UEF 1 : Informatique & Programmation

Faculté des Sciences de Nice

DEUG 2000-2001

Jérôme DURAND-LOSE

Sandrine JULIA

Jean-Paul ROY

COURS 2

Les types de données en Java

Le langage Java traite deux *types* de données :

- les **données primitives** :
  - les nombres [=> pour le calcul]
  - les booléens [=> pour la logique]
  - les caractères [=> pour les textes]
- les **objets**
  - les « instances de classes »
  - les « tableaux »

???

Les types numériques primitifs

- A la base : le BINAIRE ! Les seuls chiffres : 0 et 1
- Un chiffre 0 ou 1 se nomme un BIT.

Binary digIT...

- Exemple : $\overleftarrow{10010_2}$ en binaire se décode, ici à l'envers :
  $\mathbf{0}*2^{\mathbf{0}} + \mathbf{1}*2^{\mathbf{1}} + \mathbf{0}*2^{\mathbf{2}} + \mathbf{0}*2^{\mathbf{3}} + \mathbf{1}*2^{\mathbf{4}} == 2 + 16 == 18$
- Nombres positifs sur 8 bits : de 00000000_2 à 11111111_2, donc dans l'intervalle [0,255].

2^8-1

- D'où **plusieurs types numériques** suivant le nombre de bits occupés par un nombre en mémoire :

Nombres entiers				**Nombres approchés**	
byte	*short*	*int*	*long*	*float*	*double*

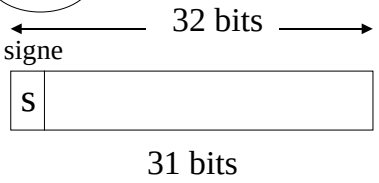
sur 32 bits, donc dans :

`[-2147483648, 2147483647]`

Nombre limité de chiffres après la virgule ==> pertes possibles de précision !

Ex: `-25.6876548702549`

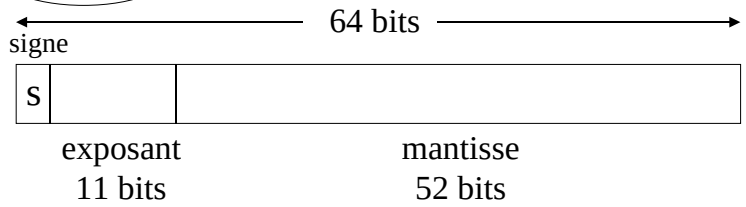
int



$x \in [-2^{31}, 2^{31}-1]$

double

$|x| \in \{0\} \cup [4.9*10^{-324}, 1.8*10^{308}]$



Un programme avec trois variables !

```
/*
    Convertisseur francs -> euros
*/
class EuroConv
{  public static void main(String[] args)
   {  double sommeFr = 3502.28;
      double euro = 6.55957;
      double sommeEu = sommeFr / euro;
      System.out.print(sommeFr);
      System.out.print(" francs = ");
      System.out.print(sommeEu);
      System.out.println(" euros.");
      System.out.println("Au revoir !");
   }
}
```


Compilation puis exécution !

```
3502.28 francs = 533.9191440902376 euros.
Au revoir !
```

Votre disquette contient dès lors deux fichiers :

`EuroConv.java` *et* `EuroConv.class`

Dissection de [ce programme] cette classe

- On retrouve l'architecture de la classe **Essai1**, précédée d'un commentaire :

```
/*
    titre, auteur, date, version, but,...
*/

class <nomDeClasse>
{  public static void main(String[] args)
   {  <instructions de la méthode main>
   }
}
```

♦ On **déclare** [en les initialisant] **trois variables** de type *double*, dont les valeurs seront des nombres approchés :

```
double sommeFr = 3502.28;
double euro = 6.55957;
double sommeEu = sommeFr / euro;
```

♦ L'ordre a de l'importance, même s'il n'est pas unique :

```
double sommeEu;                  ← déclaration
double sommeFr = 3502.28;        ← définitions
double euro = 6.55957;           ←
sommeEu = sommeFr / euro;        ← affectation
```


♦ **Déclaration** d'une variable [une seule fois !]:

<type> <nomDeVariable>;

```
int r;
double surfaceCercle;
```

♦ **Affectation** d'une variable [déjà déclarée !] :

<nomDeVariable> = <expression>;

```
r = 10;
r = (r + 1)*nbFois;
surfaceCercle = pi * r * r;
```


♦ **Définition** d'une variable [une seule fois !]:

<type> <nomDeVariable> = <expression>;

```
int r = 2;
double surfaceCercle = 3.1416 * r * r;
```

Une expression pouvant contenir des variables déjà déclarées

♦ Il faut que le résultat de l'expression soit bien du *type* attendu pour la variable :

```
int longueur;
. . . . . .
longueur = r * 2.5;   ← ERREUR à la compilation !
```


Portée [visibilité] d'une variable

♦ Jusqu'à la fin du dernier « bloc » la contenant [bloc = une suite d'instructions entre accolades].

```
class EuroConv
{  public static void main(String[] args)
   {  double sommeFr = 3502.28;
      double euro = 6.55957;
      double sommeEu = sommeFr / euro;
      System.out.println(sommeFr);
      . . . . . .
   }
}
```

Portée de sommeEu

Opérations numériques primitives

♦ Les opérations de base sont contextuelles [dépendent du type des opérandes] :

+	-	*	/	%

```
25 / 3     ⟶  8                 [dans les entiers]
25 % 3     ⟶  1                 [dans les entiers]
25.0 / 3   ⟶  8.333333333333334
25 % 3.5   ⟶  0.5
25 + 3.0   ⟶  28.0
```


♦ Les opérations plus avancées font appel aux méthodes et constantes de la *classe prédéfinie* `Math` :

```
public class Math
{ public static double sqrt(double x)
  { . . . .}
  public static double pow(double x, double y)
  { . . . .}
  etc
}
```

```
Math.sqrt(2)          ⟶  1.4142135623730951
Math.cos(Math.PI)     ⟶  -1.0
Math.pow(2,3)         ⟶  8.0
```


Les ajustements [automatiques] de types

♦ Est-ce que 2 est un *double* ? Oui, s'il le faut :

```
Math.sqrt(2)
```

bien que la méthode `Math.sqrt(…)` attende un *double* !…

♦ Ainsi 2 est ajusté [converti] en 2.0

♦ Autre exemple : 3 + 2.5 ----> 3.0 + 2.5 ----> 5.5

♦ Ce sont des **ajustements automatiques**. On essaie de perdre le moins d'information possible : il aurait été malvenu de convertir 2.5 en un *int*...

Les ajustements [manuels] de types

♦ Comment obtenir un *int* comme résultat de 2^3 ?...

```
Math.pow(2,3)  ⟶  8.0
```

♦ En forçant un *ajustement de type* :

```
double d = Math.pow(2,3);
int x = (int) d;
System.out.print(x);  ⟶  8
```

♦ Bien sûr, on perd de l'information :

```
(int) 2.5   ⟶  2
(double) 2  ⟶  2.0
```

Affichage à l'écran avec `System.out`

```
int x = 5;
System.out.print(x);
System.out.println("La valeur de x est : " + x);
```

une "chaîne de caractères"
[un texte <==> *String*]

ajustement automatique de `x` vers une *String*. Les deux chaînes sont alors *concaténées* et la chaîne résultat est affichée...