Ingegneria del Software
Esercitazione

20 Novembre 2023

Davide Yi Xian Hu

Email: davideyi.hu@polimi.it

POLITECNICO
MILANO 1863

Esercizio 1 / Functional ArrayList

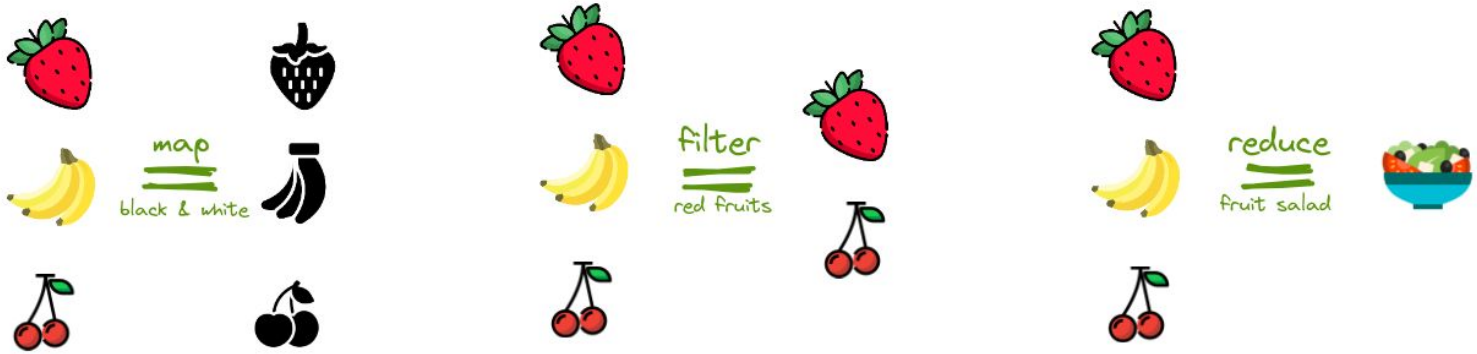
Recap teorico.

- 👉 **Data una lista di elementi di tipo T, definita come List<T>**
- 👉 **Map**: applica una funzione *f(T)* ➜ *S* che trasforma tutti gli elementi della lista di T in elementi S.
- 👉 **Filter**: filtra gli elementi della lista T utilizzando una funzione *f(T)* ➜ *true/false*. Se la funzione ritorna true per un elemento T, allora viene preservato.
- 👉 **Reduce**: applica una funzione *f(T,T)* ➜ *T* che combina una coppia di elementi della lista e ritorna un oggetto dello stesso tipo.

Esercizio 1 / Functional ArrayList

Estendere ArrayList con i metodi functional.

- 👉 **Map**: applica una funzione a tutti gli elementi.
- 👉 **Filter**: filtra gli elementi dato un predicato.
- 👉 **Reduce**: riduce l'array a un solo elemento.



Esercizio 1 / Functional ArrayList

👉 **Data una lista di elementi di tipo Triangle, definita come List<Triangle>**

👉 **Map**: definire una funzione che trasforma i triangoli in quadrati che abbiano la stessa area e applicare la funzione usando map.

👉 **Filter**: definire una funzione che ritorni true se il perimetro del triangolo e' dispari e applicare la funzione usando filter.

👉 **Reduce**: definire una funzione che somma l'area di due triangoli e applicare la funzione per trovare l'area totale dei triangoli nella lista.

Esercizio 2 / Takewhile

Estendere ArrayList con il metodo takewhile che prende un predicato e ritorna un nuovo ArrayList contente tutti gli elementi fino al primo elemento che viola il predicato.

```
var a = MyArrayList<String>();
a.add(“abc”);
a.add(“d”);
a.add(“efg”);
a.takewhile((i)->i.length() > 1); // [“abc”]
a.takewhile((i)->i.length() > 0); // [“abc”, “d”, “efg”]
a.takewhile((i)->i.length() > 5); // []
```

Info / Functional Stream +

Dato uno stream di elementi di tipo T, definito come Stream<T>

- 👉 **ForEach**: applica una funzione *f(T) ➜ ?* a tutti gli elementi dello stream.
- 👉 **FindFirst**: trova il primo elemento per cui un predicato *f(T) ➜ boolean* e' vero.
- 👉 **Sorted**: ordina lo stream secondo una funzione *f(T,T) ➜ int*.
- 👉 **Distinct**: filtra lo stream e ritorna gli elementi univoci.
- 👉 **AllMatch, AnyMatch, NoneMatch**: ritorna true se tutti, almeno uno o nessun elemento rispettano un predicato *f(T) ➜ boolean*.
- 👉 **Iterate**: genera uno stream a partire da un elemento base e una funzione *f(T) ➜ T* che genera l'elemento successivo.

Esercizio 3 / Multithreaded Account

👉 Tre persone hanno un fondo comune. Una persona, il produttore, ha il compito di depositare i soldi per tutti alla fine del mese ma non può spenderli mentre le altre due, i consumatori, posso prelevare. Il conto non può andare in rosso.

👉 Si implementino in Java 3 componenti che simulino il fondo comune e gli accessi. Il produttore deposita max 200€ ogni 5 secondi, mentre il primo consumatore può prelevare max 300€ ogni secondo, mentre l'altro max 200€ ogni tre secondi.

Esercizio 4 / Multithreaded Pot

👉 Dei campeggiatori mangiano servendosi da una pentola comune.

👉 La pentola può contenere P porzioni di cibo (con P non necessariamente maggiore del numero di campeggiatori). Ogni campeggiatore mangia una porzione per volta. Quando la pentola si svuota (e solo allora), il cuoco provvede a riempirla con nuove P porzioni.

👉 Implementare in Java per le sole parti relative alla sincronizzazione tra i processi, i programmi che realizzano i comportamenti dei campeggiatori e del cuoco e la gestione della pentola.