Programmation Orientée Objet
Syntaxe Java

Marc Champesme
mailto:Marc.Champesme@univ-paris13.fr

27 octobre 2020

Les tableaux en Java

- les tableaux sont des objets / instances de classes
- chaque type tableau est un type classe
- les classes représentant des tableaux sont les seules classes de la librairie standard ne possédant pas de documentation (javadoc)
- mais les tableaux sont décrits dans la spécification du language

Exemples de types tableau

Les éléments d'un tableau peuvent être de n'importe quel type Java : aussi bien les types primitifs, que les types classe (y compris les tableaux) :

- `int[]` : classe dont les instances sont des tableaux de `int`
- `String[]` : classe dont les instances sont des tableaux (d'instances) de `String`
- `String[] []` : classe dont les instances sont des tableaux (d'instances) de `String[]`

Création d'instances de tableaux

- `int[] tabInt; // Déclaration d'une variable de type int[], l'instance de tableau n'est pas créée`
- `tabInt = new int[5]; // Création d'une instance de tableau de 5 int, les 5 cases du tableau sont initialisées à la valeur par défaut (i.e. 0)`
- `String[] tabStr = new String[10]; // Création d'une instance de tableau de 10 instances de String, les 10 cases sont initialisées à null`
- `String[][] tabOfTabStr = new String[5][10]; Création d'un tableau de 5 String[], chaque case est initialisée avec un String[] de 10 éléments`

Création d'instances de tableaux (suite)

Erreurs à éviter :

Erreur de compilation !

```
int[5] tabInt; // la taille du tableau ne peut être fixée
qu'à la création de l'instance !!
```

Erreur à l'exécution !

```
int[] tabInt;
tabInt[0] = 574; // l'instance de tableau n'a pas encore
été créée !!
```

Attribut des classes tableaux

Les tableaux possèdent un attribut `length` dont la valeur est la taille du tableau, cet attribut est `final` = non modifiable

L'attribut `length` est `final`

```
int[] tabInt = new int[5]; // tabInt.length == 5
tabInt.length = 10; // Interdit !! length est final
```

Enum Java vs C

Enum en C

Exemples

```
typedef enum {CARREAU,
COEUR, PIQUE, TREFLE}
Famille;
typedef enum {ROUGE, NOIR}
Couleur;
```

Les enum sont des int

```
Famille f = COEUR;
f = ROUGE; // Autorisé !!
f = 23; // Autorisé !!
```

Enum en Java

Exemples

```
public enum Famille {
CARREAU, COEUR, PIQUE,
TREFLE}
public enum Couleur {ROUGE,
NOIR}
```

Les enum sont de VRAIS types classe

```
Famille f = Famille.COEUR;
f = Couleur.ROUGE; //
INTERDIT !!
```

Les types enum Java sont des classes

Les constantes d'un enum sont LES instances de la classe :

`Couleur` est une classe dont les deux UNIQUES instances sont `Couleur.ROUGE` et `Couleur.NOIR`.

Les classes enum ont des méthodes :

- `Famille.CARREAU.ordinal() == 0` / `Famille.COEUR.ordinal() == 1`
- `Famille.CARREAU.name()` vaut `"CARREAU"`
- `Famille.COEUR.compareTo(Famille.TREFLE)` : comparaison selon valeur de `ordinal()`
- `Famille.valueOf("COEUR") == Famille.COEUR`
- `Famille.values()` renvoie un tableau contenant les 4 instances de `Famille`

Classe String

Une classe presque comme les autres, sauf :

- création d'instance : `String s = new String("Bonjour")` équivalent à `String s = "Bonjour"`
- Opérateur de concaténation : `s = "Bon" + "jour" // s reçoit "Bonjour"`
- `String` est une classe non modifiable... mais ce n'est pas la seule

Nombreuses méthodes (cf. javadoc) dont `length()` qui renvoie la longueur de la chaîne.

Égalité entre chaînes de caractères

String nom1, nom2, nom3 ;
nom1 = new String("Durand") ;
nom2 = new String("Durand") ;
nom1 == nom2 est false
nom3 = nom1 ;
nom3 == nom1 est true

Heureusement la méthode equals est là !

`nom1.equals(nom2) est true`

Égalité entre objets

- La méthode `boolean equals(Object obj)` est définie pour toute classe (elle est héritée de la classe Object)
- Par défaut elle se comporte comme ==
- . . . mais on peut la redéfinir pour comparer les caractéristiques des objets

Redéfinition de equals

```
public class Point {
    private int x;
    private int y;

    // Le reste du code est omis

    public boolean equals(Object obj) {
        if (!(obj instanceof Point)) {
            return false;
        }
        Point p = (Point) obj;
        return getX() == p.getX() && getY() == p.getY();
    }
}
```