Bases de la POO / Java

Classes collections

TABLEAUX
COLLECTIONS
COLLECTIONS DE BASE

Tableaux

- `int [] t;`
- `t = new int[10];`
- `int taille = t.length;`

➔ Arrays utilities

➔ tri, remplissage, égalité…

➔ http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Arrays.html

Exemple 1


```
import java.util.Arrays;

int[]    T1={0, 6, 2, -4, 3, 8, -11, 0, 1};
String[] T2={"bleu","rouge","blanc","vert","mauve","indigo"};

Arrays.sort(T1);
for (int i=0; i<T1.length; i++) System.out.print(T1[i] +" ");
Arrays.sort(T2);
for (int i=0; i<T2.length; i++) System.out.print(T2[i] +" ");
```

➔ **Exécution**

```
-11 -4 0 0 1 2 3 6 8
blanc bleu indigo mauve rouge vert
```

Exemple 2


```
int[] T1 = new int[10];
String[] T2 = new String[7];

Arrays.fill(T1,5);

Arrays.fill(T2, 1, 3, "bleu");
```

➔ **Exécution**

```
5 5 5 5 5 5 5 5 5 5
null bleu bleu null null null null
```

Exemple 3


```
int[] T1 = {0, 6, 2, 4, 3};
int[] T2 = {0, 6, 2, 4, 3};

Tests.Unit(false, T2==T1);
Tests.Unit(false, T2.equals(T1));
Tests.Unit(true, Arrays.equals(T1, T2));
```

➔ **Exécution**

```
Valeur attendue : false
Valeur obtenue  : false

Valeur attendue : false
Valeur obtenue  : false

Valeur attendue : true
Valeur obtenue  : true
```

Collection (1)

➔ Définition

Forme d'agrégat dans lequel tous les composants (éléments) sont de MEME type (ensemble homogène)

- Nombre variable d'éléments
- Définition d'une relation d'appartenance
- Contrôle de validité et d'appartenance
- Construction récurrente possible et simple
- Gestion intégrée de la persistance
- **Généricité** vis-à-vis du type des éléments stockés

Collection (2)

→ Cahier des charges d'une collection

- Spécifications des contraintes de rangement
- Spécifications des modes de désignation
- Spécifications des contrôles en ligne
- Contraintes de performances
- Contraintes d'interface
- Contraintes opérationnelles

Collection (3)

➔ Opérations de base sur une collection

En plus des constructeurs et des surcharges des méthodes héritées de la classe Object :

- Ajout d'un nouvel élément
- Contrôle (éventuel) d'unicité
- Modification d'un élément quelconque
- Suppression d'un élément quelconque
- Calcul du cardinal de la collection
- Appliquer un algorithme sur tous les éléments

Techniques de désignation

➔ Désignation simple ou multiple

- Désignation par la position absolue
- Désignation par la position relative
- Désignation par une clé d'identification
- Désignation par le contenu
- Composition de désignations
- Désignation de sous-ensembles

Collections de base

➔ Chaque collection est fournie sous forme d'une « classe » dédiée, générique vis à vis des éléments stockés

- Ensembles (unicité, pas d'ordre)
- Tableaux, Listes chaînées
- Piles (Stack), Files
- Dictionnaires (HashMap, LinkedHashMap)
- Arbres

		Implementations				
		Hash Table	Resizable Array	Balanced Tree	Linked List	Hash Table + Linked List
Interfaces	Set	HashSet		TreeSet		LinkedHashSet
	List		ArrayList		LinkedList	
	Map	HashMap		TreeMap		LinkedHashMap

Composition de collections

➔ Possibilité de constructions récurrentes, sans limitation de la profondeur

- Tableaux de collections
- Listes de collections
- Piles de collections
- Files de collections
- Dictionnaires de collections
- Arbres de collections

Eléments d'une collection Java

➔ Chaque élément est une référence (sur objet)

- Tous les objets cibles sont de même type
- Transtypage dynamique implicite ascendant (**Object**)
- Problème des vues multiples éventuelles
  (**Clonage préalable recommandé**)