Les tableaux

Mise à niveau Java

Y. Boichut & F. Moal

Les tableaux - des objets

En Java, les tableaux sont considérés comme des objets :

- les variables de type *tableau* stockent comme valeur une adresse
- les tableaux sont créés par l'opérateur **new**
- ils possèdent une variable d'instance **length** (final)
- ils héritent des méthodes d'instance de la classe **Object**

ou presque...

- Déclaration : la taille n’est pas fixée
  - int[] tabEntiers
- Création : la taille est obligatoire
  - tabEntiers = **new** int[10]
- La taille spécifiée est définitive pour l’objet créé
- Initialisation à la volée (nécessairement à la déclaration)
  - Figure[] figures = {new Carre(6), new Cercle(6)};
  - double[] notes = {2, 3, 4, 10};
  - notes = new double[]{1,2,3};

Utilisation des tableaux

- Affectation des cases - tableau indexé de 0 à sa taille -1
  - notes[0]=1;
- Accès à la taille déclarée
  - notes.length
- Déclaration dans la signature d’une méthode - identique à la déclaration dans un bloc
  - double[] m(int[] monTableau)

Afficher les éléments d'un tableau

- La méthode toString() prédéfinie ne suffira malheureusement pas
  - Par exemple : *System.out.println((new double[]{1,2,3}).toString())* n'affichera pas *[1, 2, 3]*
- Par contre, utilisation de la méthode statique de la classe Arrays
  - *Arrays.toString((new double[]{1,2,3}))* aura l'effet attendu
- Parcours par une boucle
  - *for (double x : notes) {System.out.println(x+" ");}*
  - ou *for (int i = 0; i< notes.length;i++) {System.out.println(notes[i]+" ");}*
  - *ou ...*

D’autres méthodes/fonctions de la classe Arrays

- *Arrays.equals(t1,t2)* retourne vrai si les deux tableaux contiennent des valeurs identiques à chaque indice
  - Evidemment comme pour *toString()*, *t1.equals(t2)* ne retourne vrai que si t1 et t2 référencent la même adresse
- *Arrays.sort(t)* trie un tableau
- *Arrays.binarySearch(t,e)* effectue une recherche dichotomique de l’élément e dans t
- *Arrays.fill(t,v)* remplit le tableau t avec la valeur v
- Plus d’informations à *https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html*

Tableaux à plusieurs dimensions

- Déclaration

  `double[][] notes;`

- Chaque élément du tableau notes contient une référence vers un tableau de *doubles*
- *Création*
  - *notes = new notes[5][6]; /*matrice à dimensions fixes*/*
  - *notes = new notes[5][ ]; /*matrice avec une dimension variable*/*
- *Initialisation*
  - *double[ ][ ] notes = {{1,2,3}, {4,5}};*
  - *notes = new double[ ][ ]{{1,2,3}, {4,5}}*
- *Affectation*
  - *notes[0][2] = 3;*

Quelques méthodes spécifiques de la classe Arrays

- Arrays.deepEquals(t1,t2) : permet de tester l'égalité sémantique de deux tableaux à plusieurs dimensions
- Arrays.deepToString(t) : permet de construire une chaîne de caractère représentant le contenu d'un tableau à plusieurs dimensions

Jeu 1 : compilo

```
class Books {
    String title;
    String author;
}
```

```
class BooksTestDrive {
    public static void main(String [] args) {
        Books [] myBooks = new Books[3];
        int x = 0;
        myBooks[0].title = "The Grapes of Java";
        myBooks[1].title = "The Java Gatsby";
        myBooks[2].title = "The Java Cookbook";
        myBooks[0].author = "bob";
        myBooks[1].author = "sue";
        myBooks[2].author = "ian";
        while (x < 3) {
            System.out.print(myBooks[x].title);
            System.out.print(" by ");
            System.out.println(myBooks[x].author);
            x = x + 1;
        }
    }
}
```

Jeu 2 : compilo

```
class Hobbits {
     String name;
     public static void main(String [] args) {
          Hobbits [] h = new Hobbits[3];
          int z = 0;
          while (z < 4) {
               z = z + 1;
               h[z] = new Hobbits();
               h[z].name = “bilbo”;
               if (z == 1) {
                    h[z].name = “frodo”;
               }
               if (z == 2) {
                    h[z].name = “sam”;
               }
               System.out.print(h[z].name + “ is a “);
               System.out.println(“good Hobbit name”);
          }
```

Exercice - plateau démineur

Ecrire une classe *Plateau* avec un champ *int[ ][ ] plateau*. Le constructeur prendra deux arguments i.e. la largeur et la longueur du plateau.

- Définir la méthode *initialiser(int x)* permettant d'initialiser correctement le plateau i.e. *plateau[i][j]* sera un entier spécifiant le nombre de bombes dans son voisinage direct ou -1 si c'est une bombe. Cette méthode devra tirer aléatoirement le placement des x bombes. A chaque placement, nous incrémenterons le voisinage direct. Après le placement de la dernière bombe, nous aurons alors un plateau correct.
- Définir un affichage terminal visuellement acceptable de notre plateau.