



Come esplorare una collection?

- Dalla JDK 1.8 in poi esistono tre modi per esplorare i contenuti di una collezione Java:
  1. utilizzando le operazioni di aggregazione
  2. con il costrutto for-each
  3. utilizzando gli iteratori

Esplorare una collezione mediante operazioni di aggregazione

- Per iterare su una collezione è possibile ottenere uno stream ed eseguire operazioni aggregate su di esso.
- Le operazioni di aggregazione sono spesso utilizzate insieme alle espressioni lambda per rendere la programmazione più espressiva, utilizzando meno righe di codice.

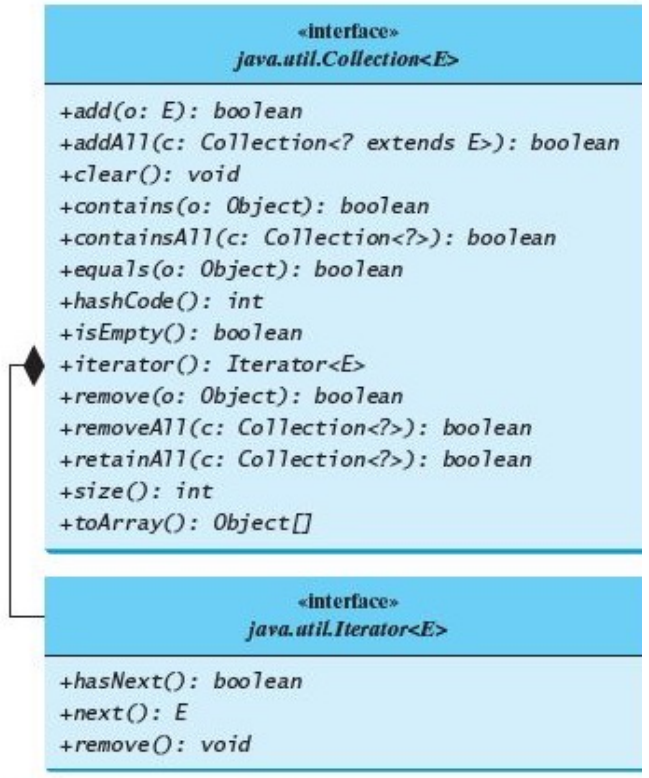
ESEMPIO:

- Il codice seguente esegue un'iterazione sequenziale su un insieme di forme stampando il nome degli oggetti il cui colore è rosso:

```
myShapesCollection.stream()
.filter(e -> e.getColor() == Color.RED)
.forEach(e -> System.out.println(e.getName()));
```

Esplorare una collezione mediante Iterator

- oggetto che rappresenta il cursore con cui esplorare sequenzialmente la collezione alla quale è associato.
- sempre associato ad un oggetto collezione.
- per funzionare deve essere a conoscenza degli aspetti più nascosti di una classe, quindi la sua realizzazione dipende interamente dalla classe collezione concreta che implementa la collezione.



Esplorare una collezione mediante for-each

- Il costrutto for-each consente di attraversare in modo conciso un insieme o un array utilizzando un ciclo.

```
for (Object o : collection)
    o.doSomething();
```

ESEMPIO: Il codice seguente utilizza il costrutto for-each per stampare il nome degli oggetti Shape il cui colore è rosso:

```
for (Shape e : myShapesCollection)
    if (e.getColor() == Color.RED)
        System.out.println(e.getName()));
```

Scansione: for-each ed Iteratori

Il costrutto `for-each`:

```
for (Object o : collection)
  System.out.println(o);
```

L'interfaccia `Iterator`:

```
public interface Iterator<E> {
  boolean hasNext();
  E next();
  void remove();
}
```

for-each *vs* Iterator

- Si usa l'iteratore al posto di for-each quando:
  - si deve rimuovere il corrente oggetto (esempio filtraggio)

```
static void filter(Collection c) {
  for (Iterator i = c.iterator(); i.hasNext();) {
    if (!cond(i.next()))
    i.remove();
}
```

  - si deve rimpiazzare un elemento nella lista o nell'array durante l'attraversamento

```
ListIterator<E> extends Iterator<E>
```

  - occorre iterare su molteplici collezioni simultaneamente