

UEF 1 : Informatique & Programmation

Faculté des Sciences de Nice

DEUG 2001-2002

Jérôme DURAND-LOSE

Jean-Paul ROY

COURS 10

T A B L E A U X

Tableaux multidimensionnels Tableaux d'objets et Collections

10-2

Tableau à plusieurs dimensions

/ déclaration */* `<type> []...[] <identifiant>;`

/ création */* `<identifiant> = new <type> [<taille 0>]...[<taille n-1>];`

/ accès */* `<identifiant> [<indice0>]...[<indice n-1>]`

Avec les valeurs :

```
int [ ] [ ] tab = { { 1, 0, 0, 0, 0, 0 },  
                   { 1, 1, 0, 0, 0, 0 },  
                   { 1, 2, 1, 0, 0, 0 },  
                   { 1, 3, 3, 1, 0, 0 } };
```

1	0	0	0	0	0
1	1	0	0	0	0
1	2	1	0	0	0
1	3	3	1	0	0

tab[0][2]

e.g. : `double [] [] [] donneesMeteo = new double [365] [24] [3];`

N.B. : mieux vaut `ReleveMeteo [] [] donneesMeteo = new ReleveMeteo [365] [24] [3];`

Tableau des notes

On désire manipuler un tableau constant les notes pour
800 étudiants et 10 unités d'enseignement

```
double [ ] [ ] notes = new double [ 800 ] [ 10 ];  
/* Il faut encore rentrer les notes .... */
```

On peut chercher à calculer la moyenne :

- pour un étudiant
- pour une UE
- de toutes les notes

10-4

Moyenne par étudiant ou UE

```
static double moyenneEtudiant ( double[][] tNotes, int etu ) {
    double somme = 0;
    for ( int j = 0 ; j < tNotes[ etu ].length ; j ++ )
        somme += tNotes[ etu ][ j ];
    return somme / tNotes[ etu ].length;
}
```

etu est fixé

```
static double moyenneUE ( double[][] tNotes, int ue ) {
    double somme = 0;
    for ( int i = 0 ; i < tNotes.length ; i ++ )
        somme += tNotes[ i ][ ue ];
    return somme / tNotes.length;
}
```

ue est fixé

10-5

Moyenne sur tout

Pas d'étudiant ou d'UE à indiquer

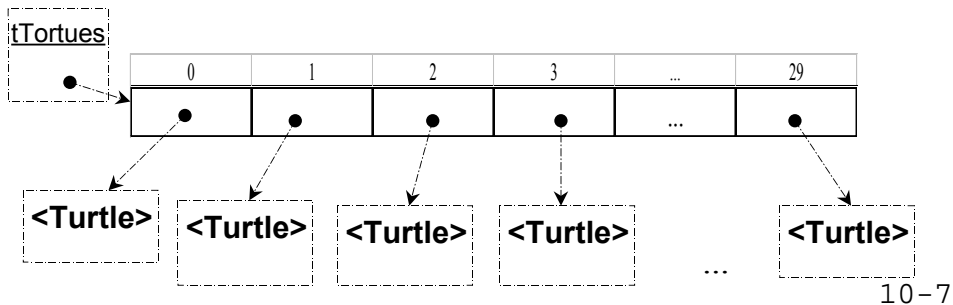
```
static double moyenneGlobale ( double[][] tNotes ) {
    double somme = 0;
    for ( int i = 0 ; i < tNotes.length ; i ++ )
        for ( int j = 0 ; j < tNotes[ i ].length ; j ++ )
            somme += tNotes[ i ][ j ];
    return somme / ( tNotes.length * tNotes[ 0 ].length );
}
```

Il faut sommer sur les étudiants et sur les UE

10-6

Tableau de références (sur des instances)

```
Turtle [] tTortues = new Turtle [ 30 ];
// Il faut encore créer les 30 instances de Turtle
for ( int i = 0 ; i < tTortues.length ; i ++ )
    tTortues[ i ] = new Turtle();
```



10-7

Les faire toutes avancer

```
for ( int i = 0 ; i < tTortues.length ; i ++ )
    tTortues[ i ].forward( Numerik.randomInt( 10, 100 );
```

tTortues => (référence sur un) tableau de ...

tTortues[i] => (référence sur l'instance de) Turtle d'indice i

=> on peut lui envoyer un message

=> tTortues[i].forward(...)

10-8

Celles ayant le « pen down »

Les compter ?

On parcourt le tableau
en les comptant

```
int nombre = 0;
for ( int i = 0 ; i < tTortues.length ; i ++ )
    if ( tTortues[ i ].penDown() )
        nombre ++;
```

La première ayant le « pen down » ?

On parcourt le tableau
jusqu'à en trouver une

```
int indicePenDown = -1;
int i = 0;
while ( ( i < tTortues.length )
        && !(tTortues[ i ].penDown() ) )
    i ++;
if ( i < tTortues.length )
    indicePenDown = i;
```

10-9

Éteindre toutes les machines

```
Machine[] toutesLesMachines = ....
....
for ( int i = 0 ; i < toutesLesMachines.length ; i ++ )
    toutesLesMachines[ i ].eteindre();
```

toutesLesMachines => (référence sur un) tableau de ...

toutesLesMachines[i] => (référence sur la) Machine d'indice i

=> on peut lui envoyer un message

=> toutesLesMachines[i].eteindre()

10-10

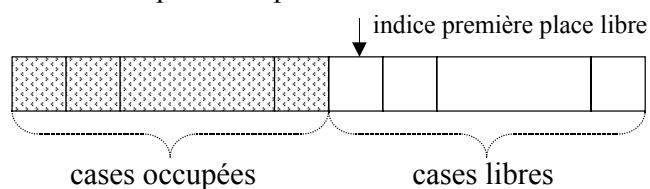
Collection de taille variable

Collection : liste d'éléments où l'on ne se soucie
ni de l'ordre ni des répétitions

On peut être amené à *ajouter* ou *supprimer* un élément de la collection
(apparition ou disparition d'un élément).

Comment faire ? un tableau avec des cases vides (surdimensionné)

Structure : tableau « suffisamment grand »
+ indice première place libre



10-11

Définition du type

type Collection de nombres approchés

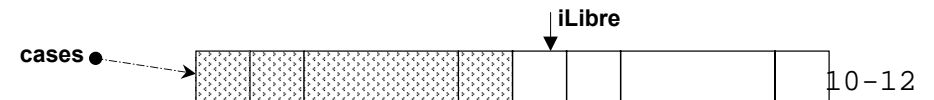
iLibre : entier

cases : tableau de nombres approchés

Algorithme pour afficher le tableau

parcourir les cases d'indice 0 inclus à iLibre exclus
afficher la case courante

```
class CollectionDouble {
    private static final int NBR_CASES = 100;
    private int iLibre = 0;
    private double cases[] = new double[ NBR_CASES ];
    public String toString() { // n'affiche pas mais renvoie une description
        String desc = "CollectionDouble[";
        for ( int i = 0; i < iLibre; i ++ )
            desc = desc + " " + cases[ i ];
        return desc + " ]";
    }
}
```



10-12

Ajouter une valeur

On ajoute
à la fin

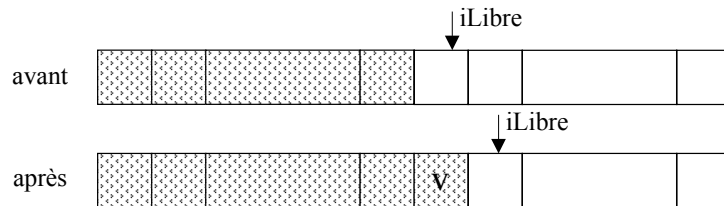
Donnée

valeur : nombre approché

Algorithme pour ajouter une valeur

cases [iLibre] ← valeur
augmenter iLibre de 1

```
void ajouterValeur( double valeur ) {  
    cases[ iLibre ] = valeur;  
    iLibre ++;  
}
```



10-13

Suppression

On récupère la valeur et
on y relogé le dernier
élément

Donnée

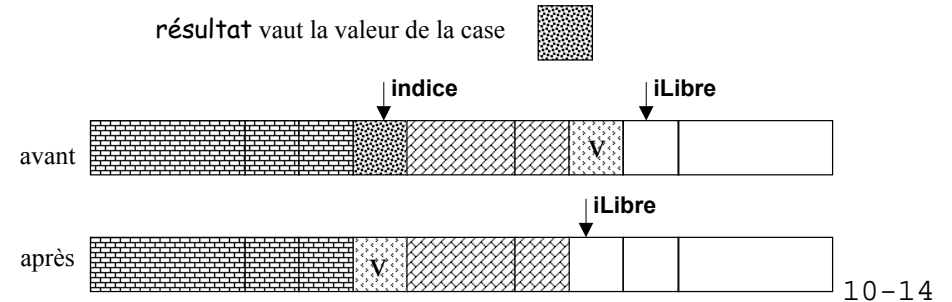
indice : entier

Valeur renvoyée

résultat : nombre approché

Algorithme pour supprimer une valeur

résultat ← cases [indice]
diminuer iLibre de 1
cases [indice] ← cases [iLibre]



10-14

Ce qui correspond à :

```
double retirerIndice( int indice ) {  
    double valeur = cases[ indice ];  
    cases[ indice ] = cases[ iLibre - 1 ];  
    iLibre --;  
    return valeur;  
}
```

Questions en suspens :

- que faire quand il n'y a plus de place ?
- comment retirer une valeur donnée (et non celle dans un case donnée)
- comment retirer toutes les occurrences d'une valeur donnée

10-15