Java, les objets : tout de suite !
Rassembler, grouper les objets

Cnam Paris
jean-michel Douin, douin@cnam.fr
Version du 26 janvier 2003

Notes de cours associées au chapitre 4
tutorial BlueJ
http://www.bluej.org/doc/documentation.html

Ce support accompagne, référence le livre de David J. Barnes & Michael Kölling
Objects First with Java A Practical Introduction using BlueJ
Pearson Education, 2003 ISBN 0-13-044929-6.

Sommaire

- **Collections et itérateurs**

- **Collections (de taille variable,extensible selon les besoins)**
  - **ArrayList**
  - **TreeSet**
  - **...**
- **Parcours**
  - **Boucle tant que,**
  - **Iterateurs,**

- **Tableaux (ou collections de taille fixe)**
  - **boucle for,**

Les collections pourquoi ?

- **Organisation des données**
  - Listes, tables, sacs, arbres, …
  - Données par centaines, milliers, millions ?
- **Quel choix ?**
  - En fonction de quels critères ?
    - Performance en temps d'exécution
      - lors de l'insertion, en lecture, en cas de modification ?
    - Performance en occupation mémoire
- **Implémentations pré définies ?**
- **Parcours de ces structures ?**

ArrayList : Une collection toute prête

- **La classe ArrayList du paquet java.util**

constructeur

ArrayList()

Constructs an empty list with an initial capacity of ten.

...

méthodes

boolean add(Object o);

Appends the specified element to the end of this list.

Object get(index i);

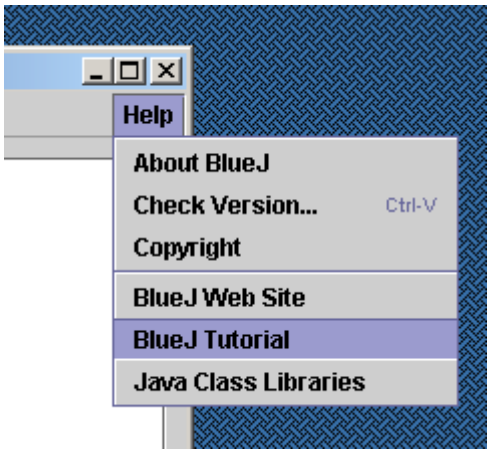
Returns the element at the specified position in this list.

int size();

Returns the number of elements in this list.

...

Documentation et tests unitaires

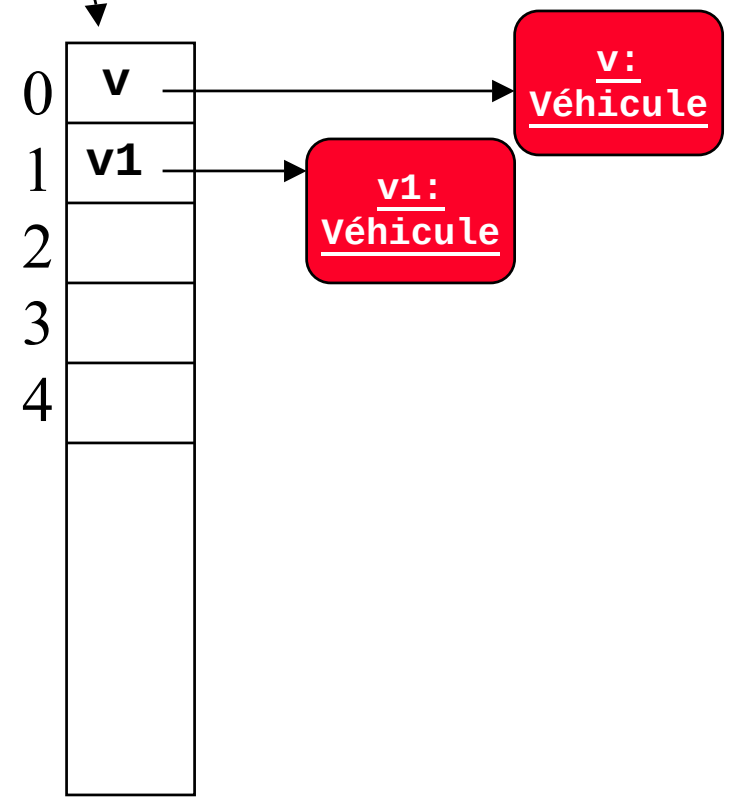


- **Documentation**
  - **Java API**
    - **item Java Class Libraries**
  - **Tutorial**
    - **item BlueJ Tutorial**

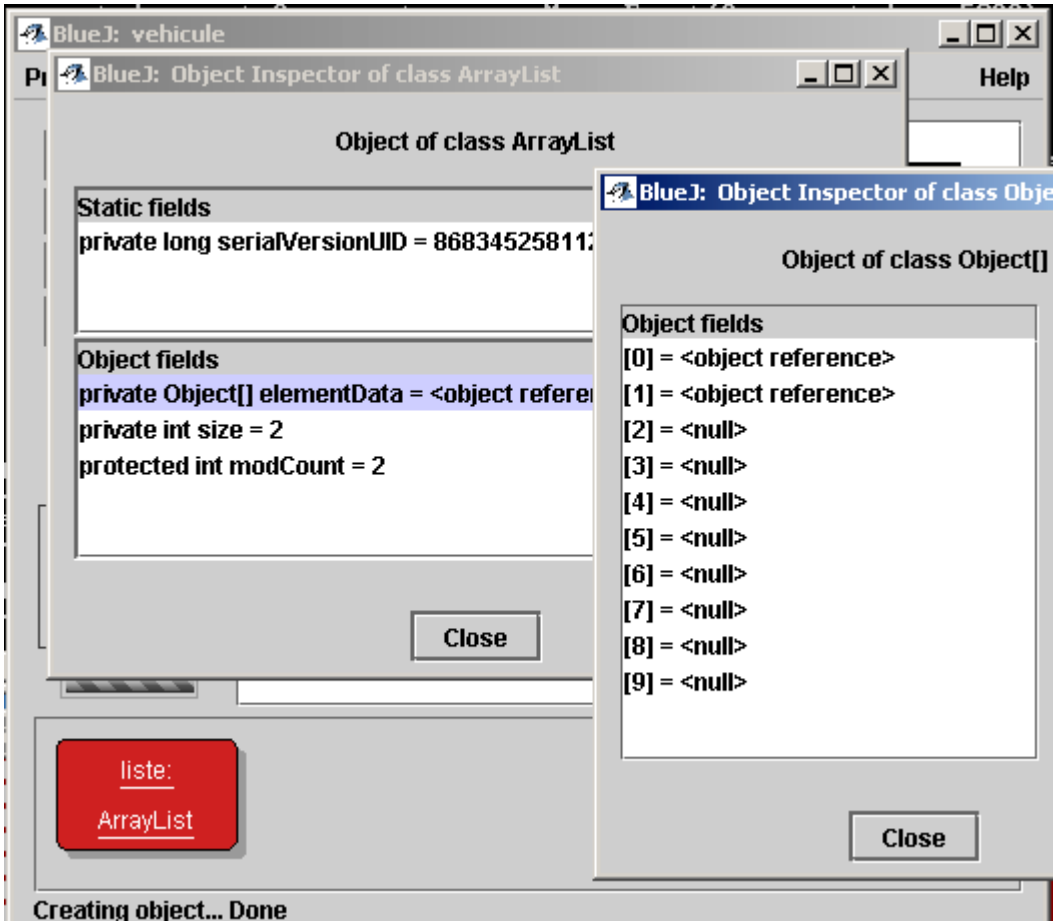
- **Tests unitaires**
  - **tutorial page 29, chapitre 9.6**

ArrayList : un schéma

```
Véhicule v = new Véhicule("FJR");
Véhicule v1 = new Véhicule("majestic");
ArrayList liste = new ArrayList();
liste.add(v);
liste.add(v1);
```



ArrayList, une instance avec BlueJ



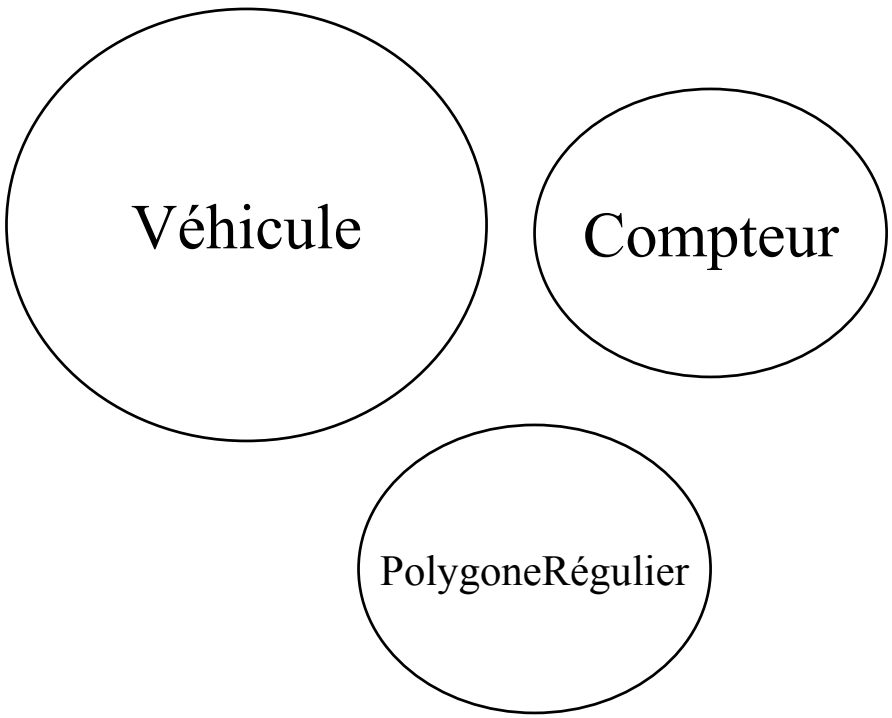
- [0] = v, référence vers FJR
- [1] = v1, référence vers majestic

Tout est donc Object ?

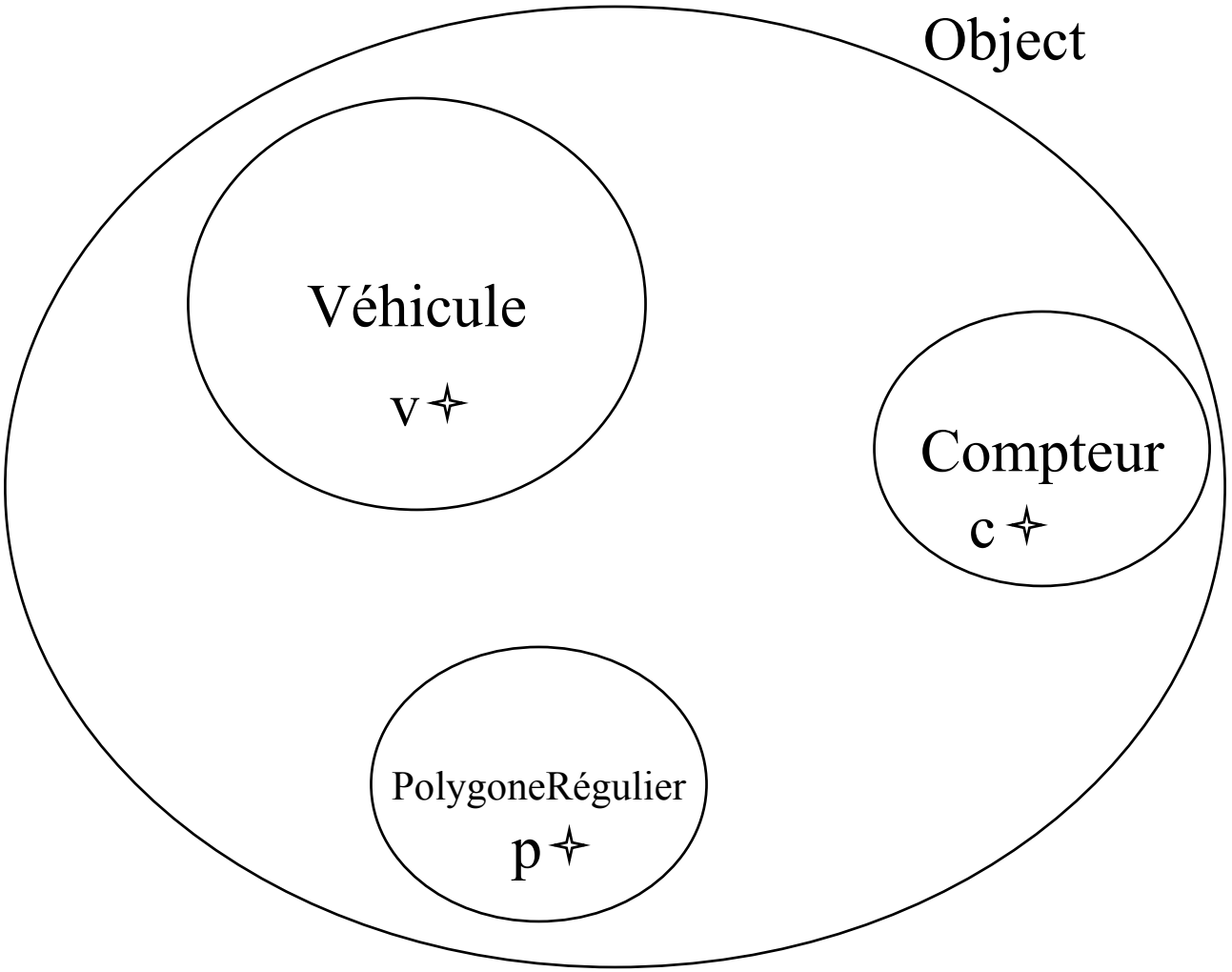
```
boolean add(Object o);   // de la classe ArrayList
```

Avec

```
liste.add(v);  // Véhicule v
```



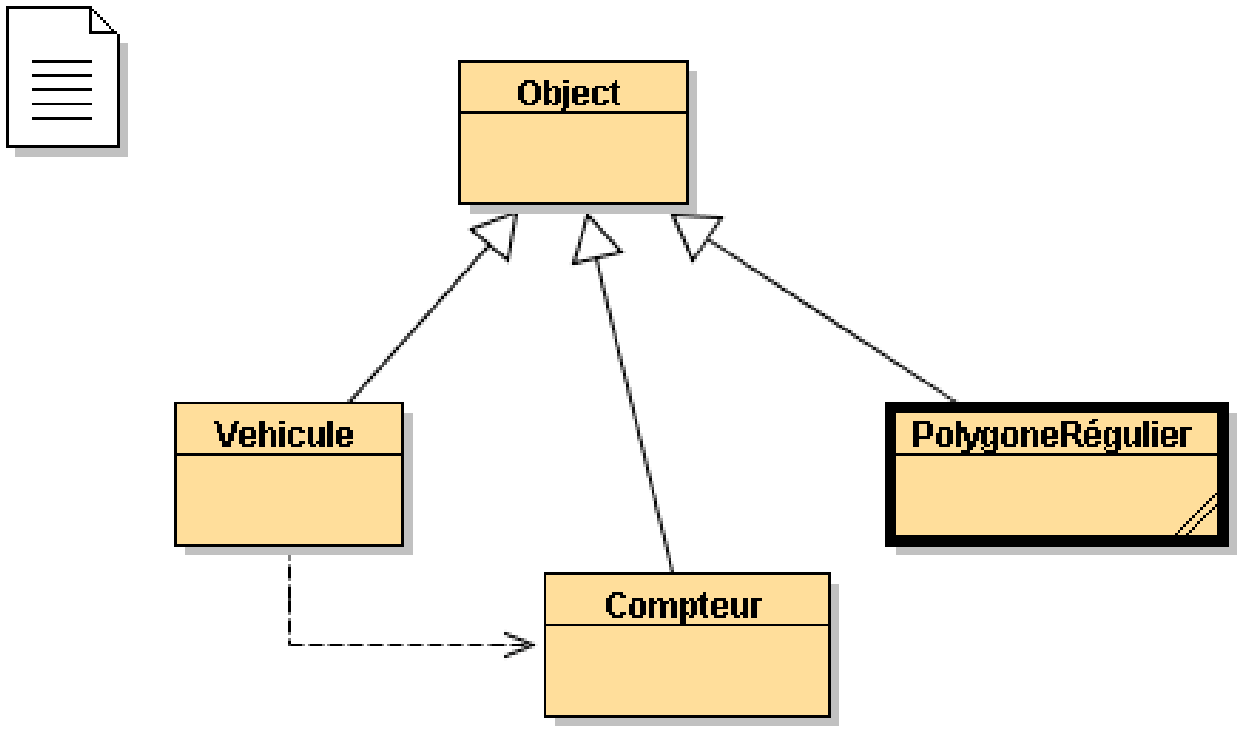
Tout est bien Object !



Object o …
o = v; // ?
o = c; // ?
o = p; // ?

p = o; // ?

Diagramme de classes



Un exemple d ’affectation

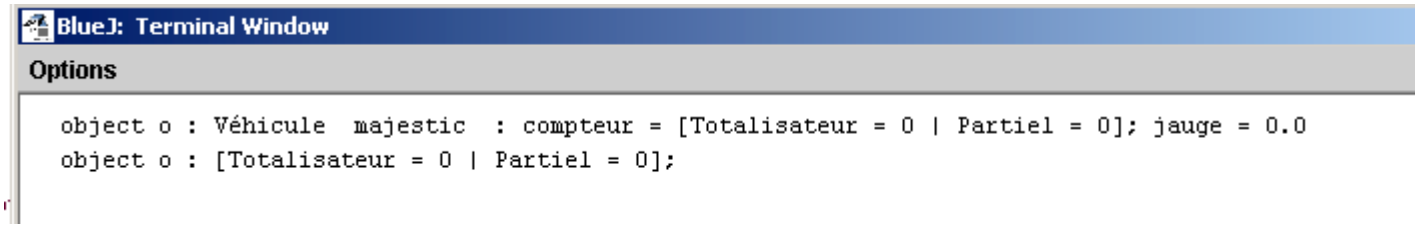
- Object o = new Object();
- Véhicule v = new Véhicule(" majestic ",0.1);

- o = v;
- System.out.println(" object o : " + o.toString());

- Compteur c = new Compteur();

- o = c;
- System.out.println(" object o : "   + c.toString());

Sélection de la bonne méthode



type déclaré et type constaté

```
Object o = new Object();
Compteur c = new Compteur();
o = c;
```

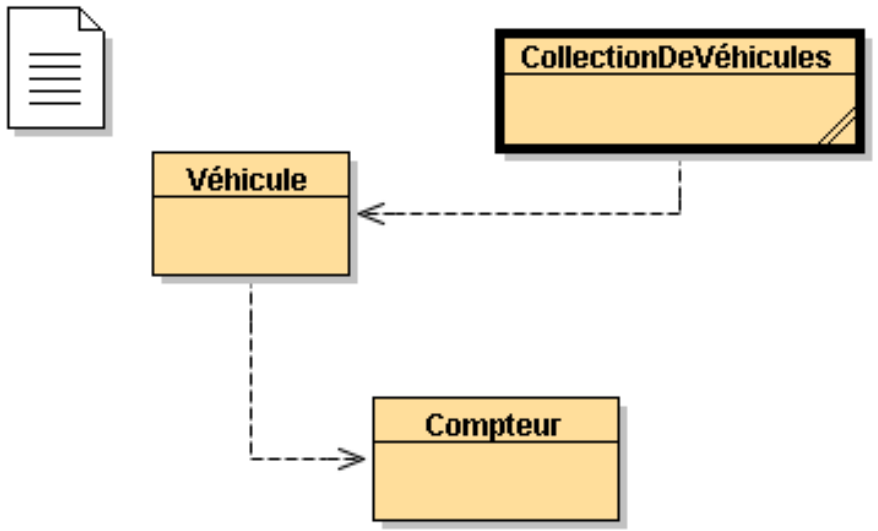
- o est de type **déclaré Object,**
- o est de type **constaté Compteur**

- **Ce sera la méthode toString de la classe Compteur qui sera exécutée**

Changement de type implicite / explicite, instanceof

```
boolean add(Object o);   // classe ArrayList

liste.add(v);  // implicite :
               //  v est recopié dans o  (o = v;)


Object get(int index);   // classe ArrayList

Véhicule v1;
v1 = liste.get(1);  // erreur de syntaxe !


v1 = (Véhicule) liste.get(1);

Object o = liste.get(1);
If (o instanceof Véhicule) v1 = (Véhicule)o;
```

Un exemple : Une collection de véhicules




```
import java.util.ArrayList;
public class CollectionDeVéhicules{
   private ArrayList liste;

   public CollectionDeVéhicules(){
       liste = new ArrayList();
   }
```

Notre exemple : Une collection de véhicules

```
import java.util.ArrayList;
public class CollectionDeVéhicules{
   private ArrayList liste;

   public CollectionDeVéhicules(){
       liste = new ArrayList();
   }

   public void ajouter(Véhicule v){
       liste.add(v);
   }

   public int nombreDeVéhicules(){
     return liste.size();
   }

 public void afficherTousLesVéhicules(){ ...
```

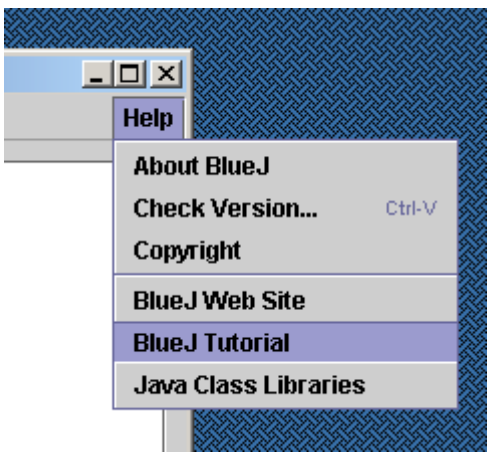
import java.util.ArrayList;

```
import java.util.ArrayList;
```

- usage de librairies standard du j2se,
- premières instructions du programme Java

documentation de Sun (item Java Class Librairies)

tutorial paragraphe 9.6, page 29



ma modeste collection

```
CollectionDeVéhicules maCollection = new CollectionDeVéhicules();

Véhicule v = new Véhicule("TERROT 500",0.05);
v.faireLePlein();v.rouler(50.500);
maCollection.ajouter(v);

v = new Véhicule("FJR 1300",0.075);
v.faireLePlein();v.rouler(24.75);
maCollection.ajouter(v);

v = new Véhicule("Bonnevile 650",0.06);
v.faireLePlein();v.rouler(36.75);
maCollection.ajouter(v);


maCollection.afficherTousLesVéhicules();
```

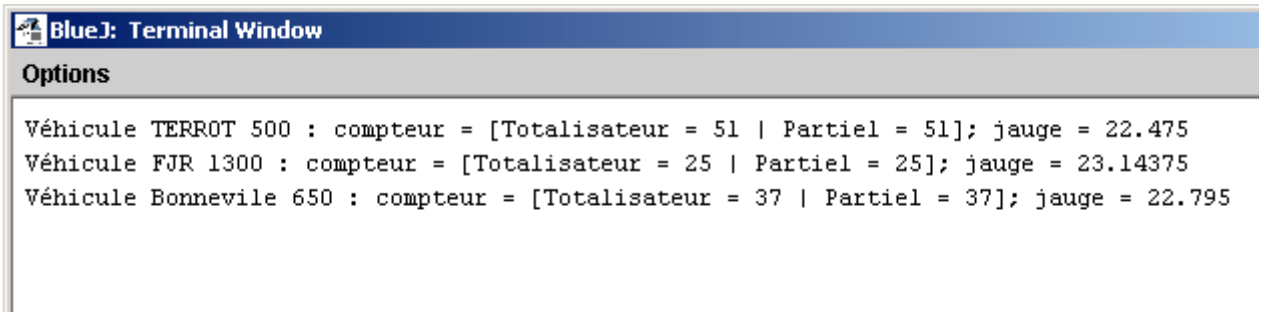
Itération : while

```
public void afficherTousLesVéhicules(){
      int index = 0;
      while( index < nombreDeVéhicules()){
        System.out.println(liste.get(index));
        index++;
      }
    }
```

maCollection.afficherTousLesVéhicules();

Quel affichage et pourquoi ?

Affichage de ma collection



- ***Type déclaré et type constaté***

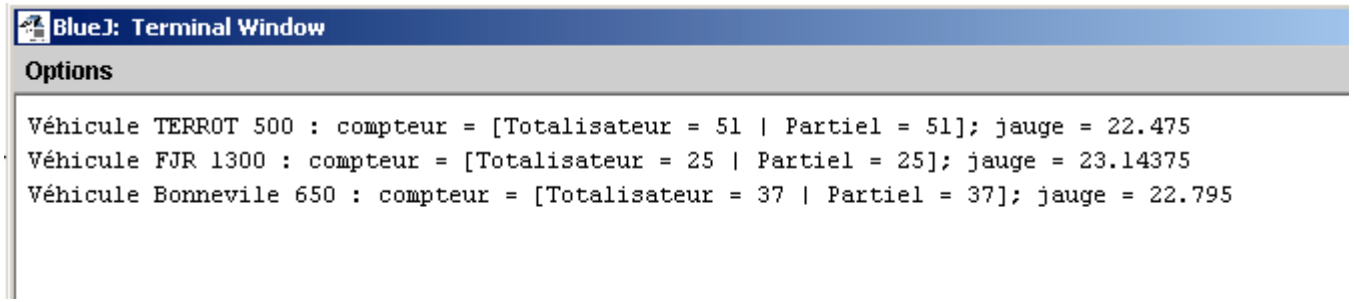
Itération : Itérateur, Iterator

```
public void afficherTousLesVéhicules(){
  Iterator it = liste.iterator();
  while( it.hasNext()){
    System.out.println(it.next());
  }
}
```

Même affichage et parce que !

type constaté et type déclaré

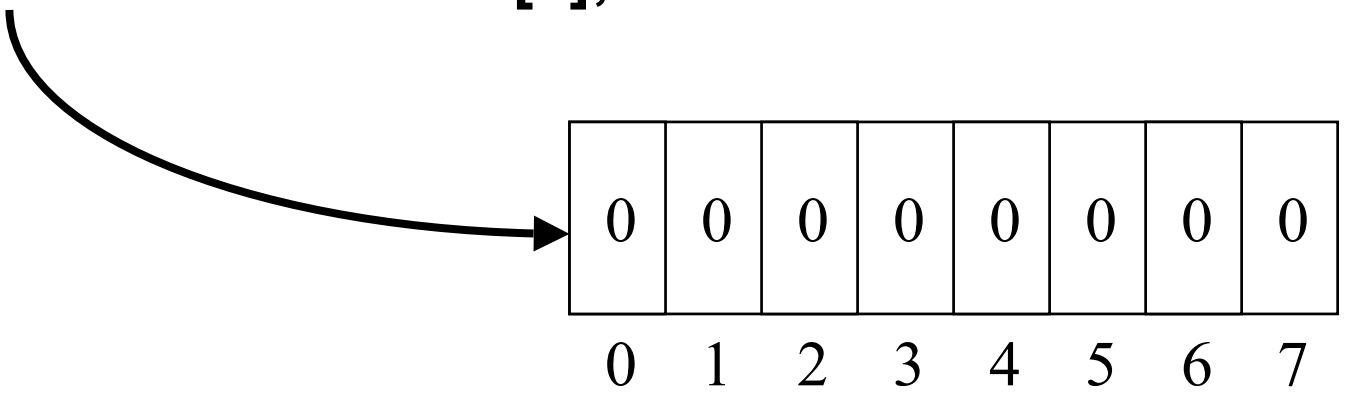
Le même affichage !



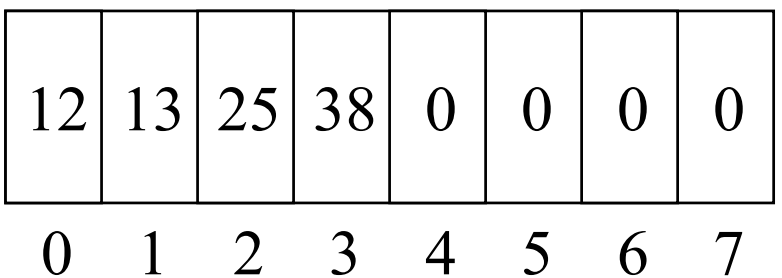
- ***Type déclaré et type constaté***

Grouper, rassembler

- **Les tableaux en Java ou les collections de taille fixe**
- **int [] tpRendus = new int[8];**



- **tpRendus[0] = 12;**
- **tpRendus[1] = 13;**
- **tpRendus[2] = 25;**
- **tpRendus[3] = 38;**



Type structuré : tableau

- **Déclarations de tableaux**

```
        int[] mois;   // mois est affecté à null ....
  ou    int[] mois = new int[12];
  ou    int[] mois={31,28,31,30,31,30,31,31,30,31,30,31};
```

- **Déclarations de tableaux à plusieurs dimensions**

  double [][] m= new double [4][4];

- **Accès aux éléments**

  le premier élément est indexé en 0

  vérification à l'exécution des bornes, erreur signalées (levée d'exception)

- **En paramètre**

  la variable de type tableau est une référence,

  le passage par valeur de Java ne peut que transmettre la référence sur le tableau

  2 syntaxes autorisées : int mois[] ou int[] mois; la seconde est préférée !, la première est devenue ancestrale ...

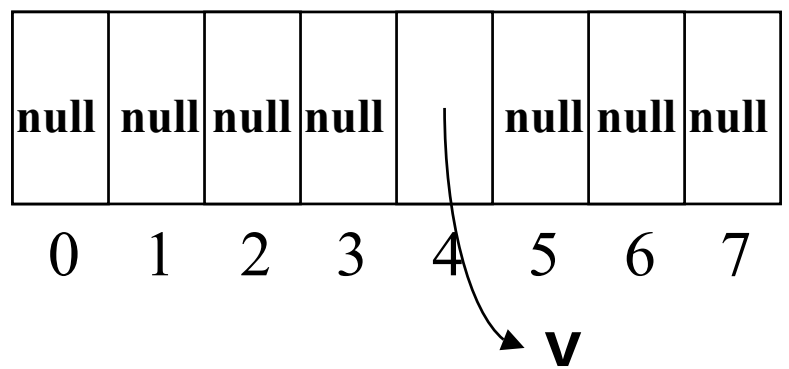
Les tableaux, structure

- **Un tableau d'entiers**

```
int[]  poids = new int[12];
```

- **un tableau de véhicules**

```
Véhicule[]  liste = new Véhicule[8];
liste[4] = v;
```

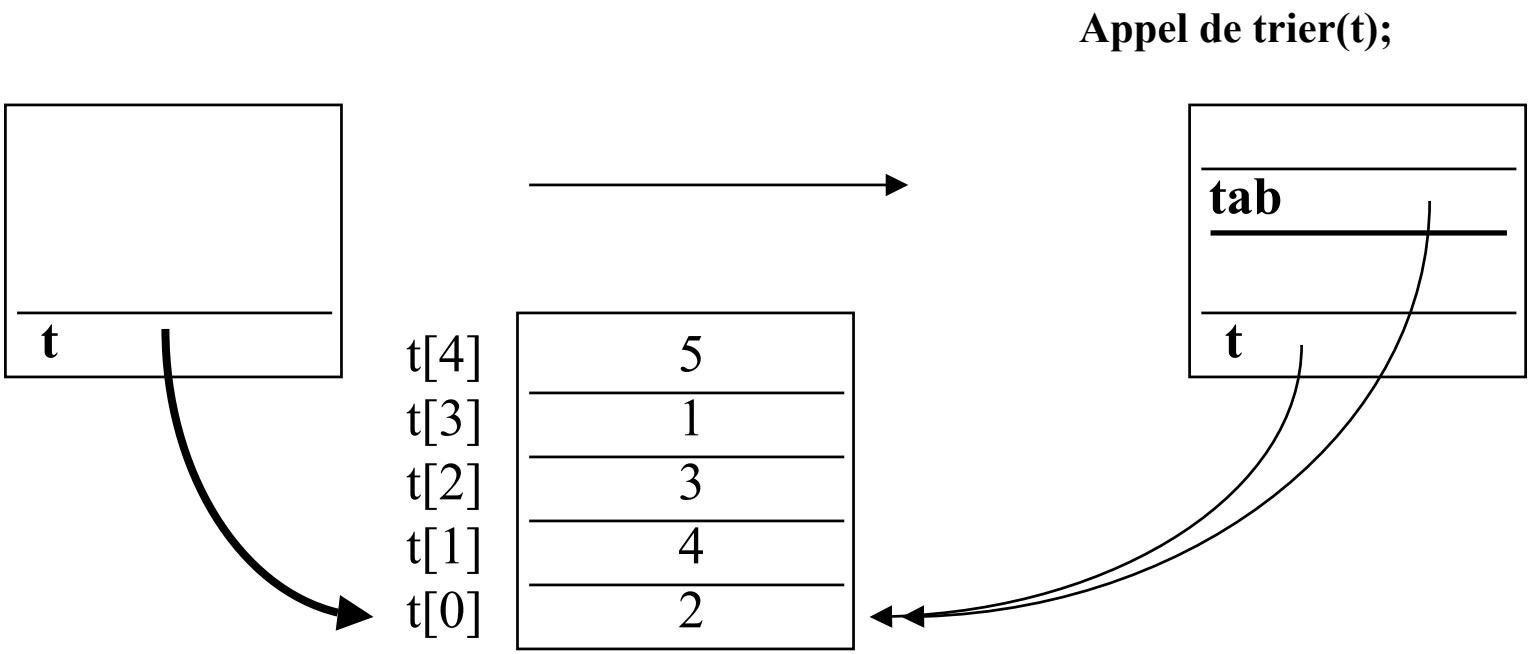


Passage de paramètres par valeur uniquement

```
static void trier(int[] tab){
  tab[0] = 8; // --> t[0] == 8;
```

```
int[] t = {2,4,3,1,5};
trier(t);
```

Un extrait de la pile d'exécution



Itération : la boucle for

- ***for (****initialisation; terminaison; itération****)***
- *instructions;*

- **<==>** *

- *initialisation;*
- ***while (****terminaison****){***
- *instructions;*
- *itération;*
- ***}***

- *** il ya des exceptions …**

Itération : la boucle for, exemples

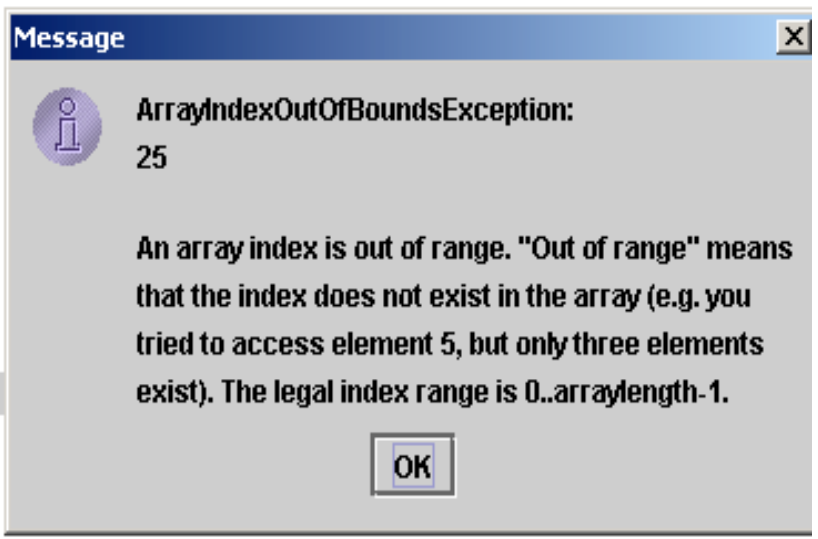
- **Itération, *for(initialisation; terminaison; itération)***

```
int[] table = new int[25];
for(int i = 0; i < table.length; i++){
  table[i] = -1;
}

Véhicule[] parc = new Véhicule[10];
// affectation des éléments
for(int i = 0; i < parc.length; i++){
  System.out.println(parc[i]); // ou parc[i].toString();
}
```

En cas de dépassement, …

```
72
73        int[] table = new int[25];
74        for(int i = 0; i <= table.length; i++){
75          table[i] = -1;
76        }
77
78
79
80      }
81
82  }
```

ArrayIndexOutOfBoundsException:
25



Mais tout est encore Object ?

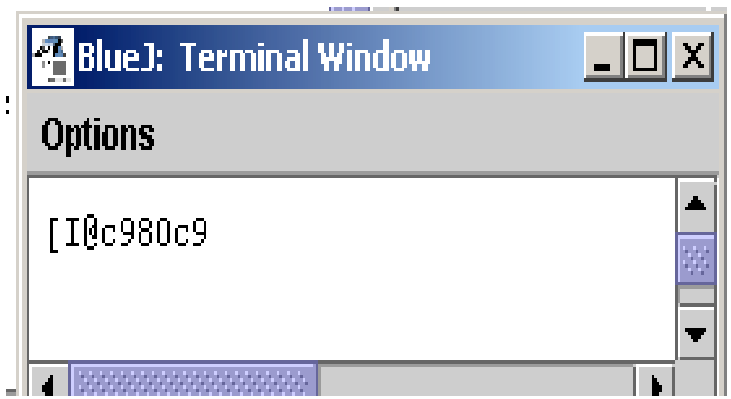
- **Les collections**

```
Object o = new Object();
ArrayList liste = new ArrayList();
o = liste;
System.out.println(o);
```

- **Et les Tableaux ?**

```
Object o;
int[] table = new int[25];
o = table;
System.out.println(o);
```

???



Le toString() encore lui

- **collections . toString();**

```
public class CollectionDeVéhicules{

   private ArrayList liste;


    public String toString(){
      return liste.toString();
    }
```

- *Mais … il ne fonctionne pas sur les tableaux*

de Tableaux en Collections

- **La classe java.util.Arrays, la méthode asList**

```
import java.util.Arrays;
……
public class CollectionDeVéhicules{
  private ArrayList liste;

  /** ajouter à la collection tous les véhicules
   *  de la table
   */
  public void ajouter(Véhicule[] table){
    liste.addAll(Arrays.asList(table));
  }
```

De Collections en Tableaux

- **De la classe ArrayList**
- **public Object[] toArray(Object[] a)**

  Returns an array containing all of the elements in this collection; the runtime type of the returned array is that of the specified array. If the collection fits in the specified array, it is returned therein. Otherwise, a new array is allocated with the runtime type of the specified array and the size of this collection.

  String[] x = (String[]) v.toArray(new String[0]);

```
public Véhicule[] uneCopie(){
  return (Véhicule[])liste.toArray(new Véhicule[0]);
}
```

Résumé, synthèse