

ENSEMBLES D'OBJETS

Ensembles d'objets

- Comment stocker un ensemble d'objets ?
- Comment manipuler des ensembles ?
- Plusieurs solutions :
 - les tableaux : **Array**
 - les collections : **Vector, Stack, Set, HashTable ...**

Les tableaux d'objets

- La création se déroule en deux étapes :
 - Allocation du tableau, chaque élément du tableau étant une référence sur un objet
 - Création des objets

Les tableaux d'objets

```
public class TestArray {  
  
    static final int DEFAULT_SIZE = 15 ; // taille fixée arbitrairement pour les tests  
  
    public static void main(String[] args) {  
        Integer[] a = new Integer[DEFAULT_SIZE] ; // instantiation du tableau  
  
        System.out.println("Taille du tableau a : " + a.length) ;  
  
        for(int i = 0 ; i < a.length ; i++) {  
            a[i] = new Integer(i*10) ; // instantiation des Integer  
            System.out.println("a[" + i + "] = " + a[i]) ;  
        }  
    }  
}
```

Les tableaux d'objets

```
public class ArrayInit {  
  
    public static void main(String[] args) {  
        Integer[] a = {  
            new Integer(1),  
            new Integer(2),  
            new Integer(3),  
        };  
    }  
}
```

Autres manières de créer un tableau d'objets.

L'autre boucle for

```
String[] nomsCouleurs = { "rouge", "vert", "bleu" };
```

```
for (String s : nomsCouleurs){  
    System.out.println(s);  
}
```

Tableau comme valeur de retour

```
public class TestArray {  
    public static void main(String[] args) {  
        String[] chaines = genererChaines(5) ;  
        for(String s : chaines) System.out.println(s);  
    }  
    public static String[] genererChaines(int nbChaines){  
        String[] resultat = new String[nbChaines] ;  
        for(int i=0 ; i<resultat.length ; i++){  
            resultat[i] = "Chaine n° " + i ;  
        }  
        return resultat ;  
    }  
}
```



Chaine n° 0
Chaine n° 1
Chaine n° 2
Chaine n° 3
Chaine n° 4

Les collections

- Les collections ont pour objectif de fournir des structures de stockage plus évoluées que les tableaux
- Les types primitifs ne peuvent pas être stockés dans une collection
- Avantage : les collections permettent de gérer des ensembles hétérogènes et de taille dynamique
- Les collections implémentent l'interface **java.util.Collection**
 - 3 principales sous-interfaces : **List<E>**, **Set<E>**, **SortedSet<E>**
 - Interfaces -> Classes génériques
- Plusieurs classes concrètes très utiles dont :
 - **Vector<E>**
 - **HashSet<E>**
 - **LinkedHashSet<E>**
 - ...

Vector<E>

```
Vector<String> couleurs = new Vector<>() ;  
couleurs.add("rouge") ;  
couleurs.add("vert") ;  
couleurs.add("bleu") ;  
couleurs.insertElementAt("jaune", 1) ; ;  
  
for(String c : couleurs){  
    System.out.println(c);  
}
```

même type

rouge
jaune
vert
bleu

dans l'ordre...

HashSet<E>

```
HashSet<String> couleurs = new HashSet<>() ;

couleurs.add("rouge") ;
couleurs.add("jaune") ;
couleurs.add("vert") ;
couleurs.add("bleu") ;

for(String c : couleurs){
    System.out.println(c);
}
```

bleu
jaune
vert
rouge

dans le désordre...

LinkedHashSet<E>

```
LinkedHashSet<String> couleurs = new LinkedHashSet<>() ;

couleurs.add("rouge") ;
couleurs.add("jaune") ;
couleurs.add("vert") ;
couleurs.add("bleu") ;

for(String c : couleurs){
    System.out.println(c);
}
```

rouge
jaune
vert
bleu

dans l'ordre !

Les Map<K,V>

- Interface **java.util.Map<K,V>** pour stocker des objets avec des paires (clé, valeur)
- Principales classes concrètes :
 - **HashTable<K,V>**
 - **HashMap<K,V>**
 - **TreeMap<K,V>**
 - ...

Exemple : HashMap<K,V>

```
HashMap <Integer, String> personnes = new HashMap<Integer, String>();
```

```
Integer numSecu = 123456789;
```

```
String nom = "Stark";
```

```
personnes.put(numSecu, nom);
```

```
personnes.put(521346987, "Rogers");
```

```
personnes.put(654987123, "Banner");
```

```
System.out.println("Listing des assurés : ");
```

```
for (Integer cle : personnes.keySet()){
```

```
    System.out.println( cle + " -> " + personnes.get(cle) );
```

```
}
```

Trace :

Listing des assurés : 123456789 -> Stark 521346987 -> Rogers 654987123 -> Banner
