Premiers pas avec Java

Algorithmique et langage de programmation

Julien Romero, Gaël Thomas

Plan du cours

1. **Mon premier programme Java**
2. Exécution d'un programme Java
3. Variables et types de données
4. Les opérateurs du langage Java
5. Les conversions de type
6. Structures algorithmiques

Dans un programme Java, on trouve

- Des mots clés : des mots qui ont un sens réservé dans le langage : class, if, while, |, &&, do...
- Des symboles : des noms servant à identifier des éléments
  - Constitué de caractères, chiffres (sauf au début), _ ou $
  - La casse est significative (majuscule vs minuscule).
- Des types : spécifie la nature de certains symboles
  - Entier (int), chaîne de caractère (String), flottant (float)...
- Des valeurs littérales : des valeurs ayant un type donné

  42 (entier), "Bonjour, monde!" (chaîne de caractère), 3.14 (flottant)...

Dans un programme Java, on trouve

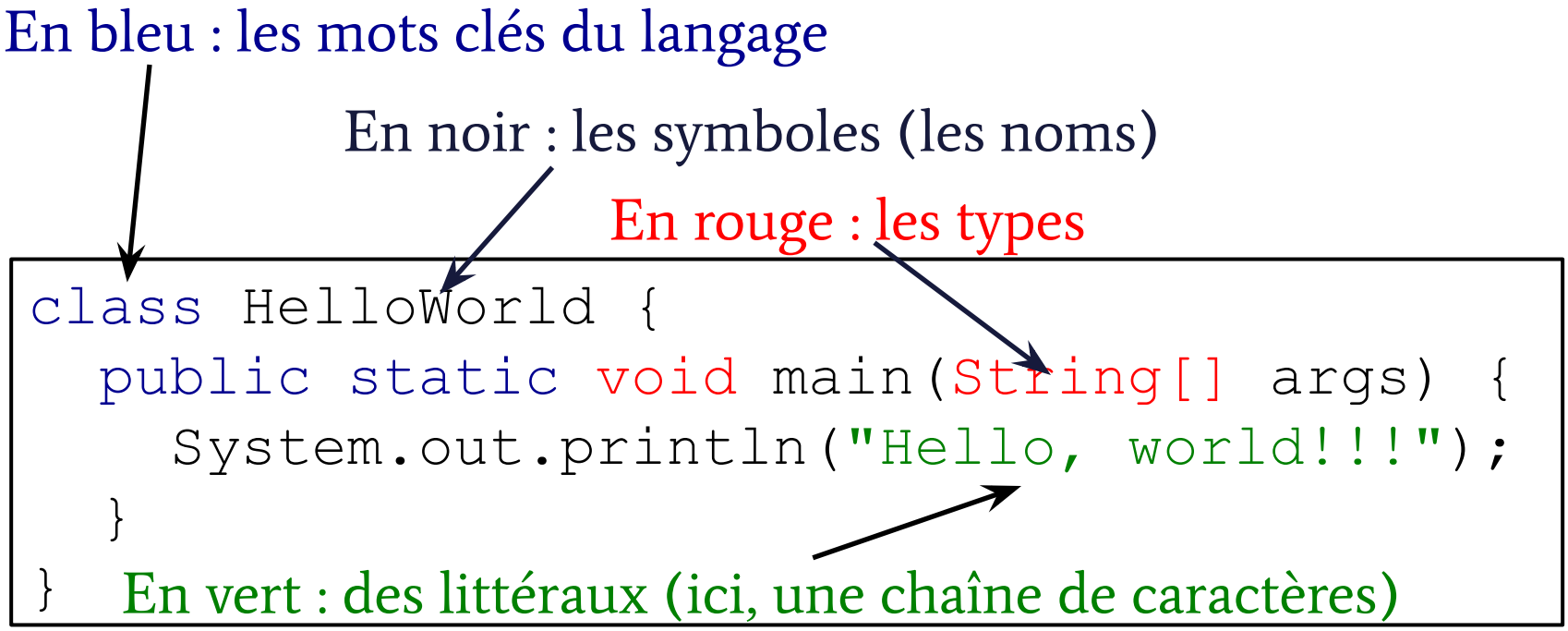
- Et des commentaires
  - Utilisés pour expliquer ce que fait le programme
  - Non exécutés, uniquement pour documenter le code

```
/*
 * ceci est un commentaire multi-lignes
 */

/* ceci est un commentaire multi-lignes sur une ligne */

// ceci est un commentaire mono-ligne
```

Mon premier programme Java



Mon premier programme Java

- Dans un premier temps, le mot clé `class` sert à définir le nom du programme
  - Ici, le nom du programme est `HelloWorld`
  - Le code du programme se trouve entre les accolades qui suivent

```
class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, world!!!");
  }
}
```

Mon premier programme Java

Interlude

L'équipe pédagogique tient à s'excuser pour un petit mensonge : le mot clé class est infiniment plus complexe que ce qui est présenté ici

Vous comprendrez le rôle exact du mot clé class dans les CI3 et CI4

Pour le moment, imaginez que class donne le nom du programme n'est pas totalement faux

Mon premier programme Java

- La ligne contenant `main` sera expliquée dans les CI3 à CI5
- Pour le moment
  - Cette ligne indique le début du programme…
  - …qui se trouve entre les accolades qui suivent

```
class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, world!!!");
  }
}
```

Mon premier programme Java

- Le code du programme est constitué de déclarations
  - `System.out.println` affiche son paramètre sur le terminal
    - ln ajoute une nouvelle ligne, `System.out.print` sinon
  - Le `"Hello, World!!!"` est le paramètre
  - Le point virgule (;) de fin de ligne indique la fin de la déclaration

```
class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, world!!!");
  }
}
```

Mon premier programme Java

- Les déclarations se terminent **toutes** par un point virgule !
  - Si on a une accolade fermante, alors pas de ;

```
class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, world!!!");
  }
}
```

Plan du cours

1. Mon premier programme Java
2. **Exécution d'un programme Java**
3. Variables et types de données
4. Les opérateurs du langage Java
5. Les conversions de type
6. Structures algorithmiques

Exécution d'un programme Java

- Étape 1 : traduction vers un fichier appelé bytecode
  - Le fichier source (contenant le programme Java) n'est pas exécutable en l'état, il doit être traduit
  - Le fichier bytecode contient une forme plus rapide à exécuter que le fichier source (validation syntaxique et sémantique effectuée)
  - La transformation source vers bytecode s'appelle la compilation
- Étape 2 : exécution dans une machine virtuelle Java
  - Un fichier de bytecode n'est pas directement exécutable par un processeur
  - Le programme `java` est capable d'interpréter un fichier bytecode

Exécution d'un programme Java

Fichier HelloWorld.java

```
class HelloWorld {… }
```

Compilation avec le programme javac

Fichier HelloWorld.class

```
0xcafebabe…
```

Interprétation avec le programme java

Mise en perspective

- En `Bash` (resp. `Python`)
  - Un programme source est un fichier `.sh` (resp. `.py`)
    (contient des déclarations `bash`, resp `python`)
  - Directement exécutable par l'interpréteur `bash` (resp. `python`)
- En `Java`
  - Un programme source est un fichier `.java`
    (contient des déclarations `Java`)
  - Doit être compilé en bytecode (fichier `.class`) avec `javac`
  - Le bytecode est exécutable par l'interpréteur `java`

Conception, compilation, exécution

```
$ emacs HelloWorld.java
```

```
class HelloWorld {
   …
}
```

Éditeur emacs
(fichier HelloWorld.java)

Première étape :
la conception

Conception, compilation, exécution

```
$ emacs HelloWorld.java
```

```
class HelloWorld {
   …
```

Attention : le nom du fichier
et le symbole qui suit class doivent coïncider

Convention majuscule en début
de symbole dans ce cas,
minuscule pour tous les autres symboles

Conception, compilation, exécution

```
$ emacs HelloWorld.java
$
```

Conception, compilation, exécution

```
$ emacs HelloWorld.java
$ ls
HelloWorld.java
$
```

Conception, compilation, exécution

```
$ emacs HelloWorld.java
$ ls
HelloWorld.java
$ javac HelloWorld.java
$
```

Deuxième étape :
la compilation

Conception, compilation, exécution

```
$ emacs HelloWorld.java
$ ls
HelloWorld.java
$ javac HelloWorld.java
$ ls
HelloWorld.class
HelloWorld.java
$
```

Conception, compilation, exécution

```
$ emacs HelloWorld.java
$ ls
HelloWorld.java
$ javac HelloWorld.java
$ ls
HelloWorld.class
HelloWorld.java
$ java HelloWorld
Hello, world!!!
```

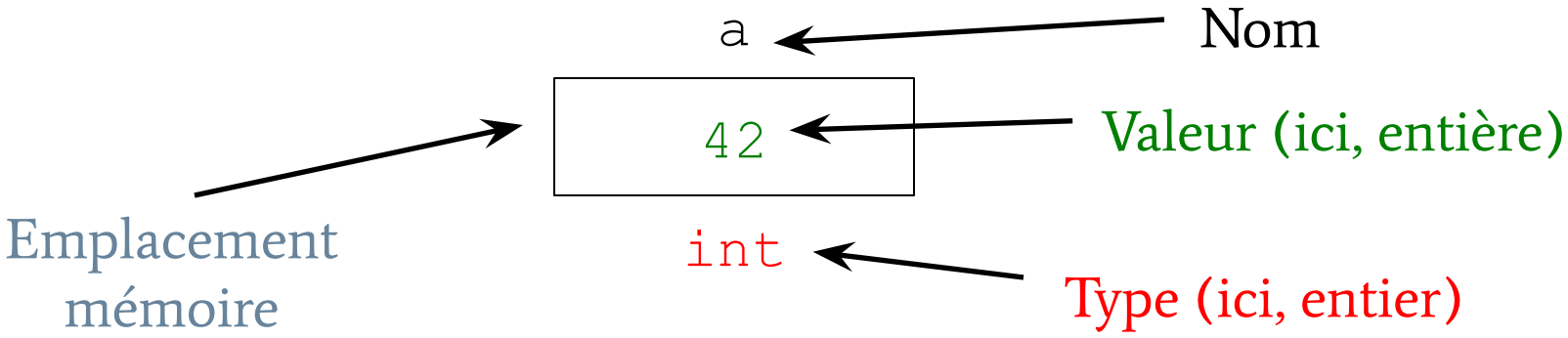
Troisième étape :
L'exécution

Plan du cours

1. Mon premier programme Java
2. Exécution d'un programme Java
3. **Variables et types de données**
4. Les opérateurs du langage Java
5. Les conversions de type
6. Structures algorithmiques

Les variables en Java (1/2)

- Une variable **est** un emplacement mémoire
  - Qui possède **un nom** (le nom de la variable)
  - **Un type** (la nature de ce que contient la variable)
  - Et **une valeur** (le contenu de la variable)



Les variables en Java (2/2)

- En Java, les types de base sont les :
  - Booléens : `boolean` (true ou false)
  - Entiers (signés) : `byte` (1 octet), `short` (2 octets), `int` (4 octets), `long` (8 octets)
  - Réels : `float` (4 octets), `double` (8 octets)
  - Caractères : `char` (un caractère **unicode**)
  - (Chaînes de caractères : `String` (plusieurs caractères), pas vraiment un type de base mais très utile)

Les variables en Java

- Définition avec :
  - `type symbole; /* ? valeur indéfinie ? */`
  - ou `type symbole = littéral;`

```
class HelloWorld {
  public static void main(String[] args) {
    String msg = "Coucou";/* Coucou */
    char c = '!';         /* ! */
    int x = 2;            /* 2 */
    int y = x + 5;        /* 7 (addition entière) */
    float z;              /* ? valeur indéfinie ? */
    boolean b = true;     /* true */
  }
}
```

Les littéraux en Java

- Un entier : une valeur entière sans symbole particulier
  - Si suivi de `l` (la lettre L minuscule), valeur de type `long` (ex: `2l`)
  - Sinon de type `int` (ex: `2`)
- Un réel : une valeur avec un point sans symbole particulier
  (ou avec exposant 3.14e7 == 3.14 * 10^7)
  - Si suivi de `f`, valeur de type `float` (ex: `3.14f`)
  - Sinon de type `double` (ex: `3.14`)
- Un caractère : un caractère entouré d'apostrophes (ex: `'a'`)
- Une chaîne de caractères : une suite de caractères entourée de guillemets (ex: "Ceci est une chaîne")

Plan du cours

1. Mon premier programme Java
2. Exécution d'un programme Java
3. Variables et types de données
4. **Les opérateurs du langage Java**
5. Les conversions de type
6. Structures algorithmiques

Les opérateurs du langage Java (1/2)

- Opérations arithmétiques sur les nombres (entiers ou flottants)
  - `+`, `-`, `*`, `/`, `%` (modulo), `++` (incrémente), `--` (décrémente)
- Opérations bit à bit (sur les entiers)
  - `&` (et), `|` (ou), `^` (xor), `~` (complément), `<<` (décalage à gauche), `>>` (décalage à droite)
- Opérations sur les booléens
  - `&&` (et), `||` (ou), `!` (non)
- Opérations sur les chaînes de caractères
  - `+` (concaténation)

Les opérateurs du langage Java (2/2)

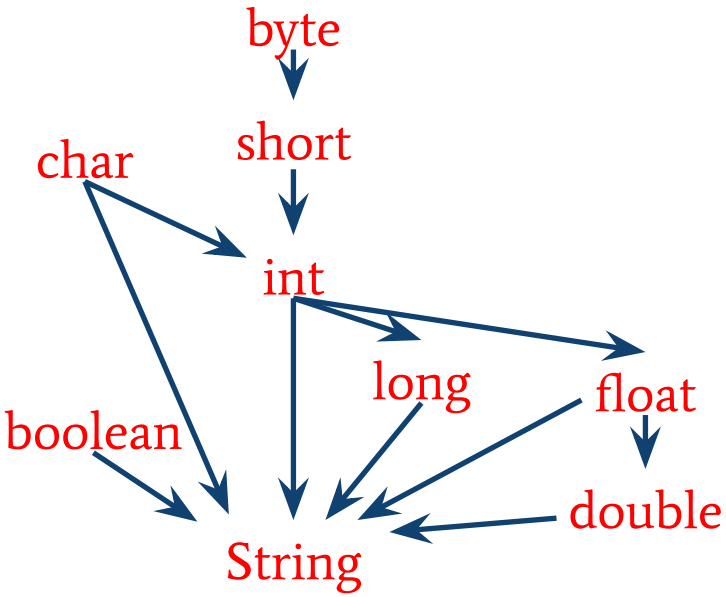
- Opérations de comparaison
  - `==`, `<=`, `<`, `>=`, `>`, `!=` (différent)
- Opération d'affectation (tous types)
  - `=`
- Combinaison d'affectation avec d'autres opérations possible `+=`, `/=`, `>>=` etc.
  - (Exemple : `a += 42` est équivalent à `a = a + 42`)

Plan du cours

1. Mon premier programme Java
2. Exécution d'un programme Java
3. Variables et types de données
4. Les opérateurs du langage Java
5. **Les conversions de type**
6. Structures algorithmiques

Conversion automatique de type

- Avant d'effectuer une opération, les valeurs peuvent être converties vers un type plus fort

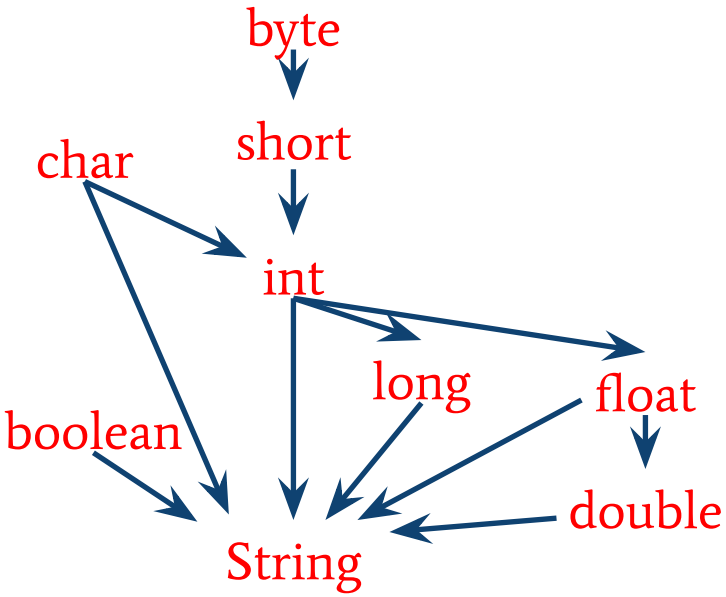


Type le plus faible

Type le plus fort

Conversion automatique de type

- Avant d'effectuer une opération, les valeurs peuvent être converties vers un type plus fort



Type le plus faible

Remarque : lors de la conversion d'un char vers un int c'est le numéro de caractère qui est renvoyé (par exemple 'a' a la valeur 97)

Type le plus fort

Conversion automatique de type - Unicode

Un ordinateur stocke des chiffres !

On doit donc utiliser une convention pour convertir les chiffres en lettres

Exemples:

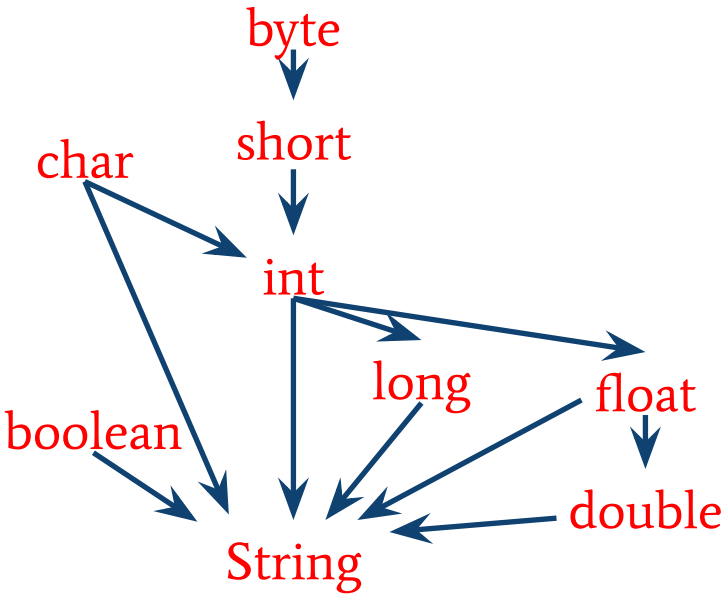
- ASCII: 7 bits, 128 caractères
- UTF-8: 8 bits, 256 caractères (+ extension unicode sur 4 bytes)
- UTF-16 (Java) : 16 bits, 65.536 caractères (+ extension unicode sur 4 bytes)
- Unicode : 4 bytes, 4 milliards de caractères

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Conversion automatique de type

- Avant d'effectuer une opération, les valeurs peuvent être converties vers un type plus fort




```
int x = 3;
Double y = x + 2.2; /* 5.2 */
String s1 = "Coucou" + x;
    /* Coucou3 */
char c = 'a';
String s2 = "Coucou" + c;
    /* Coucoua */
int d = c + 1;
    /* 98 car 'a' = 97 */
```

Conversion explicite de type

- Quand il n'existe pas de conversion implicite, on utilise la conversion explicite avec `(type)`
  - Dans ce cas, Java tronque le nombre

```
double x = 97.7;
int y = (int)x;              /* 97 */
char c = (char)y;            /* 'a' car c vaut 97 */
byte b = (byte)(y * 3); /* 291 modulo 256 = 35 */
```

Plan du cours

1. Mon premier programme Java
2. Exécution d'un programme Java
3. Variables et types de données
4. Les opérateurs du langage Java
5. Les conversions de type
6. **Structures algorithmiques**

Structures algorithmiques et blocs

- La structure d'un programme Java est donnée par des blocs
  - Commence par un { et termine par un }
  - Regroupe un ensemble de déclarations

```
class Test {
  public static void main(String[] args) {
    if(0 == 1) {
      System.out.println("Cette machine est bizarre");
      System.out.println("Elle pense que  0 == 1 !");
    }
  }
}
```

Bloc exécuté si 0 == 1

  - Accolades optionnelles si une unique déclaration dans un bloc
  - Pour être plus précis, un bloc est simplement une déclaration…
  - Même si ce n'est pas obligatoire, on indente les blocs pour la lisibilité (comme en Python) !

Schéma alternatif (1/2)

- Schéma alternatif simple
  - si alors ... sinon (si alors ... sinon ...)
  - Parties else if et else optionnelles

```
if(cond1) {
  body1
} else if(cond2) {
  body2
} else {
  body3
}
```

Exemple.java

```
class Exemple {
  static void main(String[] args) {
    int a = 21 * 2;
    if(a == 42) {
      System.out.println("Super !");
      System.out.println("Java, c'est facile !");
    }
  }
}
```

Schéma alternatif (2/2)

- Schéma alternatif complexe
  - Si val vaut v1, exécute body1
  - Sinon, si val vaut v2, exécute body2
  - Sinon, si ...
  - Sinon, exécute bodyn

```
switch(val) {
  case v1:
    body1;
    break;
  case v2:
    body2;
    break;
  …
  default:
    bodyd;
    break;
}
```

```
switch(c) {
  case 'a': System.out.println("Ceci est un a"); break;
  case 'b': System.out.println("Ceci est un b"); break;
  …
  default: System.out.println("Bizarre"); break;
}
```

Schéma alternatif (2/2)

- Schéma alternatif complexe
  - Si pas de `break`, continue l'exécution avec le `body` suivant
  - Attention au code spaghetti !

```
switch(val) {
  case v1:
    body1;
    break;
  case v2:
    body2;
    break;
  …
  default:
    bodyd;
    break;
}
```

```
switch(c) {
  case 'a': System.out.println("uniquement a");
  case 'b': System.out.println("a ou b"); break;
  case 'c': System.out.println("uniquement c");
  default:  System.out.println("ni a, ni b ☺");
}
```

Schémas itératifs

- Boucle `while`

  Tant que `cond` faire `body`

- Boucle `do ... while`

  Faire `body` tant que `cond` (body exécuté au moins une fois)

  Attention au `;`

- Boucle `for`

  Exécute `init`

  Tant que `cond` faire

  `body` puis `iter`

```
while(cond) {
  body
}
```

```
do {
  body
} while(cond);
```

```
for(init; cond; iter) {
  body
}
```

Schémas itératifs – exemples

```
int x = 0;
while(x < 10) {
   System.out.println(x);
   x++;
}
```

```
for(int x=0; x<10; x++) {
   System.out.println(x);
}
```

```
int x;
do {
   x = Math.random() % 10;
} while(x == 3);
```

Java versus Python versus Bash

- En Java
  - Le type d'une variable est explicite
    (`int x=100` ⇒ entier, `String name="Arya"` ⇒ chaîne de caractères)
  - Les blocs sont explicitement délimités par des accolades `{ … }`
- En Python
  - Le type d'une variable est implicite
    (`x=100` ⇒ entier, `name="Arya"` ⇒ chaîne de caractères)
  - Les blocs sont implicitement donnés par l'indentation
- En Bash
  - Toutes les variables sont de type chaîne de caractères
    (`PATH=/usr/bin`)
  - Les blocs d'instructions sont délimités par des mots clés
    (`do … done`, `if …; then … fi` etc…)

Notions clés

- Conception, compilation et exécution d'un programme Java
- Déclaration et typage des variables : `type var;`
  `boolean`, `byte`, `short`, `int`, `long`, `float`, `double`, `char`, `String`
- Opérateurs de base
- Structures algorithmiques
  - Schéma itératif (`if/else`)
  - Schéma alternatif (`while`, `for`, `do … while`)

If you want to know more

À propos des conversions de type en Java

Étrangeté

- Avant d'effectuer une opération arithmétique, Java convertit un `byte`, un `short` ou un `char` en `int`

```
byte x = 1;
byte y = 2;
byte z = x + y;
```

Interdit car :

Java convertit x et y en int avant d'effectuer l'opération

⇒ le résultat est un int et il est impossible d'affecter un int dans un byte

⇒ en additionnant deux bytes, on peut facilement dépasser la valeur maximale ou minimale

Encore plus d'étrangeté (pour votre culture)

- Si les deux membres de l'opération sont des littéraux entiers (`byte`, `short`, `int`) ou caractères (`char`), alors Java effectue **normalement** le calcul en `int`

- **Mais après le calcul**, s'il faut affecter le résultat à un des types entiers ou au type `char`, Java convertit le résultat si le nombre n'est pas trop grand

- **Raison**: Java peut deviner ou non si le résultat va dépasser le maximum/minimum au moment de la compilation et nous prévenir

Encore plus d'étrangeté (pour votre culture)

```
char c = 'a';
char d = c + 1;
```

Interdit car c n'est pas un littéral

(il est impossible de convertir de int vers char lors de l'affectation)

```
char c = 'a' + 1;
```

Autorisé car 'a' est un littéral

('a' devient le nombre 97, la somme vaut 98, qui représente le caractère 'b')