Informatique 1

3. Types et variables

Dr Pierre-André Mudry
pierre-andre.mudry@hevs.ch

Rev 1.45

Objectifs de ce cours

Comprendre les notions de type et de variable

- Représentation binaire des nombres
- Typage et valeurs
- Variables et constantes
- Portée des variables

Typage

Type = ensemble de valeurs possibles (acceptables) pour une donnée

Définition

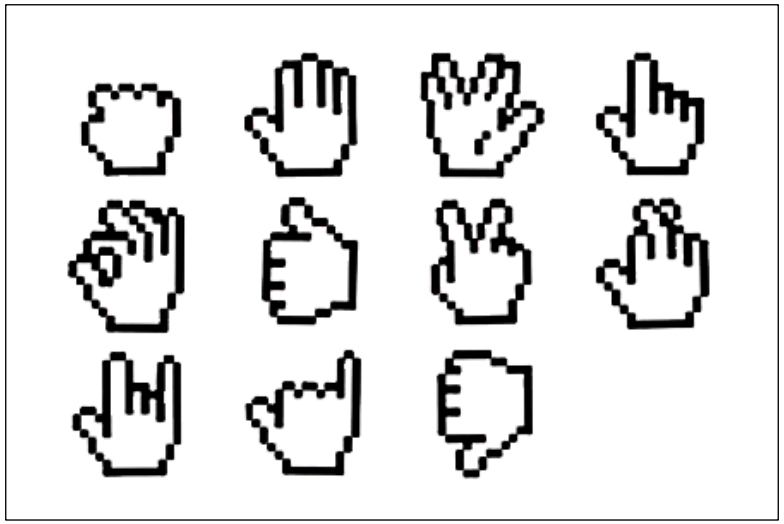
L'âge d'une personne :

Couleur de voiture :

Températures sur Terre :

Types de données

- Toutes les informations dans un ordinateur sont traitées en binaire, c'est-à-dire sous forme de séquences 1 et de 0 en mémoire



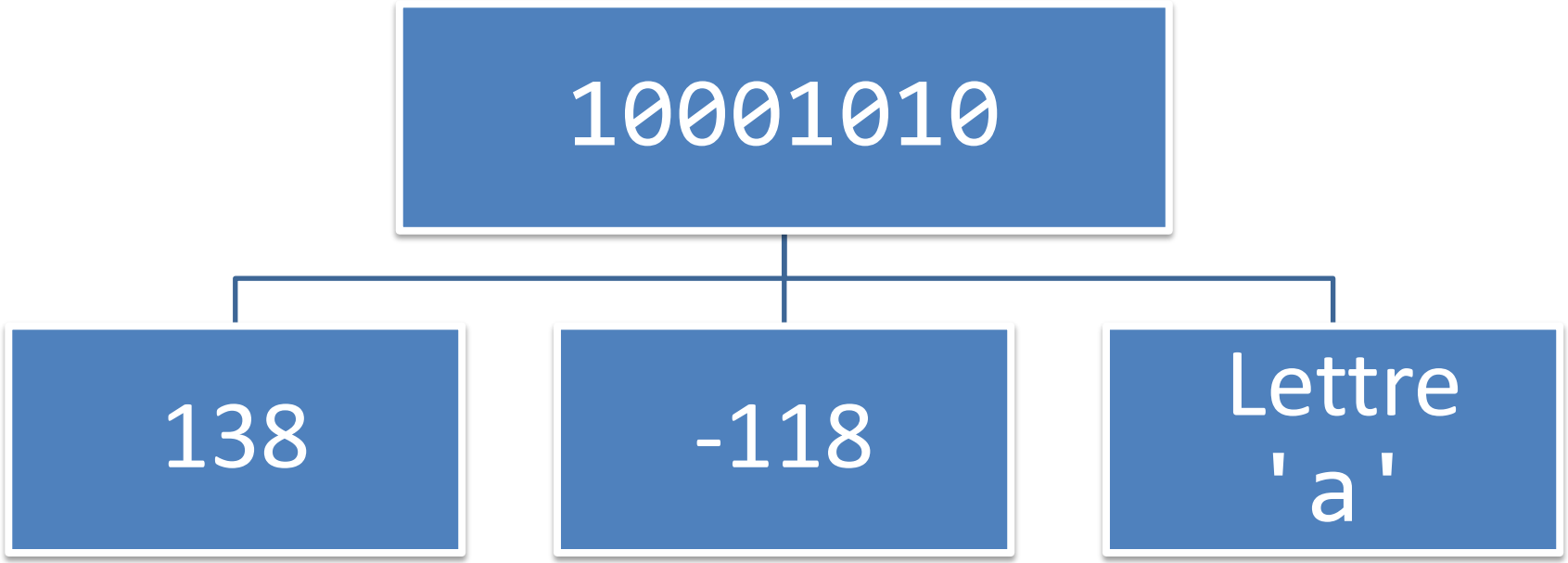
Types en Java

- *Java* est un langage fortement typé
  - Toutes les données ont un type
  - Contrôlé à la compilation [*et à l'exécution*]

Bits everywhere

3.1 REPRÉSENTATION BINAIRE DES NOMBRES

La représentation binaire



- Le rôle du typage est d'enlever cette ambiguïté

Comment cela fonctionne-t-il ?

Nombre binaires

Décimal

Nombres formés avec les symboles :

`0, 1, 2, 3, 4, 5, 6, 7, 8, 9`

Binaire

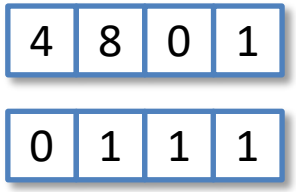
Nombres formés avec les symboles :

`0, 1`

- Un nombre est formé d'une somme de puissances d'une base

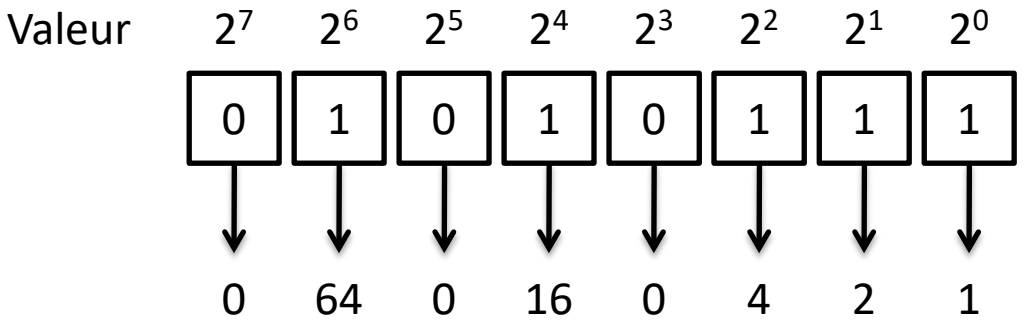
$4801_{10} = 4*10^3 + 8*10^2 + 0*10^1 + 1*10^0$

$0111_2 = 0 * 2^3 + 1*2^2 + 1*2^1 + 1*2^0$



Conversion binaire → décimal

- Convertir 01010111_2 en décimal



1 + 2 + 4 + 16 + 64 = 87

Exercice :

$0000\ 0101_2$? _____

$1110\ 0111_2$? _____

$1010\ 0101_2$? _____

Décimal	Binaire
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
…	…

Notation hexadécimale

- Base 16 : symboles possibles

`0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f`

- 1 nombre hexa = 4 bits

Binaire	Hexadécimal
0b0000	0x0
0b0001	0x1
0b0010	0x2
0b0011	0x3
0b0100	0x4
0b0101	0x5
0b0110	0x6
0b0111	0x7

Binaire	Hexadécimal
0b1000	0x8
0b1001	0x9
0b1010	0xa
0b1011	0xb
0b1100	0xc
0b1101	0xd
0b1110	0xe
0b1111	0xf

Notations informatique

- Attention ! Il est important de savoir en quelle base les nombres sont exprimés.

11_{16} != 11_{10} != 11_{2}

  - Notation mathématique (avec indice)
    - 1010_2 , 1233_{10}
  - Notation informatique
    - Décimal : *1234* (rien de particulier)
    - Binaire : *0b1010*
    - Hexadécimal : *0xAFCE ou 0xafce* (pas sensible casse)

The relation between bits and programming

3.2 TYPES ET VALEURS LITTÉRALES

Typage et mémoire

- Le typage spécifie, entre autres, le nombre de bits qu'une variable nécessite en mémoire
- Dans (presque) tous les langages : unité de base

`byte` ≡ 8 bits

Types entiers

- En *Java*, tailles des types

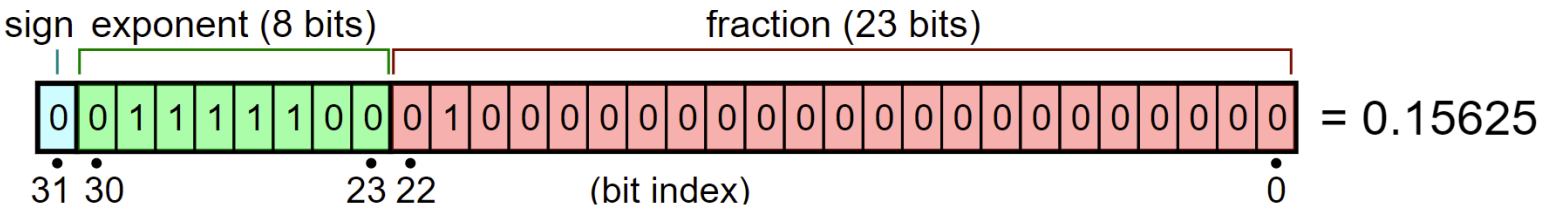
`boolean`	1 bit (valeurs *true* or *false*)
`byte`	8 bits
`short`	16 bits (2^{16} valeurs possibles)
`int`	32 bits (pour *integer*)
`long`	64 bits

- Tous sont signés (sauf **`boolean`**)

Nombres réels

- **`float`** et **`double`** (simple ou double précision)
- Aussi du binaire, mais avec mantisse et exposant
  - *Float* : 23 bits mantisse, 8 bits exposant, 1 bit signe
  - *Double* : 52 bits mantisse, 11 bits exposant, 1 bit de signe
- Représentation *approximative* nombres entiers

Float representation



Source https://en.wikipedia.org/wiki/Single-precision_floating-point_format

$$\text{value} = (-1)^{\text{sign}} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right) \times 2^{(e-127)}$$

$$\text{sign} = b_{31} = 0$$

$$1 + \sum_{i=1}^{23} b_{23-i} 2^{-i} = 1 + 1 \cdot 2^{-2} = 1.25$$

$$2^{(e-127)} = 2^{124-127} = 2^{-3}$$

Nombres réels (2)

- Nombres à virgule représentables (approximativement)
  - Type `float` dans la zone $\pm 2^{127}$, en gros 7 chiffres après la virgule
  - Type `double` dans la zone $\pm 2^{1023,}$ en gros 16 chiffres après la virgule

- Attention! Séparateur est le . et non ,
  - 3,14 n'existe pas, il faut utiliser 3.14

Caractères et booléens

- Pour représenter des valeurs binaires
  - Type **`boolean`** (valeur possible **`true`** ou **`false`**)

- Pour représenter des caractères
  - Type **`char`**

Caractères (2)

- ASCII

Dec	Hex	Sym	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	32	20		64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	TAB	41	29	)	73	49	I	105	69	i
10	A	LF	42	2A	*	74	4A	J	106	6A	j
11	B	VT	43	2B	+	75	4B	K	107	6B	k
12	C	FF	44	2C	,	76	4C	L	108	6C	l
13	D	CR	45	2D	-	77	4D	M	109	6D	m
14	E	SO	46	2E	.	78	4E	N	110	6E	n
15	F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	▯

Caractères en Java

- **`char`** sur 16 bits (caractères *Unicode*)

"Unicode provides a unique number for every character,
no matter what the platform,
no matter what the program,
no matter what the language.

Where to store data

3.3 VARIABLES ET CONSTANTES

Variables

Variable = emplacement dans la mémoire

Définition

Caractéristiques

Nom	• *Identificateur* (vitesse, age…)
Taille	• *Type* (par exemple `int`, `byte`)
Valeur	• 12, 'c', 3.2

Variables (2)

- Avant d'utiliser une variable, il faut la déclarer afin de réserver de l'espace en mémoire.
- La syntaxe est la suivante :

type identifier [= literal]

Exemples

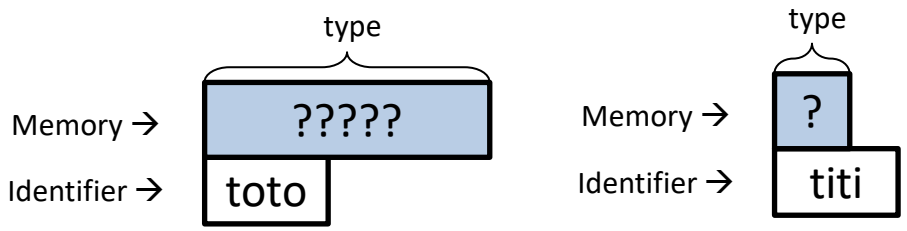
```
boolean checked;
int val = 23;
double salary = 232.2;
```

Variables (3)

- Deux phases dans la création

1.Déclaration

```
int toto;
boolean titi;
```



2.Initialisation

```
toto = 3;
titi = false;
```

Variables (4)

- Déclaration et initialisation :

- Une fois la variable déclarée, il faut l'initialiser (erreur en *Java*, pas en C/C++)

Assignation de variables

- L'assignation (opérateur « = ») permet de donner une valeur à une variable.
- Attention ! Ce n'est pas une égalité au sens mathématique.
- L'assignation remplace l'ancienne valeur

```
toto = 35;
```

signifie `toto` vaut désormais 35

Exemple assignation

```
int foo = 4;
int bar = 5;
foo = bar;
bar = 3;
```

La notion d'état

- Etat = ensemble des variables déclarées à un moment donné
  - Ce qui demeure entre 2 instructions
- Notre programme va influencer l'état par les instructions…. et c'est tout ce qu'il fera

Assignation de variables (2)

- Exemple

```
int i = 3;
i = i + 1;
```

- Processus dynamique :

Exercice : échanger le contenu de deux variables

Assignation nombres réels

```
double foo = 3.2; → ok

float foobar = 3.2; → erreur
float foobar = 3.2f; → ok
```

- Une valeur littérale de type `x.y` : `double`
- Pour forcer l'obtention d'un `float`, il faut ajouter `'f'` à la fin.

Constantes

- Un emplacement mémoire dont la valeur peut être assignée une seule fois
- Par exemple : constantes physiques et mathématiques (π, *c*, …)
- Syntaxe

```
final type identifier [= literal];
```

Exemples

```
final boolean IS_WEEKEND = false;
final float PI = 3.141592f;
```

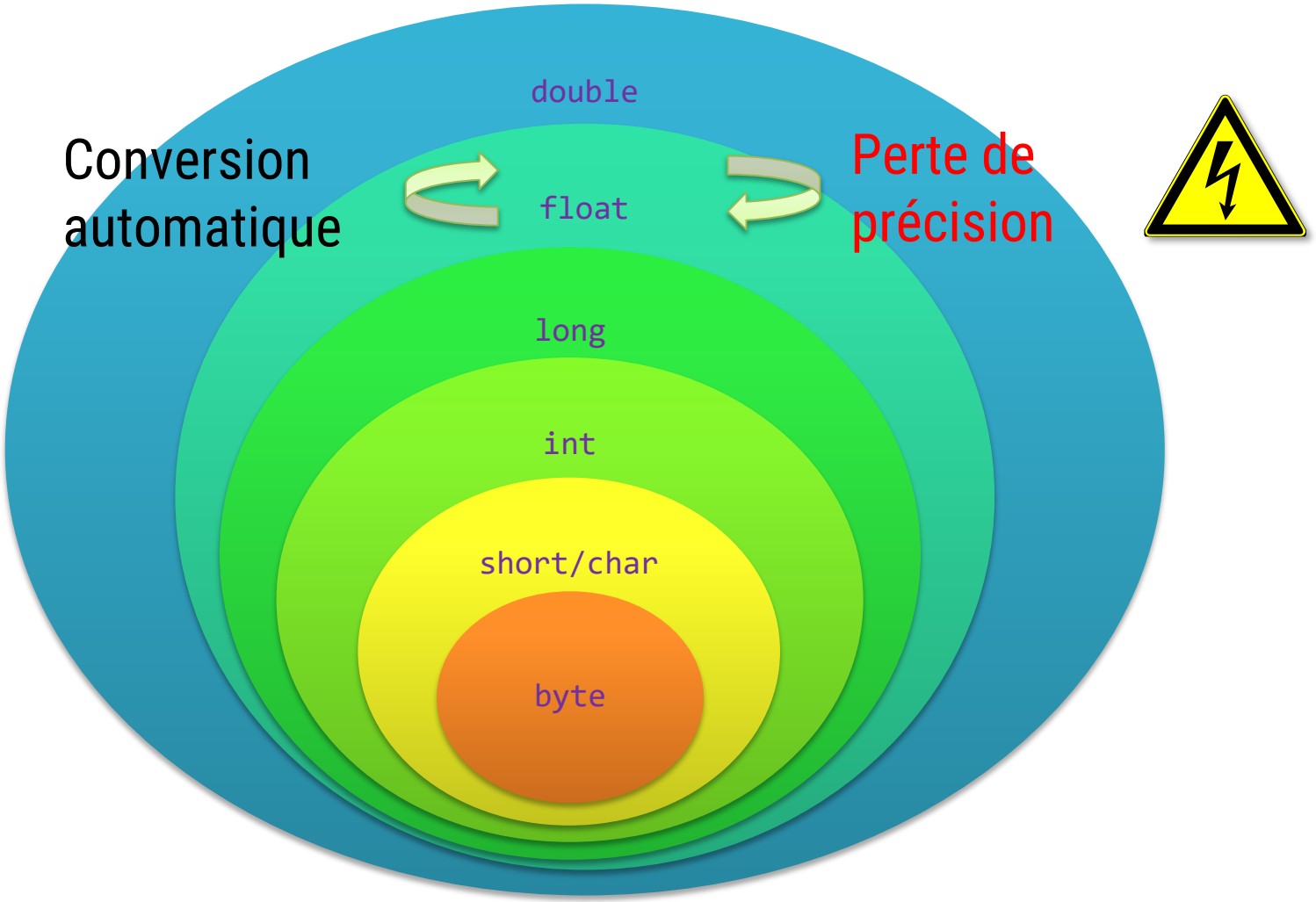
Promotion des types

- *Exemple* : valeurs **`byte`** inclus valeurs **`short`**
  - On peut donc sans problème transformer un **`byte`** en **`short`** → conversion automatique
- Attention à l'inverse !
  - Possible également, avec un *cast* → opérateur **`( )`**

```
int a = 0; double b = 3.2;
a = (int) b; // Precision loss
```

  - Perte de précision → **`a`** vaut 3, pas 3.2 !
  - Si pas de conversion explicite, erreur du compilateur

Compatibilité des types



Types scalaires VS objets

- Tous les types vus jusqu'à présent = scalaires
- Il existe d'autres types en *Java* = objets
  - Programmation objet sujet 2ème semestre
  - Pour l'instant un objet ≈ variable de type spécial
    - Même déclaration, initialisation et assignation.
  - *Java* = langage objet…

Objets et classes en Java

Pour l'instant

- Une classe = un type
- Un objet = une variable de ce type

Par exemple la classe **`String`** peut contenir des chaînes de caractères

```
String name = "Henri";
```

Classe — Objet — Val. initiale

From where to where

3.4 PORTÉE DES VARIABLES

Portée d'une variable

- Portée = durée de vie d'une variable
- Un bloc est le contenu entre { }

La portée d'une variable débute à sa déclaration et se termine une fois le **bloc** terminé.

Définition

```
{
  // A.
  int x = 1;
  // B.
  {
   // C.
   int y = 2;
   // D.
  }

  // E.
}
// F.
```

Quelles variables sont déclarées à quel endroit ?

Portée - Redéfinition

- La redéfinition d'une variable déclarée n'est pas possible en *Java*. Ce n'est pas le cas dans tous les langages.
- Dans un même bloc, il ne peut pas y avoir deux variables de même nom.

Conclusion

- Nous avons vu
  - Le lien entre nombres binaires et variables
  - La notion de type
  - Comment déclarer des variables et des constantes
  - La portée / durée de vie des variables