

Algorithmique et programmation par objets

Inf F3

Licence 2 MIASHS

Université Grenoble Alpes

Jerome.David@univ-grenoble-alpes.fr

2023-2024

<http://miashs-www.u-ga.fr/~davidjer/inff3/>

Cours 4 - Structures de contrôle et tableaux

- Les structures conditionnelles, itératives et embranchements
- Les tableaux à une dimension et à plusieurs dimensions

Les structures de contrôle

- Les structures conditionnelles
 - If-else
- Les structures itératives (boucles)
 - While
 - Do-while
 - For
- Les branchements inconditionnels
 - return, break, continue, étiquettes
- Les Structures de sélection
 - switch

If-else

- Structures possibles

```
if (expression_booleenne) <ENONCE_1>
```

```
if (expression_booleenne) <ENONCE_1> else <ENONCE_2>
```

```
(expression_booleenne)?<INSTRUCTION_1>:<INSTRUCTION_2>;
```

<ENONCE_1> et <ENONCE_2> sont soit une instruction (**prohibé dans ce cours**), soit un bloc d'instructions, soit une structure de contrôle

```
if (expression_booleenne) {  
    // instructions exécutées  
    // si expression_booleenne  
    // est évaluée à true  
}  
else {  
    // instructions exécutées  
    // si expression_booleenne  
    // est évaluée false  
}
```

```
if (exp_1) {  
    // exp_1 est vrai  
}  
else if (exp_2) {  
    // exp_1 est faux mais exp_2 est vrai  
}  
else {  
    // exp_1 et exp2 sont faux  
}
```

While et do-while

- Principe : répéter l'exécution d'un bloc d'instructions tant qu'une condition est vraie
 - La condition peut être vérifiée au début de la boucle ou après

```
while (expression_booléenne) {  
  // instructions à exécuter  
  // tant que expression_booléenne  
  // est évaluée à true  
}
```

```
do {  
  // instructions à exécuter  
  // tant que expression_booléenne  
  // est évaluée à true  
} while (expression_booléenne);
```

```
int i=10;  
while (i<10) {  
  System.out.println(i);  
  i++;  
}
```

```
int i=10;  
do {  
  System.out.println(i);  
  i++;  
} while (i<10);
```

Quelle est la différence entre ces deux boucles ?

Boucle for

- La boucle for est sans doute la plus utilisée
 - Elle exécute l'initialisation avant la première itération
 - Ensuite elle vérifie la condition
 - A la fin de chaque itération, l'instruction d'étape est réalisée
- Syntaxe :

```
for ( intialisation ; condition ; étape ) {  
    // instructions à exécuter  
    // tant que condition est évaluée à true  
}
```

```
for ( int i=1 ; i<10 ; i++ ) {  
    System.out.println(i);  
}
```

Instructions break et continue

- Elles permettent de contrôler l'exécution d'une boucle
 - break permet de stopper l'exécution de la boucle
 - continue stoppe l'itération courante et poursuit la prochaine itération
- Les instructions peuvent utiliser aussi des labels
 - C'est à éviter dans la mesure du possible !!!

Structure de sélection

- Permet de sélectionner des instructions à exécuter en fonction de la valeur d'une expression
 - L'expression est de type int, short, char, ou byte
 - Le cas default est optionnel

```
switch (expression) {  
    case valeur1 : // instructions  
                  break;  
    case valeur2 : // instructions  
                  break;  
    // ...  
    default : // instructions  
}
```

Les Tableaux

- Un tableau est une collection indicée d'objets ou de primitives (appelés éléments du tableaux)
- Les éléments d'un tableaux ont tous le même type
- Une référence de tableau est déclarée en utilisant le nom du type suivi de « [] »

```
int [] monTableau ;
```

- Notation alternative (à éviter) : `int monTableau[] ;`
- Un tableau est un objet. Une référence de tableau est par défaut initialisée à `null`

Initialisation des tableaux

- Un tableau sont manipulé par référence :
 - Par défaut la référence est initialisée à `null`
 - il faut donc allouer un espace mémoire (sur le tas)
- Création et initialisation d'un tableau

```
int[] monTableau = new int[] {1,2,3};
```

- Un tableau de 3 entiers, initialisé avec les valeurs 1, 2 et 3

```
int[] monTableau2 = new int[5];
```

- Créer un tableau de 5 entiers. Les éléments du tableaux sont initialisés avec la valeur par défaut des int, i.e. 0

Question : qu'est ce qui se passe si j'écris :

```
int[] t = new double[2];
```

Erreur de compilation

Erreur de compilation

```
int[] ty = new int[] {1,1.0,2.1};
```

```
double[] tx = new double[] {1,1.0,2.1};
```

OK

Tableau et allocation mémoire

- Exercice : Représenter l'état de la mémoire après exécution du code suivant
 - Un tableau est représenté par

1	2	3
---	---	---

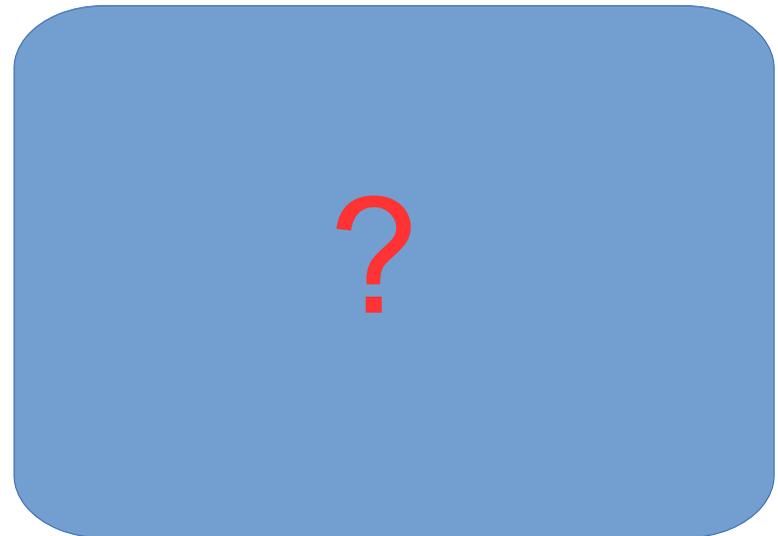
```
int[] t1 = new int[]{1,2,3};  
int[] t2 = new int[4];  
int[] t3 = new int[]{1,2,3};  
int[] t4=t1;  
int[] t5;
```

Pile ou Stack



?

Tas ou heap



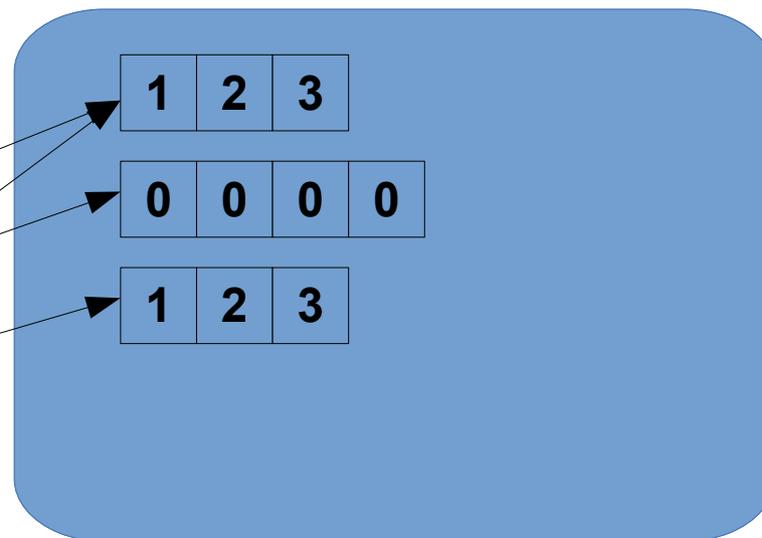
Tableaux et allocation mémoire

```
int[] t1 = new int[]{1,2,3};  
int[] t2 = new int[4];  
int[] t3 = new int[]{1,2,3};  
int[] t4=t1;  
int[] t5;
```

Pile ou Stack

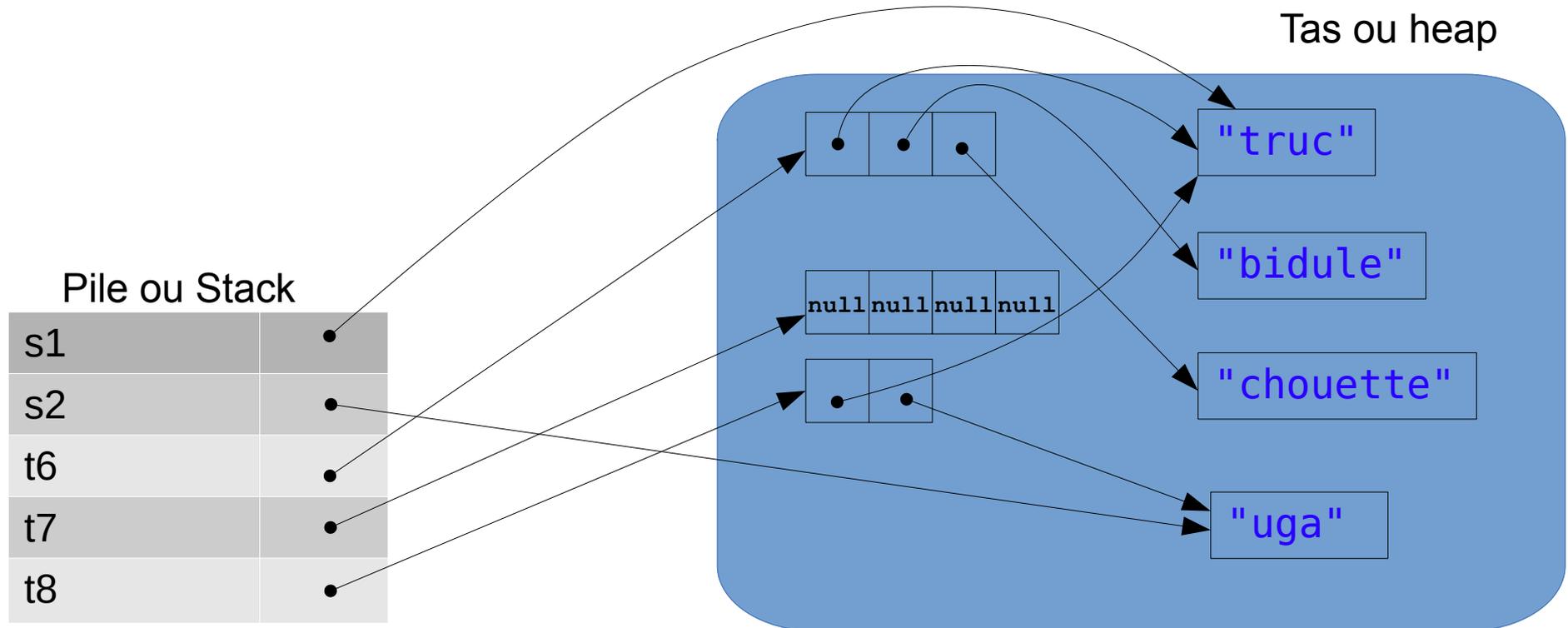
t1	•
t2	•
t3	•
t4	•
t5	null

Tas ou heap



Tableaux : cas des objets

```
String s1="truc";  
String s2="uga";  
String[] t6=new String[] {s1,"bidule","chouette"};  
String[] t7= new String[4];  
String[] t8=new String[] {s1,s2};
```



Manipulation des tableaux

- On peut accéder aux éléments d'un tableau en donnant leur indice (position dans le tableau)
 - 1^{er} élément : `t1[0]`
 - n^e élément : `t1[n-1]`
- Tout tableau possède un attribut donnant sa taille :
 - `t1.length`
 - Cet attribut n'est accessible qu'en lecture !
- Accéder à un élément hors des limites lève une exception : `ArrayIndexOutOfBoundsException`

Exemple de manipulation de tableau

- Un exemple
 - Initialisation d'un tableau d'entiers de taille aléatoire entre 1 et 100
 - Remplissage du tableau avec des entiers entre 0 et 999 générés aléatoirement
 - Affichage des éléments du tableau

```
int[] tab= new int[(int) (Math.random()*100)+1];
for (int i = 0 ; i < tab.length ; i++ ) {
    tab[i]=(int) (Math.random()*1000);
}
for (int element : tab) {
    System.out.println(element);
}
```

Boucle foreach

- Forme succincte du « for » pour le parcours de tableaux (ou autres collections)
 - Ne permet que l'accès en lecture

```
double[] tab = new double[]{1,2,3};  
for (double element : tab) {  
    System.out.println(element);  
}
```

Exercice :
donnez la version de ce parcours avec le for
classique

Tableaux multidimensionnels

- Ce sont des tableaux de tableaux
 - i.e. un tableau dont les éléments sont des tableaux
- Le nombre dimensions n'est pas limité

- **Création:**

- **En une fois :**

```
int[][] matrice = new int[10][20] ;
```

- **En plusieurs fois :**

```
int[][] matrice2 = new int[10][] ;
```

```
matrice2[2] = new int[2] ;
```

```
matrice2[0] = new int[3] ;
```

- **Avec initialisation des éléments**

```
int[][] matrice = new int[][]{ {1,2,3} , {1,2,3} } ;
```

Tableaux multidimensionnels

- Exercice

- Créer une matrice d'entiers triangulaire de dimension n

- La première ligne à 1 élément, la seconde 2 éléments, etc.

```
int n=3;
int[][] triangle = new int[n][];
for (int i=0 ; i<n ; i++) {
    triangle[i] = new int[i+1];
}
```

Utilitaires sur les tableaux

- La classe `java.util.Arrays` fourni de nombreuses procédures utilitaires
 - Pour l'utiliser, il faut mettre au début du fichier du programme : `import java.util.Arrays;`
- La représentation en chaîne de caractères de tableaux
 - `Arrays.toString(monTableau)` : 1 dimension
 - `Arrays.deepToString(monTableau)` : >1 dimensions

```
int[] tab = new int[]{3,5,1,6};
System.out.println(Arrays.toString(tab));

int[][] tab = new int[][]{ {1,2,3} , {3,4,5} };
System.out.println(Arrays.deepToString(tab));
```

Egalité de tableaux

- Rappel
 - l'opérateur `==` appliqué à deux variables de type tableau retourne l'égalité des références
 - La méthode `equals(...)` appliquée sur un tableau donne le même résultat que `==` !!!
- Pour comparer leur contenu :
 - `Arrays.equals(...)` : compare le contenu de deux tableaux à une dimension
 - Les tableaux ont la même taille, les éléments sont deux à deux égaux et dans le même ordre
 - Les références sont comparées via `equals(...)` et les valeurs via `==`
 - `Arrays.deepEquals(...)` : compare le contenu de deux tableaux de dimension > 1

Egalité de tableaux - Exemple

- Qu'affiche cette séquence ?

```
String s1 = new String("un");
String s2 = new String("deux");
String s3 = new String("trois");
String[] tab1 = new String[]{s1, s2, s3};
String[] tab2 = new String[]{s1, s2, s3};
String[] tab3 = new String[]{new String("un"), new String("deux"), new String("trois")};

System.out.println(tab1==tab2); // idem tab1==tab3 ou tab2==tab3
System.out.println(tab1.equals(tab2)); // dem tab1.equals(tab3) ou tab2.equals(tab3)

System.out.println(Arrays.equals(tab1,tab2));
System.out.println(Arrays.equals(tab1,tab3)); // idem Arrays.equals(tab2,tab3)
```

Remplir un tableau

- On peut remplir un tableau une valeur ou référence donnée :

```
Arrays.fill(leTableau, v)
```

```
Arrays.fill(leTableau, idxDebut, idxFin, v)
```

```
int[] tab = new int[10];  
Arrays.fill(tab, -1);  
System.out.println(Arrays.toString(tab));
```

```
int[] tab = new int[10];  
Arrays.fill(tab, 2, 8, -1);  
System.out.println(Arrays.toString(tab));
```

Trier et chercher

- Pour trier un tableau

- `Arrays.sort(monTableau)`

- Attention pour des tableaux d'objets, les objets doivent être Comparables (comme les String, etc.)

```
int[] tab = new int[]{3,5,1,6,2};
Arrays.sort(tab);
System.out.println(Arrays.toString(tab));
```

- Pour chercher dans un tableau (trié)

- `Arrays.binarySearch(monTableau, element)`

- `Arrays.binarySearch(monTableau, idxDebut, idxFin, element)`

```
int idx= Arrays.binarySearch(tab,5);
if (idx>-1) {
    System.out.println("5 est dans le tableau à la position "+idx);
}
else {
    System.out.println("5 n'est pas dans le tableau");
}
```

Créer une copie d'un tableau

- Pour créer une copie d'un tableau

```
copieTab = Arrays.copyOf(tabOriginal,  
tailleCopie)
```

- Si `tailleCopie < tabOriginal.length`
 - Le tableau est tronqué
- Si `tailleCopie > tabOriginal.length`
 - Le reste du tableau est rempli par la valeur par défaut (ou null)

```
CopieTab = Arrays.copyOfRange(tabOriginal,  
idxDebut, idxFin)
```

```
int[] tab = new int[]{3,5,1,6};  
int[] copie = Arrays.copyOf(tab,2);  
System.out.println(Arrays.toString(copie));  
int[] copie2 = Arrays.copyOfRange(tab,1,3);  
System.out.println(Arrays.toString(copie2));
```