

# Containers

java.util.\*

## Interface Collection

### ArrayList

add(obj) / addAll(Collection)

size()

contains(obj)

retainAll → partea 75 din  
(Coll) intersectia

remove(obj) → (prima gasita)  
clear()

obj[] = toArray(obj[])

equals

se foloseste  
equals pt.  
determinare

ArrayList < Book > ---

## Interface List

get(index)  
set(index, obj)  
lastIndexOf(obj)

~~get(i)~~

## Interface Collection

Iterator iterator() {  
    boolean hasNext()  
    Obj next()  
    void remove()  
}

Iterator it = bookList.iterator();

```
while (it.hasNext())  
(Book) it.next().setPages(10);
```

~~it = new Iterator();~~

## • Linked List

(implements List)

↗ eff for inserting / removing

↘ eff for random access

+ multe alte tipuri de liste

Set

HashSet  
TreeSet

## • HashSet

add (never equals)

păstrează  
doar  
valori  
unice

(verifică egalitatea de hashCode)

$x.equals(y) \rightarrow x.hashCode == y.hashCode$

```
public int hashCode()  
{  
    return myValue;  
}
```

• **Linked Hash Set**  $\rightarrow$  păstreaza ordinea de inserție a elementelor

• **Tree Set**  $\rightarrow$  ordonat „natural”

String.compareTo(Stn)  $\begin{matrix} \swarrow -1 \\ \searrow 1 \end{matrix}$   
(ord. lexicografică)

MyClass implements Comparable < MyClass >  
{  
  public int compareTo(MyClass o);  
}

---

implements Map  
• **HashMap** < Obj, Obj >  
  put(key, val)    key    value

put(key, val == null? 1: val + 1)

val = map.get(key)

~~< int, int >~~  
< Integer, ... >

• Tree Map  $\rightarrow$  sortate

---

Variable number of arguments

void add ( MyStringElement... elem)

set.add ( m<sub>1</sub>, m<sub>2</sub>, m<sub>3</sub> )

---

Clone (din clasa Object)

class implements Cloneable

$\rightarrow$  doar atunci este permisă metoda

interfață de marcat

! atentie la Aliasing

$\rightarrow$  se clonează valorile componentelor  
(se copiază referințele)

Shallow Cloning

## Deep Cloning

- > se copiază din valurile (frecare)
- > se clonează frecare referință