Programmation orientée objet

Chapitre2: Éléments de base du langage JAVA

Variables et Objets

En Java, il existe 3 catégories de types :

1- les types de base (types primitifs)

2- le type Classe

3- le type tableau.

Les types de base (en nombre de 8 plus le type **`void`**) sont les seuls types statiques (non dynamiques). Les Objets et les tableaux sont toujours et implicitement des données dynamiques créées par l'intermédiaire du mot clé **`new`**.

Types de base

Le langage Java offre 8 types de base appelés aussi primitives ou **types primitifs**. Le mot clé **`void`** n'est plus un type, il est utilisé uniquement pour définir les procédures.

Type	Désignation	Plage de valeurs
byte	Entier signé 1 octet	-128 à 127
short	Entier signé 2 octets	-32768 à 32767
int	Entier signé 4 octets	-2^{31} = -2147483648 à $2^{31}-1$ = 2147483647
long	Entier signé 8 octets	-2^{63} = -9223372036854775808 à $2^{63}-1$ = 9223372036854775807
float	Réel sur 4 octets	$1.4*10^{-45}$ → $3.4*10^{38}$
double	Réel sur 8 octets	$4.9*10^{-324}$ à $1.7*10^{308}$
char	Caractère 2 octets	Code en Unicode: **char** c = 'x'; ou **char** c = '**\u**0058';
boolean	Type logique	**true** et **false**

Classes et Objets

1- Création

La création des objets est toujours réalisée par l'intermédiaire de l'opérateur new :

```
NomDeClasse objet;
objet = new NomDeClasse(paramètres d'un constructeur);
```

Ou encore:

```
NomDeClasse objet = new NomDeClasse(paramètres d'un constructeur);
```

Exemple :

```
String S = new String("ceci est une chaîne de caractères");
```

Classes et Objets

2- Copie

Un objet peut aussi référencer l'adresse d'un objet existant. Les deux objets désigneront alors la même information. Cependant la destruction de l'un des objets ne causera pas la destruction de l'autre.

```
NomDeClasse objet1 = new NomDeClasse(paramètres d'un constructeur);
NomDeClasse objet2 = objet1;
```

Exemple :

```
String S1 = new String("ceci est une chaîne de caractères");
String S2 = S1;
```

Les chaînes de caractères constituent un cas particulier. Elles peuvent en plus être initialisées à des constantes chaînes :

```
S2 = "abc"
```

Classes et Objets

3- Destruction

Un objet en mémoire est détruit automatiquement par l'intermédiaire du Garbage Collector. Aucune destruction explicite n'est alors nécessaire. Cependant, la constante **`null`** peut être affectée à un objet pour supprimer le lien avec l'espace mémoire qu'il référençait auparavant. Ce qui entraînera la libération de la mémoire si celle-ci n'est pas référencée par un autre objet.

```
objet = null;
```

Les tableaux

1- Déclaration

```
Type NomDuTableau[] ;
```

- Avec Type est l'un des deux cas suivant :
  - un type primitif
  - une classe

Exemple

```
int T1[] ;

Vector T2[] ;

String T3[] ;
```

Les tableaux

2- Création du tableau

```
NomDuTableau = new Type[taille] ;
```

Exemple

```
T1 = new int[20] ;

T2 = new Vector[10] ;

T3 = new String[15] ;
```

Remarque:

Dans le cas des tableaux de classes, seul le tableau est créé mais pas les éléments ce qui nécessite pour le cas de T2 et T3 la création de chaque T2[i] et chaque T3[j]

Les tableaux

3- Création des éléments d'un tableau

- Pour T1 (**tableau de primitifs**) :

```
T1[0] = 34 ;
T1[1] = 45 ;
…
```

- Pour T2 et T3 (**tableaux d'objets**) *: les éléments doivent être créés avant de s'en servir*

```
T2[0] = new Vector() ; T2[0].add(…) ; …
T3[0] = new String("….") ; if (T3[0].equals(…)) …
```

Les tableaux

4- Le tableau est un objet

Il dispose des méthodes héritées de la classe Object, en plus il dispose de la propriété à lecture seule : **length**

int l = T1.length ; ➔ 20

Remarque1 :

Lorsqu'un tableau est créé, il est impossible de le redimensionner.

La solution :

- créer un nouveau tableau avec la taille souhaitée
- copier les éléments du tableau initial
- supprimer ce tableau

Les tableaux

4- Le tableau est un objet

Exemple :

```
int T1[] = new int[20] ;
…
int tmp[] = new int[21] ;
for (int i=0 ; i<T1.length ; i++) tmp[i] = T1[i] ;
tmp[20] = nouvelle valeur;
T1 = tmp;
```

Les tableaux

4- Le tableau est un objet

Remarque2 :

1- Les 2 écritures suivantes sont équivalentes :

```
Type T[] ;
```

↔

```
Type []T ;
```

2- Une fonction peut retourner un tableau:

```
int []nomDeMethode(…) {
…
}
```

Les tableaux

5- Initialisation des tableaux

Un tableau peut être initialisé lors de sa déclaration :

- *Cas d'un tableau de primitifs :*

int T1[] = {3, 5, 7, 34} ;⬅➔ int T1[] = new int[4] ;

T1[0] = 3 ; T1[1] = 5 ; T1[2] = 7 ; T1[3] = 34 ;

- *Cas d'un tableau d'objets :*
  - *1ère Solution :*

Vector v1 = new Vector() ;

Vector v2 = new Vector() ;

…

Vector T2[] = {v1, v2, …} ;

Les tableaux

5- Initialisation des tableaux

- *2ème Solution (pratique) :*

```
Vector T2[] = {
new Vector(), // ce sont des objets créés à la volet (objets anonymes)
new Vector(),
…
} ;
```

Les tableaux

6- Création de tableaux anonymes

Soit une méthode ayant un tableau comme paramètre :

```
void p1(int T[]) {
…
}
```

- Appeler la méthode :
  - ***1ère solution :***

```
int T1[] = {2, 6, 17} ;
p1(T1) ;
```

  - ***2ème Solution*** : *tableau anonyme*

```
p1(new int[] {2, 6, 17}) ;
```

Syntaxe générale :

```
… new type [] {valeur/objet, valeur/objet, …} …
```

Les tableaux

7- Tableaux à plusieurs dimensions

Type nomDuTableau[][]… ;

Exemple : `int M[][] ;`

Création : `M = new int[3][10] ;` *Accès:* `M[i][j] = ..`

Remarque :

Si on suppose que la 1ère dimension c'est les lignes et la 2ème dimension c'est les colonnes :

- M.length : le nombre de lignes
- M[i].length : le nombre de colonne de la ligne i

Initialisation : `int M [][] = {{2, 3, 4},{1, 45, 6}};`

Les tableaux

7- Tableaux à plusieurs dimensions

➔**Chaque M[i] est un tableau :**

- `M[0]` ➔ `{2, 3, 4}`
- `M[1]` ➔ `{1, 45, 6}`

➔`M.length = 2 et M[0].length = M[1].length = 3`

Résultat : *Chaque ligne de la matrice peut avoir un nombre d'élément différent:*

Exemple :

```
int M [][] = {{2, 3},{1, 45, 6, 3},{3, 7, 9}};
```

Les tableaux

7- Tableaux à plusieurs dimensions

Remarques :

1. Une matrice M est un tableau de tableaux
2. Chaque M[i] est un tableau pouvant être créé séparément
3. La matrice M peut être créée comme un tableau dont les éléments sont créés ultérieurement

Exemple :

```
int M[][] ;
M = new int[3][] ;
…
M[0] = new int[2] ;
M[1] = new int[4] ;
M[2] = new int[3] ;
…
```

TP2

Programmation structurée en java