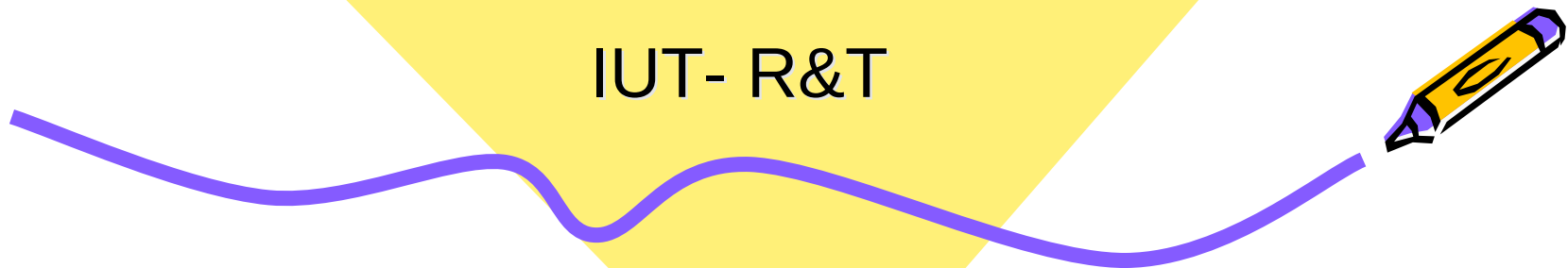


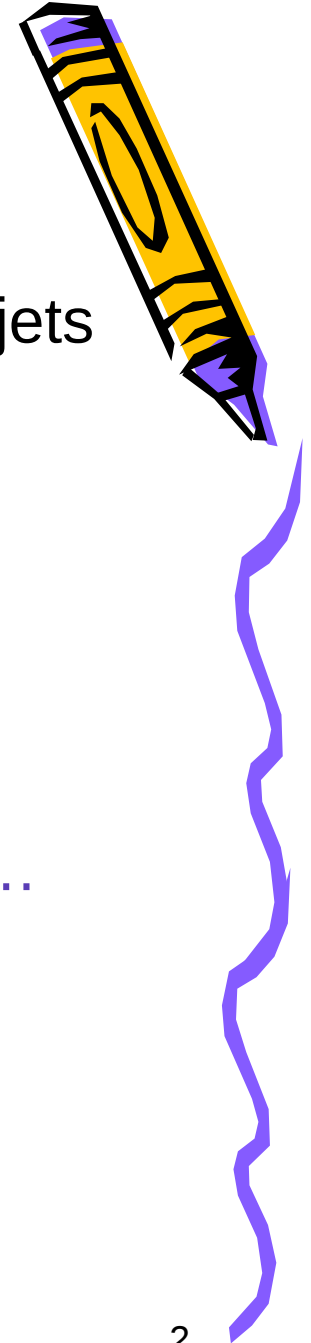


Collections

Karima Boudaoud
IUT- R&T



Collections



○ Définition

- une collection est un objet qui contient d'autres objets
- exemple: un tableau est une collection

○ Classes & interfaces

- ✓ **AbstractCollection, ArrayList, Arrays, Collections, HashSet, LinkedList, TreeSet, Vector...**
- ✓ **List, Map, Set, SortedMap, SortedSet...**

○ Package

- Ces classes et interfaces se trouvent dans le paquetage **java.util**



Collections



○ Problème

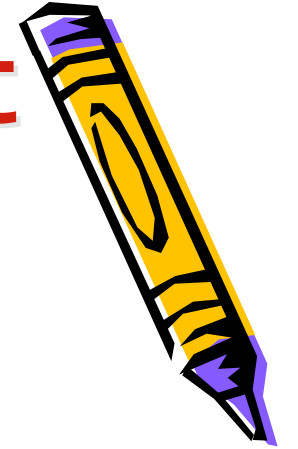
- les tableaux ne répondent pas toujours à tous les besoins
- quand un nombre inconnu d'objets sont à stocker...
 - ...on pourrait créer un très grand tableau, mais
 - ✓ ce n'est pas très « propre »
 - ✓ ce n'est jamais assez grand !



Peter Sander

ESSI-Université de Nice Sophia
Antipolis

java.util.ArrayList

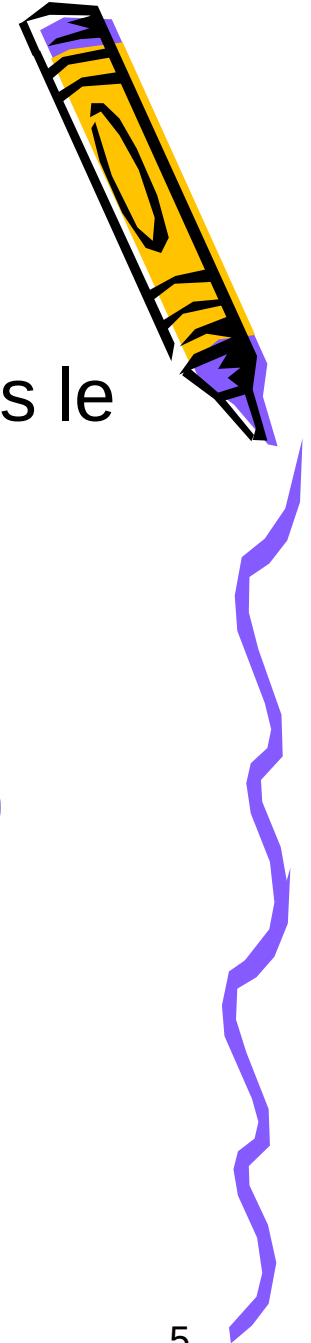


○ Solution

- la classe `java.util.ArrayList`
- c'est la classe la plus utilisée
- un `ArrayList` se comporte comme un tableau
 - ❑ il contient plusieurs objets (de la classe `Object` uniquement)
 - ✓ ne peut contenir des types primitifs
 - ❑ il accède à ses éléments à l'aide d'un index
 - ❑ il grossit automatiquement
 - ✓ quand plus de place pour contenir de nouveaux objets
 - ❑ il existe des méthodes pour ajouter ou enlever un élément



ArrayList

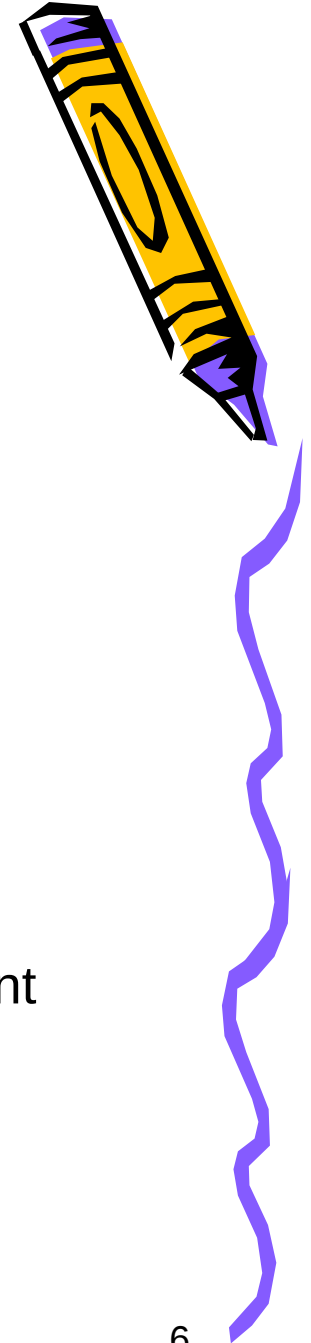


○ Création d'un ArrayList

- il est possible d'indiquer la taille initiale dans le constructeur
- Il y a 2 constructeurs :
 - ✓ `ArrayList()`
 - ✓ `ArrayList(int initialCapacity)`



ArrayList

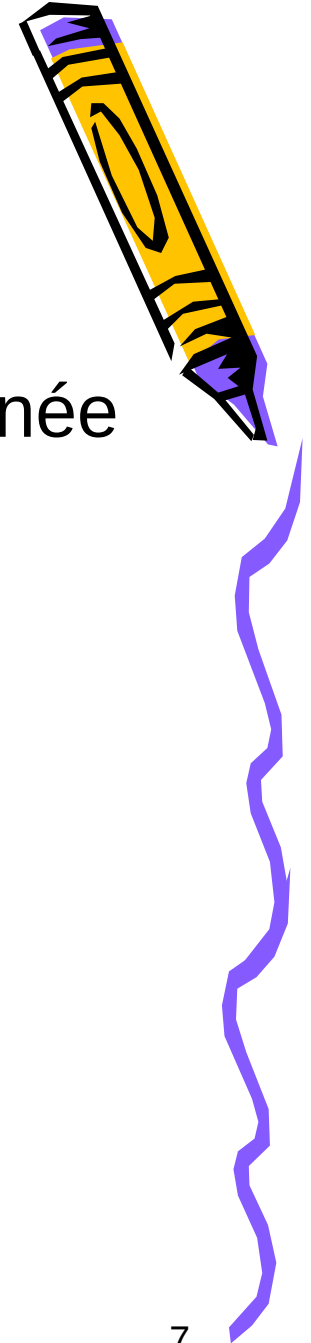


○ Modification d'éléments

- Il y a deux manières d'ajouter un élément
 - ❑ à la fin d'un **ArrayList** avec la méthode
boolean add(Object newElement)
 - ❑ à une position donnée
void add(int index, Object newElement)
throws IndexOutOfBoundsException
- ✓ le paramètre **index** indique où insérer le nouvel élément
- ✓ si position incorrecte, une exception est levée



ArrayList



○ Modification d'éléments

- pour remplacer un objet à une position donnée

Object set(int index, Object newElement)

throws IndexOutOfBoundsException

- cette méthode fonctionne comme

void add(int index, Object newElement)

- ✓ sauf que l'élément à la position ***index*** est remplacé



ArrayList



○ Accès aux Éléments

- pour accéder à un élément
 - ✓ il n'y a pas d'indexation comme pour les tableaux
 - ✓ il faut utiliser la méthode spécialisée

Object get(int index) throws IndexOutOfBoundsException

- exemple :

```
ArrayList aList = new ArrayList();  
aList.add(new PacMan());  
aList[0].display(); // interdit  
aList.get(0).display(); // ok
```



ArrayList



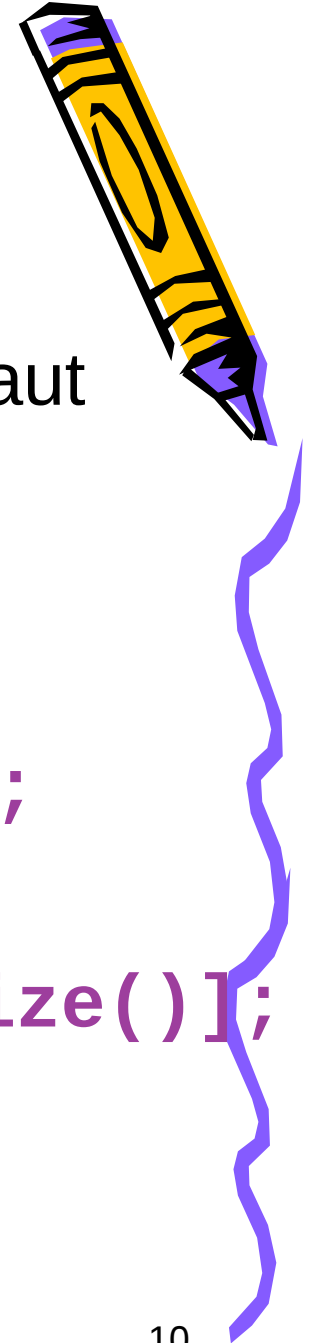
○ Accès aux Éléments

- pour tester le contenu, il existe la méthode **boolean isEmpty()**
- pour connaître le nombre d'éléments dans la liste, il faut utiliser la méthode : **int size()**
- exemple :

```
if (!aList.isEmpty()) {  
    for (int i=0; i<aList.size(); i++){  
        System.out.println(aList.get(i));  
    }  
}
```



ArrayList



○ Recopie

- pour recopier une liste dans un tableau, il faut utiliser la méthode

Object[] toArray()

- exemple :

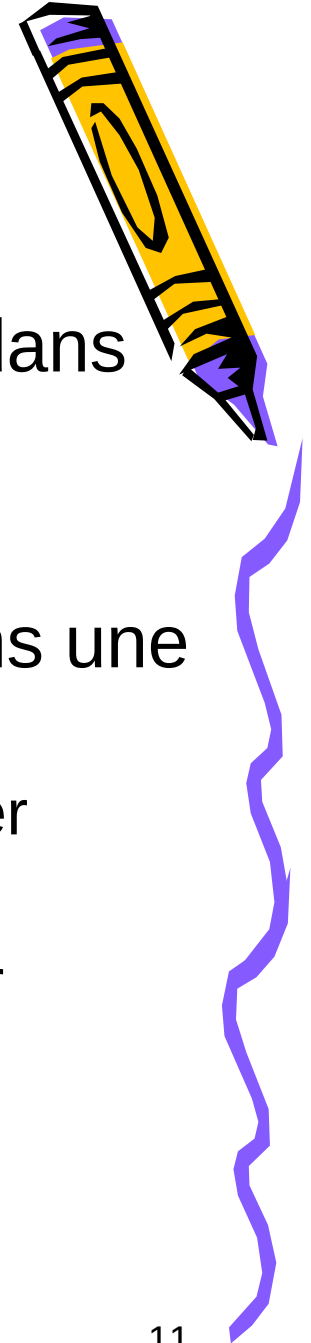
```
ArrayList aList = new ArrayList();
```

...

```
Object[] tab = new Object[aList.size()];  
tab = aList.toArray();
```



ArrayList



○ Recherche d'éléments

- pour savoir si un objet est présent ou non dans une liste, il faut utiliser la méthode

boolean contains(Object obj)

- pour connaître la position d'un élément dans une liste, on peut utiliser deux méthodes

- ✓ pour avoir la *première occurrence*, il faut utiliser

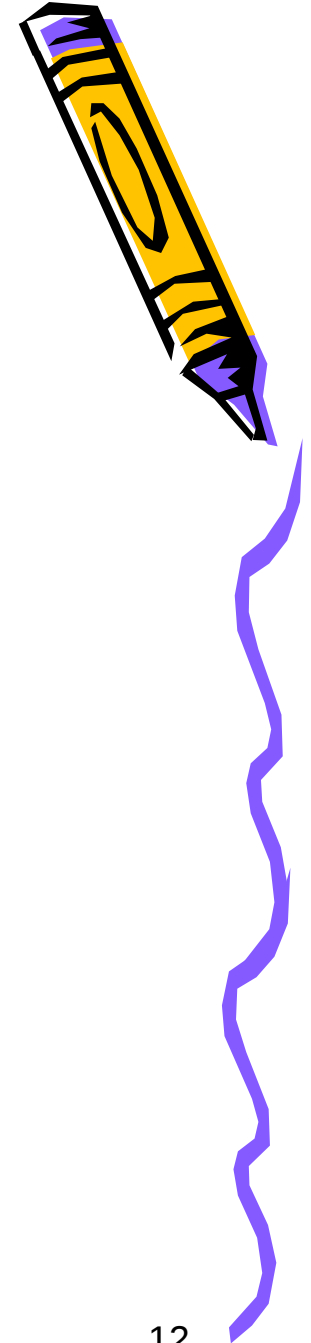
int indexOf(Object obj)

- ✓ pour avoir la *dernière occurrence*, il faut utiliser

int lastIndexOf(Object obj)



ArrayList



○ Suppression d'éléments

- Pour supprimer un élément à une position donnée, il faut utiliser la méthode

Object remove(int index)

throws IndexOutOfBoundsException



Collections



○ Autre classe

- il existe une autre classe qui est aussi très utile :

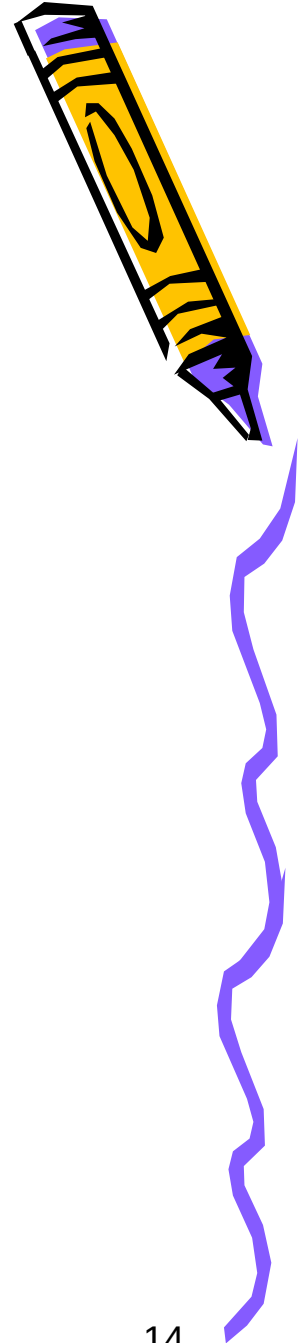
java.util.Vector

- voir le package java.util.Vector pour connaître les différences avec **ArrayList**



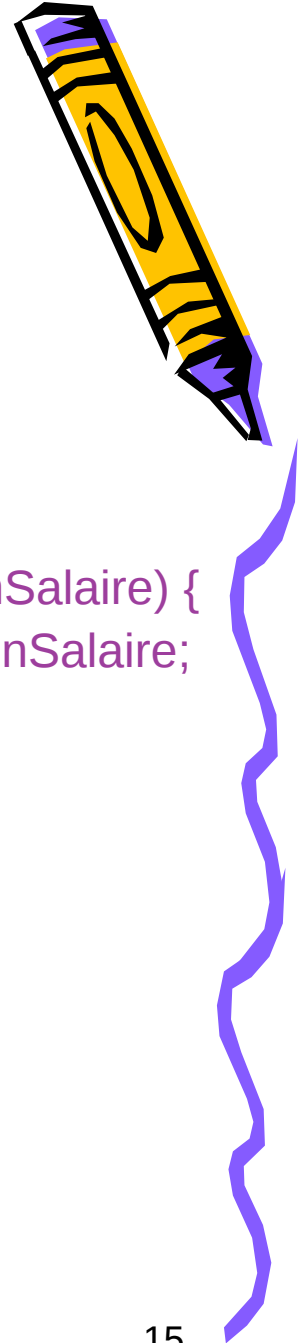
Résumé

- boolean add(Object obj)
- void add(int indice, Object obj)
- boolean contains(Object obj)
- Object get(int indice)
- int indexOf(Object obj)
- int lastIndexOf(Object obj)
- void remove(int indice)
- void set(indice, Object obj)
- int size()



Exemple

```
public class Employe {  
    private String leNom, lePrenom;  
    private double leSalaire  
    public Employe (String unNom, String unPrenom) {  
        leNom = unNom; lePrenom = unPrenom;  
    }  
    public Employe (String unNom, String unPrenom, double unSalaire) {  
        leNom = unNom; lePrenom = unPrenom; leSalaire = unSalaire;  
    }  
    public String getNom()  
    {  
        return leNom;  
    }  
}
```



Example

```
public static void main(String [] args) {  
    ArrayList tableauEmployes = new ArrayList();  
    Employe emp1 = new Employe("Charles", "McCathieNevile");  
    Employe emp2 = new Employe("Peter", "Sander");  
  
    tableauEmployes.add(emp1); tableauEmployes.add(emp2);  
    ...  
    If (!tableauEmployes.isEmpty()) {  
        for (int i = 0; i<tableauEmployes.size(), i++)  
        {  
            System.out.println((Employe) tableauEmployes.get(i).getNom());  
        }  
        tableauEmployes.remove(1);  
    }  
}
```

