Cours d'introduction à la programmation (en Java)
Types avancés I (en Java)

Jamila Sam
Jean-Cédric Chappelier
Vincent Lepetit

Faculté I&C

Rencontre du 5e type

À ce stade du cours, la représentation des **données** se réduit aux types élémentaires `int`, `double`et `boolean`.

Ils permettent de représenter, dans des *variables*, des concepts simples du monde modélisé dans le programme :
dimensions, sommes, tailles, expressions logiques, ...

Cependant, de nombreuses données **plus sophistiquées** ne se réduisent pas à un objet informatique élémentaire.

☞ un langage de programmation évolué doit donc fournir le moyen de **composer les types élémentaires** pour construire des types plus complexes, les *types composés*.

Exemples de données structurées

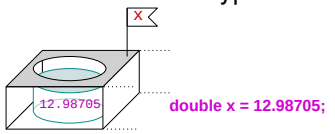
Âge
20
35
26
38
22

Nom	Taille	Âge	Sexe
Dupond	1.75	41	M
Dupont	1.75	42	M
Durand	1.85	26	F
Dugenou	1.70	38	M
Pahut	1.63	22	F

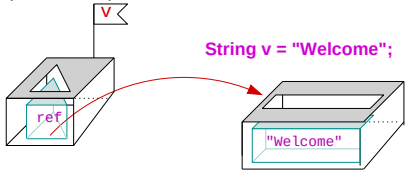
- tableaux
- structures de données hétérogènes
  (par exemple, « un enregistrement » dans le tableau de droite ci-dessus)
- chaînes de caractères
  (par exemple, le « nom »)
- ...

Types de base et types évolués

- Toute variable de type de base stocke directement **une valeur** :



- Toute variable de type évolué, comme les tableaux ou les chaînes de caractères (`String`) que vous allez voir dans ce cours, stocke **une référence** (adresse) vers une valeur :



Types de base et types évolués (2)

- Toute variable de type de base stocke directement **une valeur**
- Toute variable de type évolué stocke **une référence** vers une valeur

☞ **Attention** : ceci a une très grande incidence sur la sémantique des opérateurs `=` et `==` en Java !

☞ Cela a aussi un incidence sur l'affichage

- `double x = 13.5` veut dire « *J'affecte à x la valeur 13.5* »
- `String s = "coucou"` veut dire « *J'affecte à s une référence à la chaîne de caractères coucou* »

En clair, si `v1` et `v2` sont de type évolué :

- `v1 = v2` affecte l'adresse de `v2` à la variable `v1`
- `v1 == v2` compare l'adresse de `v2` avec celle de `v1`
- `System.out.println(v1);` affiche l'adresse de `v1` (dans le cas général)

Nous y reviendrons . . .

Petit exemple introductif

Supposons que l'on souhaite écrire un programme de jeu à plusieurs joueurs.

Score	Écart à la moyenne
1000	-1860
1500	-1360
2490	-370
6450	3590
...	...

Commençons modestement par... deux joueurs.

Solution avec les moyens actuels

```
...
Scanner keyb = new Scanner(System.in);

// Lecture des donnees et calculs
System.out.println ("Score Joueur 1:");
int score1 = keyb.nextInt();
System.out.println ("Score Joueur 2:");
int score2 = keyb.nextInt();
// Calcul de la moyenne
double moyenne = (score1 + score2);
moyenne /= 2;
// Affichages
System.out.println("Score     Ecart Moyenne");
System.out.println(score1 + " " + (score1 - moyenne));
System.out.println(score2 + " " + (score2 - moyenne));
...
```

Comment passer à plus de joueurs ?

☞ utiliser plus de variables

Solution avec les moyens actuels (2)

```
System.out.println ("Score Joueur 1:");
int score1 = keyb.nextInt();
System.out.println ("Score Joueur 2:");
int score2 = keyb.nextInt();
System.out.println ("Score Joueur 3:");
int score3 = keyb.nextInt();
System.out.println ("Score Joueur 4:");
int score4 = keyb.nextInt();
System.out.println ("Score Joueur 5:");
int score5 = keyb.nextInt();

// calcul de la moyenne
double moyenne = (score1 + score2 + score3 + score4 + score5);
moyenne /= 5;
```

Comment faire les affichages ?

Solution avec les moyens actuels (3)

```
System.out.println ("Score Joueur 1:");
int score1 = keyb.nextInt();
System.out.println ("Score Joueur 2:");
int score2 = keyb.nextInt();
...
System.out.println ("Score Joueur 5:");
int score5= keyb.nextInt();

// calcul de la moyenne
double moyenne = (score1 + score2 + score3 + score4 + score5);
moyenne /= 5;


// Affichages
System.out.println("Score         Ecart Moyenne");
for(int i = 1; i <= 5; ++i) {
  System.out.println(scorei + "    " + scorei -  moyenne);
}
```

Solution avec les moyens actuels : limites

Mais

1. comment l'écrire (`scorei` n'est pas correct) ?
2. comment faire si on veut considérer 100, 1000... joueurs ?
3. comment faire si le nombre de joueurs n'est pas connu au départ ?

☞ Solution : les **tableaux**

Solution avec tableau

```
System.out.print ("Donnez le nombre de joueurs:");
int n = keyb.nextInt();
if (n > 0) {
   double moyenne = 0;
   int scores [] = new int[n];

    // Lecture des scores
   for (int i = 0; i < n; ++i) {
      System.out.println ("Score Joueur " + i + " :");
      scores[i] = keyb.nextInt();
      moyenne += scores[i];
   }
   moyenne /= n; // calcul de la moyenne

    // Affichages
   System.out.println(" Score " + " Ecart Moyenne");
   for (int i = 0; i < n ; ++i) {
      System.out.println(scores[i] + " " + (scores[i] - moyenne));
   }
}
```

Les tableaux

Un **tableau** est une collection de valeurs *homogènes*,
c'est-à-dire constitué d'éléments qui sont tous du **même type**.

Exemple : Tableau `scores` contenant 4 `int`

1000	1500	2490	6450
`scores[0]`	`scores[1]`	`scores[2]`	`scores[3]`

On utilise donc les tableaux lorsque *plusieurs* variables de *même* type doivent être *stockées*/mémorisées.

On pourra définir des tableaux d'`int`, de `double`, de `bool`, ...
... mais aussi de n'importe quel autre type à disposition
par exemple des tableaux de tableaux.

Les différentes sortes de tableaux

Il existe en général quatre sortes de tableaux :

		taille initiale connue *a priori* ? non	taille initiale connue *a priori* ? oui
taille pouvant varier lors de l'utilisation du tableau ?	oui	1.	2.
	non	3.	4.

Remarques :

- avec le premier type de tableau (1.), on peut faire tous les autres
  ☞ les autres permettent des *optimisations*
- pratiquement aucun langage de programmation n'offre les 4 variantes

Les tableaux en Java

En Java, on utilise :

		taille initiale connue *a priori* ? non	taille initiale connue *a priori* ? oui
taille pouvant varier lors de l'utilisation du tableau ?	oui	`ArrayList`	`ArrayList`
	non	tableaux de taille fixe	tableaux de taille fixe

Dans un premier temps, nous allons nous intéresser aux

tableaux de taille fixe

dont la taille, en Java, peut se décider avant ou pendant l'exécution, mais qui une fois choisie ne peut plus varier pendant le déroulement du programme.

Viendront, lors du prochain cours, les tableaux dynamiques, dont la taille peut varier pendant l'exécution du programme.