Le Langage Java

1ère année

M. Bastreghi J. Beleho P. Bettens M. Codutti
A. Hallal C. Leruste D. Nabet N. Pettiaux A. Rousseau

Haute École de Bruxelles — École Supérieure d'Informatique

Année académique 2011 / 2012

Leçon 2

Les conversions

- Présentation
- Dans les expressions
- Lors d'une assignation (et d'un return)
- Lors d'un appel de méthode
- Avec les chaines de caractères
- Boxing et unboxing
- Le casting
- Récapitulatif

Que sait-on déjà ?

Java est fortement typé ; les types doivent correspondre

- Ex : Lors d'une assignation, la valeur doit être du type de la variable

Parfois, le compilateur convertit pour nous

- Ex : **double** d = 1; (conversion de **int** vers **double**)

Mais souvent ce n'est pas le cas

- Ex : **int** i = 1.0; (refusé par le compilateur)

Quelles sont les **règles précises** ?

Contextes et sortes de conversions

Il y a **8 groupes** (sortes) de conversions.

Peuvent être utilisées dans **5 contextes** (éléments de programmes) différents

	8 sortes
5 contextes	en fonction du contexte seulement certaines sortes seront permises

Où en sommes-nous ?

Les conversions dans les expressions

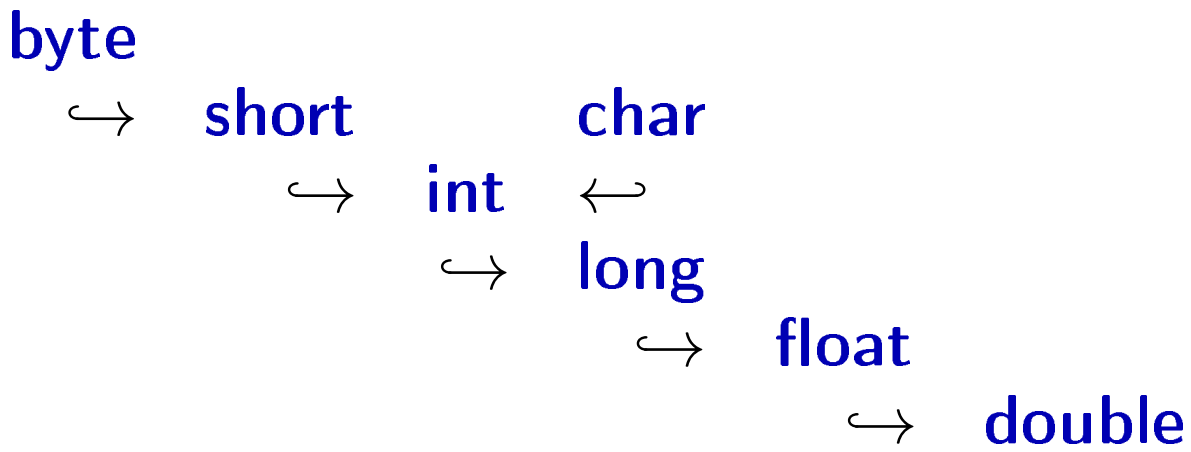
Il existe des conversions implicites dans les expressions

- Uniquement pour des expressions **numériques**
- Adapte le type des opérandes à ceux attendus par l'opérateur
- Conversion la plus fréquente : **élargissante de type primitif**

Les conversions dans les expressions

Conversion élargissante de type primitif (*widening primitive conversion*)

- Vers un type plus général



- entier vers réel : perte de précision possible

Les conversions dans les expressions

Cas des opérateurs **binaires**

- Opérateurs : $+$, $-$, $*$, $/$, $\%$, $<$, $>$, $<=$, $>=$, $==$, $!=$
- Si opérandes de types différents
  $\Longrightarrow$ on *élargit* le moins large
- **Exemples**

```
System.out. println ( 7 / 2. )
System.out. println ( 7. / 2. )
System.out. println ( 7. + 2 )
System.out. println ( 7.f + 2. )
System.out. println ( 1 <= 1.0 )
```

Les conversions dans les expressions

Cas des opérateurs **binaires** (suite)

- On élargit au minimum vers **int**
- Car les opérateurs n'existent pas en dessous
- **Exemples**

```
short s = 1;
char c = '3';
int i = 4;
... s + i ...
... s + 5L ...
... c - s ...
... 2 * i ...
... 2 <= i ...
```

Les conversions dans les expressions

Cas des opérateurs unaires

- Opérateurs : $+$, $-$
- Ainsi que l'indice d'un tableau
- **byte**, **short**, **char** $\longrightarrow$ **int**
- Exemples

```
short s = 1;
byte b = 2;
... +s ...
... -b ...
... -'A' ...
... t[s] ...
... t['1'] ...
```

Les conversions dans les expressions

Remarque sur ++

- Existe pour tous les types numériques
- **Exemple**

```
char c = 'a';
c = c+1; // assignation impossible
```

ne se comporte pas comme

```
char c = 'a';
c++;   //  c vaut 'b'
```

Où en sommes-nous ?

Conversion lors d'une assignation

Que se passe-t'il lors d'une **assignation** ?

- Adapte le type de l'expression au type de la variable
- Opérateurs : $=$, $+=$, $-=$, $*=$, $/=$, $\%=$
- Permet la **conversion élargissante** déjà vue
- Mais aussi la **conversion arrondissante**

Conversion lors d'une assignation

Conversion arrondissante de type primitif
(*narrowing primitive conversion*)

double
↪ **float**
↪ **long**
↪ **int**
↪ **short** ⟷ **char**
↪ **byte** ↙ ↗

- À chaque étape : **perte de précision** possible

Conversion lors d'une assignation

Conversion arrondissante **si et seulement si**

- La variable est de type **byte**, **short**, ou **char**
- L'expression est
  - constante
  - de type **byte**, **short**, **char** ou **int**
  - sa valeur est représentable dans le type de la variable

Si ce n'est pas le cas, erreur à la compilation

Conversion lors d'une assignation

Exercice : identifiez les instructions correctes

```
long l1 = 12;
long l2 = 'a'+1;
short s1 = 12;
short s2 = 1+2;
byte b1 = 123245;
byte b2 = s1+1;
byte b3 = 21L;
```

Conversion de la valeur de retour

Situation similaire pour la **valeur de retour** d'une méthode

- Le type de l'expression accompagnant l'instruction **return** doit pouvoir être ramené au *ResultType*
- Assimilable à une assignation $\Longrightarrow$ mêmes règles
- **Exemple**

```
public static long add( int opérandeGauche, int opérandeDroite) {
    return opérandeGauche + opérandeDroite;
}
```

Où en sommes-nous ?

- Présentation
- Dans les expressions
- Lors d'une assignation (et d'un return)
- Lors d'un appel de méthode
- Avec les chaines de caractères
- Boxing et unboxing
- Le casting
- Récapitulatif

Conversion lors d'un appel de méthode

Conversion des **paramètres** effectifs

- **Conversion élargissante** possible
- **Exemple** : avec la méthode suivante

```
public static int add( int opérandeGauche, int opérandeDroite) {
    return opérandeGauche + opérandeDroite;
}
```

les arguments lors de l'appel peuvent être **byte**, **short**, **char** ou encore **int**

Conversion lors d'un appel de méthode

Difficulté liée à la **surcharge**

- S'il existe plusieurs méthodes avec le même nom, il peut y avoir plusieurs candidates pour un appel
- **Exemple**

```
public static int f(long op1, long op2) {...}
public static int f(int op1, int op2) {...}
```

Quelle méthode est choisie avec cet appel ?

```
short s1=1, s2=2:
f(s1,s2);
```

Conversion lors d'un appel de méthode

Exemple

```
public static double op( double opérandeGauche, double opérandeDroite) {
    return opérandeGauche * opérandeDroite;
}

public static int op( int opérandeGauche, int opérandeDroite) {
    return opérandeGauche / opérandeDroite;
}
```

- Que retourne op(3.,2.) ?
- Que retourne op(3,2) ?
- Que retourne op(3.,2) ?

Conversion lors d'un appel de méthode

Cas particulier

- Parfois, il n'y a pas de méthode plus spécifique.
- **Exemple**

```
public static int f( int op1, long op2 ) {...}
public static int f( long op1, int op2 ) {...}
```

Accepté mais certains appels seront ambigus (erreur à la compilation)

- f(1,2L) // 1ère méthode
- f(1L,2) // 2ème méthode
- f(1,2) // ambigu

Où en sommes-nous ?

- Présentation
- Dans les expressions
- Lors d'une assignation (et d'un return)
- Lors d'un appel de méthode
- Avec les chaines de caractères
- Boxing et unboxing
- Le casting
- Récapitulatif

Conversion en chaines de caractères

Opérateur + avec un opérande de type String
⇒ l'autre opérande converti en String

- **Exemples**

```
System.out. println ("1+1␣=␣"+2);
String  s  = "Pi␣=␣" + 3.1415;
```

- Que donnera ceci ?

```
System.out. println ("1"+2+3);
System.out. println (1+2+"3");
```

Où en sommes-nous ?

Les wrappers

À chaque type primitif correspond une classe **englobante** (**wrapper**)

boolean	:	Boolean	**int**	:	Integer
byte	:	Byte	**long**	:	Long
char	:	Character	**float**	:	Float
short	:	Short	**double**	:	Double

- Objet avec la même valeur que la valeur primitive
- Utile là où un objet est requis
- **Exemple**

```
List<Integer> list = new ArrayList<Integer>();
```

Boxing / unboxing

Conversions automatiques

- Du primitif vers le wrapper (**boxing**)
- Du wrapper vers le primitif (**unboxing**)
- Ajoutées par le compilateur
  - Dans les expressions
  - Dans les assignations
  - Avec les paramètres effectifs
- **Exemple** : repérez les (un)boxing implicites

```
List<Integer> list = new ArrayList<Integer>();
list.add(1);
System.out.println( list.get(0)+1);
list.set(0, list.get(0)+1 );
int i = list.get(0);
```

Où en sommes-nous ?

- Présentation
- Dans les expressions
- Lors d'une assignation (et d'un return)
- Lors d'un appel de méthode
- Avec les chaines de caractères
- Boxing et unboxing
- Le casting
- Récapitulatif

Le *casting*

On peut imposer une conversion que le compilateur ne ferait pas.

- C'est la **conversion explicite**
- Valeur convertie dans le type explicité par le casting

Casting :
(*Type*) *Expression*

- Grammaticalement, c'est assimilé à un opérateur

Tableau des priorités et associativités

priorité			associativité
forte	post unaires	(params), ., expr++, expr−−	⟹
	pré unaires	(Type), ++expr, −−expr,	
		−, +, !, **new**	⟸
	multiplicatif	∗, /, %	⟹
	additif	−, +	⟹
	relationnels	<, >, <=, >=	⟹
	égalité	==, !=	⟹
	et	&&	⟹
	ou	||	⟹
	condition	:?	⟸
faible	assignations	=, +=, −=, ∗=, /=, %=	⟸

Le *casting*

Quelles sont les conversions permises ?

- **élargissante**
- **arrondissante**
- **(un)boxing**
- **identique**
  - conversion dans le type que possède déjà l'expression
  - par facilité (si on n'est pas sûr)
  - permise aussi dans les autres contextes car simplifie les règles

Le *casting*

Remarques sur les conversions **arrondisantes**

- aucune contrainte
- si entier vers entier
  - on prend les octets de poids faible
  - $\Longrightarrow$ valeur convertie peut être fort différente
- **Exemple** : **byte** b = (**byte**) 256;

$$256 = \boxed{0}\boxed{0}\boxed{0\ldots0\ 1}\boxed{0} \Longrightarrow b = \boxed{0} = 0$$

Exemples de *casting*

```
int entier = (int) 5;
int entier = (int) 5l;
int entier = (int) 12000000000000000000000l;
    // erreur à la compilation ...
int entier = (int) 1200000000000L; // Accepté mais ...
double réel = (double) 12;
String mot = (String) 12; // erreur à la compilation
String mot = (String) "mot";
boolean b = (boolean) 1; // erreur à la compilation
int entier = (byte) 500-400;
byte entier = (byte) 500-400; // erreur à la compilation
byte entier = (byte) (500-400);
int entier = (byte) (190-(byte)100);
```

Récapitulatif

Il y a **8 sortes** de conversions

- Élargissante / arrondissante de type primitif
- Conversion en chaine de caractères
- Boxing / Unboxing
- Conversion identique
- Élargissante / arrondissante de type référence
  On n'insiste pas sur celles-ci mais on peut en retrouver une illustration dans les méthodes equals

Récapitulatif

Il y a **5 contextes** de conversion

- La promotion (calcul) numérique
- L'assignation
- Le casting
- La chaine de caractères
- L'appel de méthode

Récapitulatif

	élargis.	arrondi	chaine	boxing	unboxing	ident.
promotion num.	√				√	√
assignation	√	√ (*)		√	√	√
chaine			√			√
casting	√	√		√	√	√
méthode	√			√	√	√

(*) : sous certaines conditions

Crédits

Ce document a été produit avec les outils suivants

- La distribution `Ubuntu` du système d'exploitation `Linux`
- `LaTeX` comme système d'édition
- La classe `Beamer` pour les transparents
- Les packages `listings`, `fancyvrb`, ...
- Les outils `make`, `rubber`, `pdfnup`, ...