~ by katsucurry 

Collection Framework

wrappers dei tipi primitivi

Definizione un wrapper di tipo primitivo é un oggetto che incapsula un attributo di tipo primitivo.
↳ stesso comportamento del tipo primitivo
↳ può essere usato come Object
↳ il compilatore trasforma in automatico wrapper ↔ tipo primitivo

Tipo Primitivo	Tipo Wrapper
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Osservazione
- In caso dobbiamo confrontare 2 wrapper, é meglio utilizzare equals invece di == perché il compilatore effettua operazioni che non li confronta come tipi primitivi.
- Meglio fare un cast esplicito perché potrebbero essere null e generare un'eccezione.

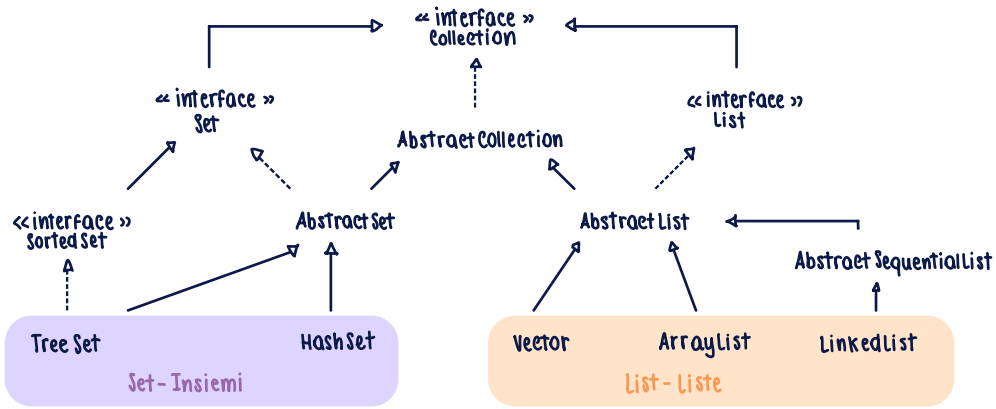
esempio
```
String s = "12";
Integer in = new Integer(s);   ⟶ stampa 12

System.out.println(Integer.parseInt(s)); ⟶ stampa 12
String hexValue = Integer.toHexString(in); ⟶ stampa c
```

Collezioni

Definizione Una collezione (o container) é un elemento container di oggetti denominati elementi.
↳ Java Collection Framework = insieme di interfacce e di classi per l'implementazione di collezioni.

- interfaccia Collection
  - ↳ interfaccia Set
    - ↳ interfaccia SortedSet
  - ↳ interfaccia List



Sintassi Collection < Nome classe > Nome Collection

esempio Collection <Object> C1;
Collection <Animale> C2;

metodi

add (E,e)	aggiunge e alla collezione
clear()	svuota la collezione
contains (Object o)	ritorna true se esiste un elemento e t.c. e.equals(o)
remove (Object o)	rimuove l'elemento o dalla collezione
Size()	ritorna il numero degli elementi nella collezione
toArray()	ritorna un array con gli elementi della collezione

Interfaccia Set

Definizione Il Set è una Collection che non ammette elementi duplicati (individuato da equals).
↳ estende Collection

Sintassi Set < Nome classe > Nome Set

esempio Set <Object> S1;
Set <Animale> S2

metodi

add (E,e)	aggiunge e alla collezione se non è già presente (equals)
clear()	svuota la collezione
contains (Object o)	ritorna true se esiste un elemento e t.c. e.equals(o)
remove (Object o)	rimuove l'elemento o dalla collezione
Size()	ritorna il numero degli elementi nella collezione
toArray()	ritorna un array con gli elementi della collezione

Interfaccia SortedSet

Definizione Il SortedSet è un Set totalmente ordinato

Sintassi `SortedSet<E>`

esempio `SortedSet<Object> S1;`
`SortedSet<Animali> S2;`

metodi

`add(E,e)`	aggiunge e alla collezione se non è già presente (equals)
`clear()`	svuota la collezione
`contains(Object o)`	ritorna true se esiste un elemento e t.c. e.equals(o)
`remove(Object o)`	rimuove l'elemento o dalla collezione
`Size()`	ritorna il numero degli elementi nella collezione
`toArray()`	ritorna un array con gli elementi della collezione
`E first()`	ritorna il primo elemento dell'insieme
`SortedSet<E> headSet(E toEl)`	ritorna una vista dell'insieme contenente tutti gli elementi strettamente minori di toEl
`E last()`	
`SortedSet<E> subSet(E fromEl, E toEl)`	
`SortedSet<E> tailSet(E fromEl)`	

Interfaccia List

Definizione La lista è una collection con elementi ordinati e ciascuna posizione è individuata da un indice.

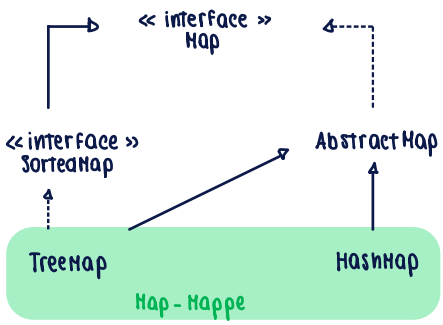
Sintassi `List<NomeClasse> NomeList`

esempio `List<Object> L1;`
`List<Animale> L2;`

metodi

`add(E,e)`	aggiunge e in fondo alla collezione di elementi
`add(int index, E e)`	aggiunge e alla posizione index
`clear()`	svuota la collezione
`contains(Object o)`	ritorna true se esiste un elemento e t.c. e.equals(o)
`remove(Object o)`	rimuove il primo elemento e della collezione (equals)
`remove(int index)`	rimuove l'elemento alla posizione index
`Size()`	ritorna il numero degli elementi nella collezione
`toArray()`	ritorna un array con gli elementi della collezione
`E get(int index)`	ritorna l'elemento alla posizione index
`int indexOf(Object o)`	ritorna la posizione della prima occorenza di un elemento e.equals(o) oppure -1
`int lastIndexOf(Object o)`	ritorna la posizione dell'ultima occorenza di un elemento e.equals(o) oppure -1
`set(int index, E element)`	sostituisce l'elemento della collezione alla posizione index con element
`List<E> sublist(int fromIndex, int toIndex)`	ritorna la porzione della lista che va da fromIndex a toIndex (escluso).

Interfaccia Map



Definizione La mappa é una struttura dati dove ad ogni elemento viene associata una chiave univoca

---

Vector: array ridimensionabile. Implementa List senza ulteriori metodi. Incluso nel java.util

HashCode usa hashcode. 2 elementi in un HashTable sono uguali se hanno stesso hashcode che é lo stesso se i parametri sono uguali