

Fondamenti di Informatica T-1

Iteratori

Tutor:
Angelo Feraudo
angelo.feraudo2@unibo.it

a.a. 2017/2018

Interfaccia <<Collection>>: Ripasso

- L'interfaccia <<Collection>> introduce l'idea di *collezione di elementi*
- È estremamente generica: non si fanno ipotesi sulla natura della collezione
- Metodi forniti:
 - add → aggiunge un elemento
 - remove → rimuove un elemento
 - isEmpty → verifica se la collezione è vuota
 - size → indica la cardinalità della collezione
 - add(index, element) → aggiunge un elemento nella posizione indicata
 - get(index) → restituisce l'elemento della posizione indicata
 - set(index, element) → sostituisce l'elemento nella posizione indicata con l'elemento indicato
 - **iterator()** → ottiene un oggetto di tipo iteratore

Iteratore

Mezzo per **iterare** su una collezione di elementi

- l'iteratore svolge per la collezione un ruolo analogo a quello di una variabile di ciclo in un array
- garantisce che ogni elemento venga considerato una e una sola volta, indipendentemente dal tipo di collezione e da come essa sia realizzata

Iteratore: metodi

- **hasNext()** → ritorna **true** se e solo se esiste un altro elemento all'interno della collezione
- **next()** → restituisce il prossimo elemento della collezione
- **remove()** → rimuove dalla collezione l'ultimo elemento che è stato restituito dalla chiamata di **next()**

Esempio

Esempio:

```
List <Tipo> myList = new ArrayList<Tipo>();  
//Come ottengo l'iteratore?  
Iterator <Tipo> iteratore = myList.iterator();  
while(iteratore.hasNext())  
{  
Tipo temp = iteratore.next(); //Ho ottenuto  
l'elemento della lista  
//eseguo altre operazioni...  
}
```

Esempio: metodo aggiungiLibro()

```
public boolean aggiungiLibro(Libro l)
{
    boolean aggiunto = false, trovato = false;
    Iterator <Libro> iteratore = libri.iterator();
    while(iteratore.hasNext() && !trovato) {
        Libro temp = iteratore.next();
        if(temp.getISBN().equalsIgnoreCase(l.getISBN())) {
            trovato = true;
            if(l.getCosto() < temp.getCosto()) {
                iteratore.remove();
                libri.add(l); //in questo caso viene aggiunto in fondo alla lista
                aggiunto = true;
            }
        }
    }
    if(!trovato) {
        libri.add(l);
        aggiunto = true;
    }
    return aggiunto;
}
```

Esercizio (1)

Un'**azienda sanitaria** desidera creare un archivio elettronico per la gestione dei propri medici di base e delle liste dei relativi pazienti.

Si realizzi:

- una classe **Medico** avente un nominativo con i relativi metodi getter/setter
- una classe **Paziente** avente un numero (intero) di tessera sanitaria ed un riferimento al proprio medico curante, con i relativi metodi getter/setter
- una classe **AziendaSanitaria** avente una lista dei propri **medici** di base e una lista dei relativi **pazienti**

Esercizio (2)

La classe **AziendaSanitaria** ha i seguenti metodi:

- **aggiungiPaziente** che prende in ingresso un numero di tessera sanitaria e un medico curante, crea l'istanza della classe **Paziente** e inserisce il paziente nella lista di pazienti. Il metodo deve restituire un valore booleano che attesti l'avvenuto inserimento del paziente. L'inserimento andrà a buon fine solo se il medico curante è presente nella lista dei medici relativi all'azienda sanitaria. Inoltre verificare che non ci siano pazienti duplicati (controllo sul numero di tessera sanitaria).

Esercizio (3)

- **aggiungiMedico** che prende in ingresso un medico e lo inserisce nella lista di medici. Il metodo deve restituire un valore booleano che attesti l'avvenuto inserimento del medico. Verificare che non ci siano medici duplicati.
- **listaMedico** restituisce una lista con tutti e soli i pazienti che hanno il medico passato come parametro
- **statMedico** restituisce il medico avente il maggior numero di pazienti

N.B. Si faccia utilizzo degli iteratori.

Esercizio (4)

Si realizzi infine un main di prova che verifichi il funzionamento di tutti i metodi della classe **AziendaSanitaria**