Le Langage Java

1re année

J. Beleho (bej) C. Leruste (clr) M. Codutti (mcd)
P. Bettens (pbt) F. Servais (srv) C. Leignel (clg)
D. Nabet (dna) J. Lechien (jlc)

Haute École de Bruxelles — École Supérieure d'Informatique

Année académique 2014 / 2015

Séance 11

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
- Le casting
- Récapitulatif

Conversions

Crédit photo

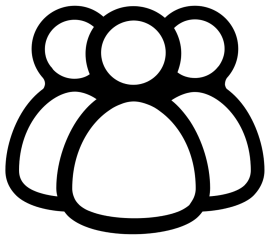
Conversions

Java impose que les types correspondent.
Parfois le compilateur convertit implicitement.

Quels sont les règles précises ?

- ✔ **double** d = 1;
- ✘ **int** i = 1.0;

Contextes et sortes de conversions



8 groupes

5 contextes

Où en sommes-nous ?

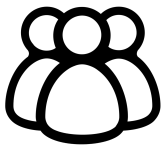
- Présentation
- Dans les expressions
- Lors d'une assignation (et d'un return)
- Lors d'un appel de méthode
- Avec les chaines de caractères
- Le casting
- Récapitulatif

Les conversions dans les expressions

Expressions

- Expressions **numériques**
- Adapte le type des opérandes
- Conversion **élargissante de type primitif**

Les conversions dans les expressions



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

byte
↪ **short** **char**
↪ **int** ↩
↪ **long**
↪ **float**
↪ **double**

Les conversions dans les expressions

- Opérateurs binaires, opérateurs unaires
- Le moins large et au minimum **int**

Les conversions dans les expressions

Exemples

```
System.out.println( 7 / 2. )
System.out.println( 7. / 2. )
System.out.println( 1 <= 1.0 )
byte b = 2; short s = 1; char c = '3'; int i = 4;
... s + i ...
... s + 5L ...
... c - s ...
... 2 <= i ...
... +s ...
... -b ...
... -'A' ...
... t[s] ...
... t['1'] ...
```

Où en sommes-nous ?

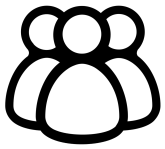
- Présentation
- Dans les expressions
- Lors d'une assignation (et d'un return)
- Lors d'un appel de méthode
- Avec les chaines de caractères
- Le casting
- Récapitulatif

Conversion lors d'une assignation

Assignation

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

Dans le cadre d'une assignation

- Elles ne sont pas toutes permises
- **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;
```

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

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*

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
- Le casting
- Récapitulatif

Conversion lors d'un appel de méthode

Appel de méthode

- Conversion des **paramètres** effectifs
- **Conversion élargissante** possible

Exemple

```
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

Attention à la **surcharge** (*overloading*)

Exemple

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

Quelle méthode est choisie avec cet appel ?

```
short s1=1, s2=2;
max(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 max( int op1, long op2 ) {...}
public static int max( long op1, int op2 ) {...}
```

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

- max(1,2L) // 1ère méthode
- max(1L,2) // 2ème méthode
- max(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
- Le casting
- Récapitulatif

Conversion en chaines de caractères

Chaine

Opérateur + avec un opérande de type 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 ?

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

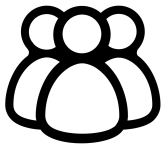
Le *casting*

Casting

Conversion **explicite**

Casting :
(*Type*) *Expression*

Le *casting*



- élargissante
- arrondissante
- (un)boxing
- identique

Le *casting*

Remarques sur les conversions **arrondissantes**

- Aucune contrainte, on tronque

Exemple

byte b = (**byte**) 256;

- 256 = | 0 | 0 | 1 | 0 | (**int** codé sur 4 bytes)
- $\Longrightarrow$ b = | 0 | = 0

Le *casting*

Ne **peut pas** être utilisé pour

- Convertir en chaine
  - `(String) monEntier` **interdit**
  - Écrire `""+monEntier` à la place
- Convertir une chaine qui contient un numérique
  - `(int) "12"` **interdit**
  - Écrire `Integer.parseInt("12")` à la place

Exemples de *casting*

Exemples

```
int entier = (int) 5;
int entier = (int) 5l;
int entier = (int) 120000000000000000000l;
    // erreur à la compilation ...
int entier = (int) 1200000000000L; // Accepté mais ...
double réel = (double) 12;
String mot = (String) 12; // erreur à la compilation
int entier = (int) "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);
```

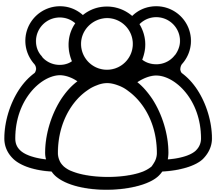
Rappel des règles de priorité

Crédit photo

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	=, +=, −=, ∗=, /=, %=	⟸

Récapitulatif



Il y a **8 sortes** de conversions

- Élargissante / arrondissante de type primitif
- Conversion en chaine de caractères
- Conversion identique
- Boxing / Unboxing
- Élargissante / arrondissante de type référence

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	ident.
promotion num.	√			√
assignation	√	√ (*)		√
chaine			√	√
casting	√	√		√
méthode	√			√

(*) : sous certaines conditions

Crédits

Ces slides sont le support pour la présentation orale de l'unité d'enseignement **DEV1-JAV** à HEB-ÉSI

Crédits

Les distributions Ubuntu et/ou debian
du système d'exploitation GNU Linux.
LaTeX/Beamer comme système d'édition.
Git et GitHub pour la gestion des versions et le suivi.
GNU make, rubber, pdfnup, ... pour les petites tâches.

Images et icônes

deviantart, flickr, The Noun Project