Informatique 1

8. Structures de données

Dr Pierre-André Mudry
pierre-andre.mudry@hevs.ch

Rev 1.65

Contenu cours

Structures de données

- 8.1 Types énumérés
- 8.2 Introduction tableaux
- 8.3 Opérations sur les tableaux

Introduction de `enum`

8.1 TYPES ÉNUMÉRÉS

Problème

- Écrire un programme qui permet d'afficher différents messages en fonction de l'état d'une porte (qui peut être *ouverte*, *fermée*, *cassée* ou dans un état *inconnu*).
- Comment coder cela ?

Solution 1

- Coder l'état avec une constante entière

```
public final int OPEN = 0;
public final int CLOSED = 1;
public final int BROKEN = 2;
public final int UNKNOWN = 3;
int doorState = OPEN;

if(doorState == CLOSED){
   …
}
```

Solution 2

- Coder l'état avec un *String*

```
String doorState = "OPEN";
if(doorState.equalsTo("OPEN"){
    …
}

if(doorState.equalsTo("CLOSED"){
    …
}
…
```

Solution à utiliser

- Un **type énuméré**

Permettent de stocker des énumérations **finies**, c'est-à-dire contenant un nombre fixe (constant) d'éléments

Types énumérés

Exemples

Syntaxe

- Syntaxe de déclaration :

```
enum identifier {VAL1, VAL2, …};
```

- *Exemple :*

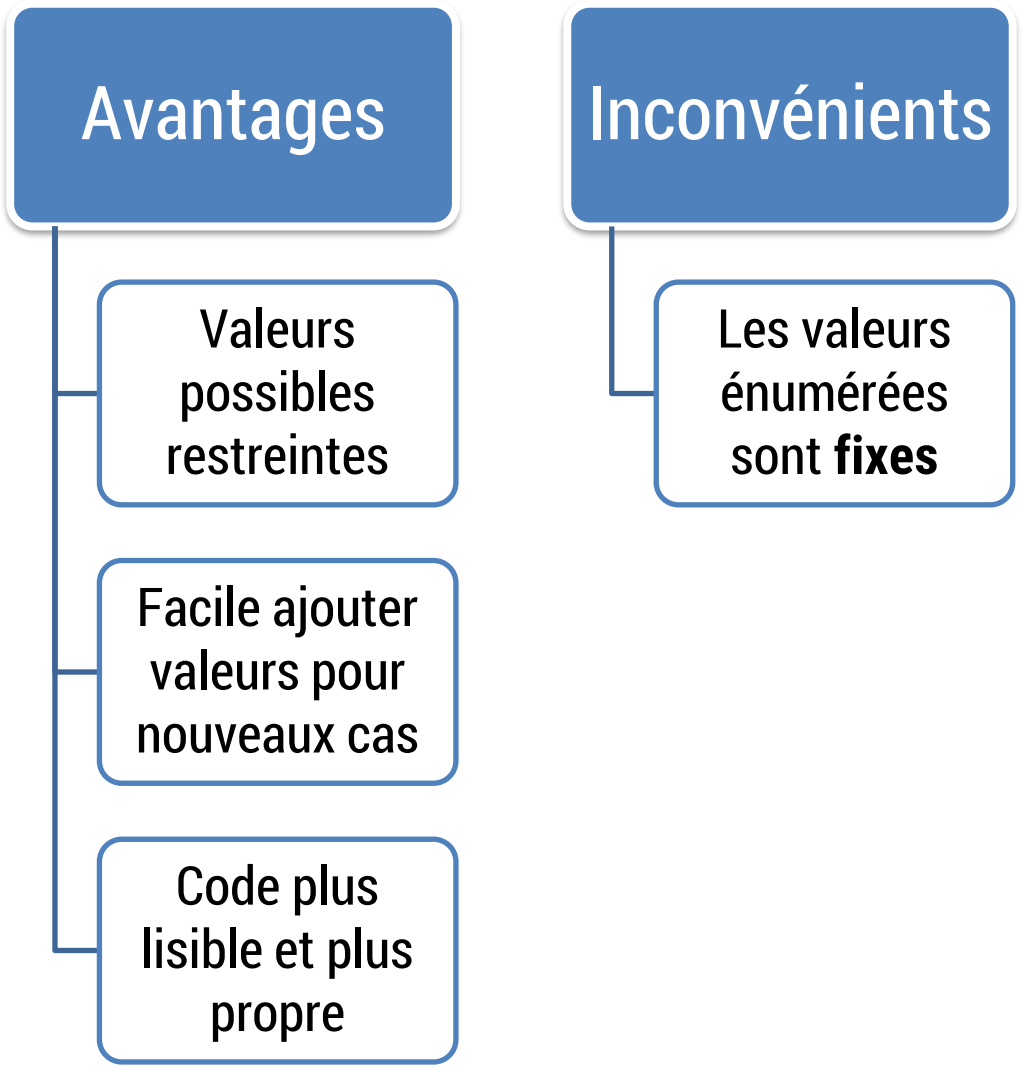
```
enum Sizes {S, M, L, XL, XXL};
```

Les types énumérés (1)

```
public class EnumDemo{
 enum State {OPEN, BROKEN};

 public static void main(String args[]){
  State doorState = State.OPEN;
  switch(doorState){
   case OPEN:
           System.out.println("The door is open");
           break;
   case BROKEN :
           System.out.println("The door is destroyed");
           break;
   default:
           System.out.println("Door state is unknown");
           break;
   }
 }
}
```

Les types énumérés (2)



Exercice

- Déclarer des types énumérés pour :
  - Arôme de glace
  - Races de chat

Little boxes

8.2 TABLEAUX

Les tableaux

Une variable pour stocker plusieurs variables de même type.

Tableau

```
double[] marks = {5, 5.5, 3, 4};
```

Les tableaux statiques (1)

- Syntaxe 1 (init. explicite):

```
TYPE[] arrayName = {v1,v2,v3…};
```

- *Exemple* :

```
int[] anArray = {2, 12, 45}; // With init
```

Les tableaux statiques (2)

- Syntaxe 2 (sans init.):

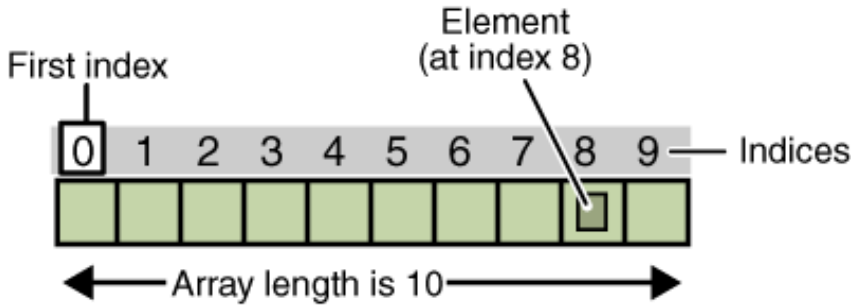
```
TYPE[] arrayName = new TYPE[length];
```

- *Exemple* :

```
double[] a2= new double[100]; // No init.
a2[0] = 23; // Assignation
```

Les tableaux statiques (3)

- Accès par indice



- Le premier élément est *0* (et pas 1)

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

array[0] array[1] array[2]

Modification d'une valeur du tableau

- Changer une valeur dans le tableau se fait avec l'opérateur d'assignation
- On spécifie la position modifiée dans les [ ] après le nom du tableau

```
int[] foo = {5,12,20}; // declares an array of integers

foo[1] = 25; // Now the content of the array is {5,25,20}
```

Exemple de tableaux

```java
int[] anArray; // declares an array of integers

anArray = new int[4]; // allocates memory for 4 integers

anArray[0] = 100; // initialize first element
anArray[1] = 200; // initialize second element
anArray[2] = 300; // etc.
anArray[3] = 400;

System.out.println("Element at index 0: " + anArray[0]);
System.out.println("Element at index 1: " + anArray[1]);
System.out.println("Element at index 2: " + anArray[2]);
System.out.println("Element at index 3: " + anArray[3]);
```

Déclaration et initialisation de tableaux

```
byte[] anArrayOfBytes;
short[] anArrayOfShorts;
long[] anArrayOfLongs;
float[] anArrayOfFloats;
double[] anArrayOfDoubles;
boolean[] anArrayOfBooleans;
char[] anArrayOfChars;
String[] anArrayOfStrings;

// Init
anArrayOfBytes = new byte[10];
anArrayOfShorts = new short[22];
…
```

Taille des tableaux

- Tableaux sont des objets
  - Variable d'instance ***Length***
  - A la création, ***Length*** automatiquement égal à la taille !
  - Taille FIXE

```
Boolean[] anotherArray = {true, true, false}
int x = anotherArray.length;
```

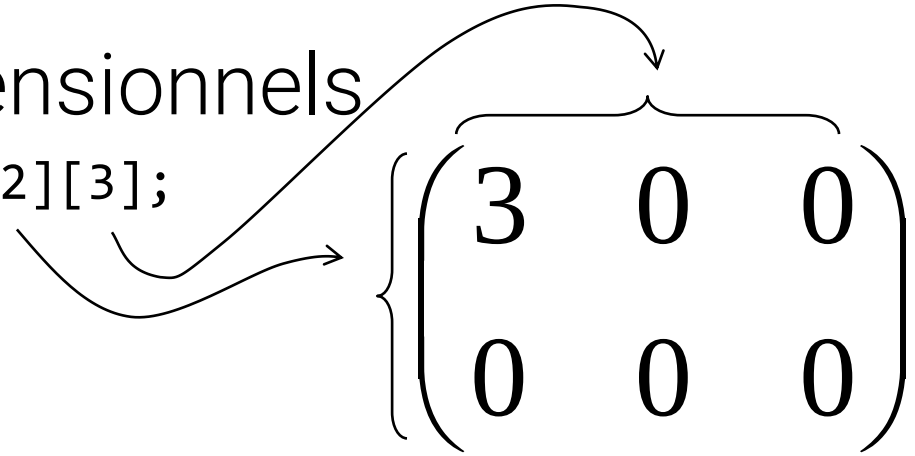
Parcourir, passer en argument et retourner des tableaux

8.3 OPÉRATIONS SUR LES TABLEAUX

Tableaux à plusieurs dimensions

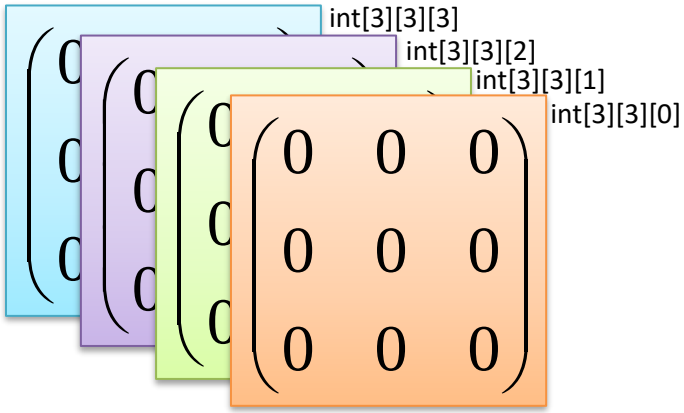
- Tableaux multidimensionnels

```
int[][] array = new int[2][3];
array[0][0] = 3;
```



- Pas limités à 2 dimensions !

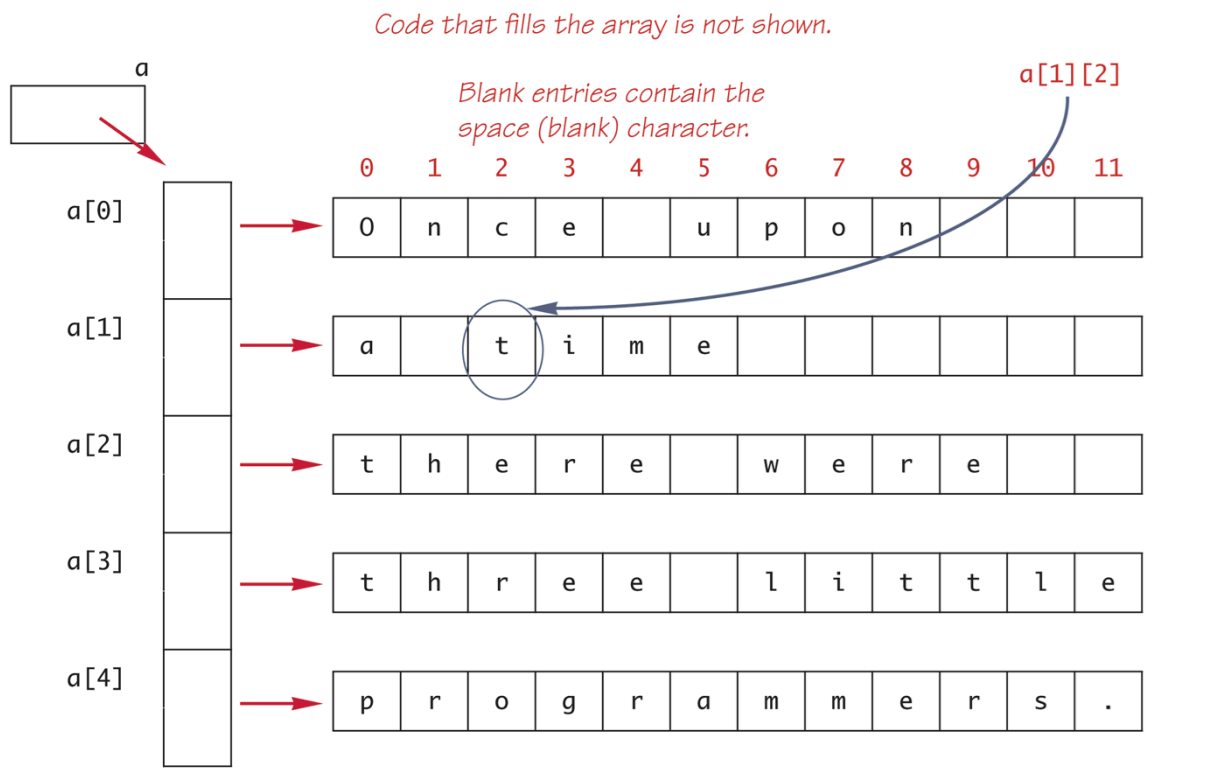
```
int bigArray[][][] = new int[3][3][4];
```



Tableaux à plusieurs dimensions

Display 6.17 **Two-Dimensional Array as an Array of Arrays**

```
char[][] a = new char[5][12];
```



(continued)

Tiré de Savitch, *Absolute Java*, 4ème éd.

Tableaux statiques

- En Java, contrôle dynamique des bornes.

```
int[] anArray = new int[10];
anArray[22] = 34;
```

→Génère une erreur à **l'exécution**

→*ArrayOutOfBounds*

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
Index 22 out of bounds for length 10
at course.arrays.ArrayDemoInClass.main(ArrayDemoInClass.java:12)
```

Parcourir tous les éléments d'un tableau

Taille n

```
for(int i = 0; i < foo.length; i++) {
    Do something with foo[i]
}
```

Taille nxm

```
for(int i = 0; i < foo.length; i++) {
    for(int j = 0; j < foo[i].length; j++) {
        Do something with foo[i][j]
    }
}
```

Taille nxmxq

```
for(int i = 0; i < foo.length; i++) {
    for(int j = 0; j < foo[i].length; j++) {
        for(int k = 0; k < foo[i][j].length; k++) {
            Do something with foo[i][j][k]
        }
    }
}
```

Tableaux en argument de fonction

- Syntaxe standard

```
static double computeAverage(double[] v) {
        double result = 0;

        for (int i = 0; i < v.length; i++) {
                result = result + v[i];
        }

        result = result / v.length;
        return result;
}
```

Tableaux comme valeur de retour

Il est possible de retourner un tableau depuis une fonction

```
TYPE[] Method_Name(Parameter_List)
{
        TYPE[] temp = new TYPE[ArraySize];
        <Some code to fill temp goes here>
        return temp;
}
```

```
public static int[] incrementedArray(int[] a, int increment)
{
        int[] temp = new int[a.length];
        for (int i = 0; i < a.length; i++)
                temp[i] = a[i] + increment;

        return temp;
}
```

Danger : '=' avec tableau

- Un objet tableau est une référence
  - un nom de tiroir mais pas le tiroir lui-même!

- Opérateur = fait que assigner le nom
  - Ne crée pas un nouveau tiroir !
  - Ne copie pas le contenu du tiroir !
  - `a = b` signifie que a *référence* même tiroir que b

Danger: '=' avec tableau

- Comment réaliser une vraie copie ?

'==' avec tableau

- Pour les mêmes raisons, égalité n'est pas contrôlable avec == !

```
double[] a = {1.2, 2.3, 5.3, 4.4};
double[] b = {1.2, 2.3, 5.3, 4.4};
boolean c = a == b;
```

- Vérifier élément par élément avec une boucle !

Conclusion

- Deux structures de données
  - Types énumérés
  - Tableaux