

R1.01

INITIATION AU DÉVELOPPEMENT

Cours 4, partie 1 : – Valeur et Référence – Classes Enveloppe

(résumé dans Mémento x)

Hervé Blanchon & Anne Lejeune

Université Grenoble Alpes

IUT 2 – Département Informatique

Sommaire

 Notion de valeur et de référence en Java

 Classes enveloppes

 Exemple récapitulatif

VALEUR ET RÉFÉRENCE EN JAVA

Que contient une variable ?

 Le contenu d'une variable est différent selon que ...

 ... c'est une variable de **type primitif**

 ... c'est une variable de **type Classe** (un objet)

 Il est important de faire la différence...

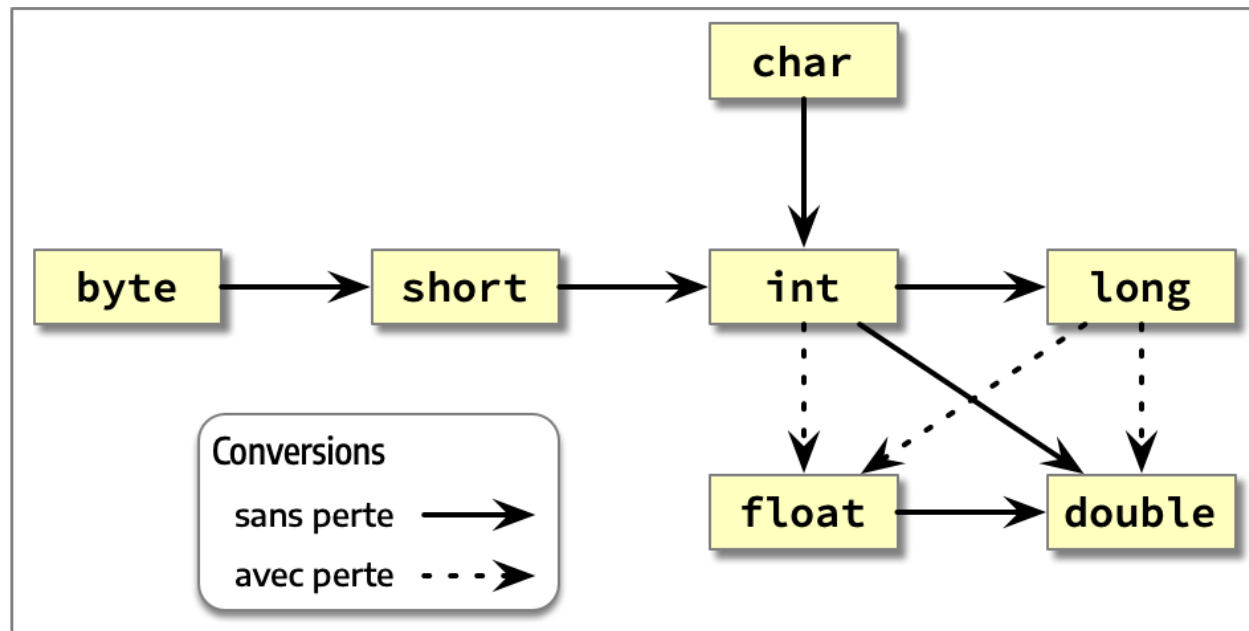
Les types primitifs (rappel)

ATTENTION :
ce ne sont
pas des
classes



| Type primitif | Signification | Intervalle de valeurs / valeurs | Place occupée en mémoire |
|---------------|------------------------------|--------------------------------------|--------------------------|
| byte | entier très court | [-128 ; 127] | 1 octet / 8 bits |
| short | entier court | [-32 768 ; 32 767] | 2 octets / 16 bits |
| int | entier | [-2 147 483 648 ; 2 147 483 647] | 4 octets / 32 bits |
| long | entier long | $[-2^{63} ; 2^{63} - 1]$ | 8 octets / 64 bits |
| float | nombre réel | $[-1.4 * 10^{-45} ; 3.4 * 10^{38}]$ | 4 octets / 32 bits |
| double | nombre réel double précision | $[4.9 * 10^{-324} ; 1.7 * 10^{308}]$ | 8 octets / 64 bits |
| char | caractère unicode | 65 536 caractères possibles | 2 octets / 16 bits |
| boolean | booléenne | true ou false | 1 octets / 8 bits |

Conversions de types primitifs

Conversions autorisées (automatiques)



Conversions forcées (avec perte d'information)

-  opérateur cast : `(nouveau_type)`
-  syntaxe : `(nouveau_type) valeur_variable_ou_expression`

Conversion de types primitifs

Exemple

Code Java

```
float monFloat = 12.57f;  
// dans le code Java, les flottants se notent avec un f  
// sinon, Java considère que ce sont des doubles  
// 12.57f -> de type float  
// 12.57 -> de type double  
  
int monInt = (int) monFloat;  
// conversion forcée d'un flottant en entier  
// seule la partie entière est conservée (pas d'arrondi)  
  
System.out.println("monFloat = " + monFloat + ", monInt = " + monInt);
```

Trace

```
monFloat = 12.57, monInt = 12  
// un flottant et un entier
```

Contenu d'une variable...

... de Type primitif

 déclaration-initialisation

```
int i = 12;
```

 état de la mémoire

| variable | adresse | contenu mémoire |
|----------|---------|-----------------|
|----------|---------|-----------------|

| | | |
|---|------------|----|
| i | \$ff34ef24 | 12 |
|---|------------|----|

 À RETENIR !

 **une variable de type primitif
contient une valeur**

... de Type Classe d'Objet

 déclaration-initialisation

```
String s = new String("ABC");
```


 état de la mémoire

| variable | adresse | contenu mémoire |
|----------|---------|-----------------|
|----------|---------|-----------------|

| | | |
|---|------------|------------|
| s | \$ff34ef68 | \$ff34effa |
|---|------------|------------|

| | | |
|--|------------|-------|
| | \$ff34effa | "ABC" |
|--|------------|-------|

 À RETENIR !

 **une variable de type Classe
contient une référence** à un objet
(un pointeur vers un objet,
l'adresse d'un objet)

Contenu d'une variable de type classe

Représentation en mémoire

 déclaration-initialisation

```
String s = new String("ABC");
```


 état de la mémoire

| variable | adresse | contenu mémoire |
|----------|---------|-----------------|
|----------|---------|-----------------|

| | | |
|---|------------|------------|
| s | \$ff34ef68 | \$ff34effa |
|---|------------|------------|

| | | |
|--|------------|-------|
| | \$ff34effa | "ABC" |
|--|------------|-------|

 À RETENIR !

 **une variable de type Classe**
contient une référence à un objet
(un pointeur vers un objet,
l'adresse d'un objet)

Représentation graphique

 déclaration-initialisation

```
String s = new String("ABC");
```


 état de la mémoire

| variable | adresse | contenu mémoire |
|----------|---------|-----------------|
|----------|---------|-----------------|

| | | |
|---|------------|--|
| s | \$ff34ef68 | |
|---|------------|--|

| | | |
|--|------------|-------|
| | \$ff34effa | "ABC" |
|--|------------|-------|

 **Lecture de la flèche**


 la variable s contient un pointeur
sur l'objet qui se trouve à
l'adresse \$ff34effa

CLASSES ENVELOPPES

WRAPPERS

Pourquoi ? À quoi ça sert ?

 Quand on a besoin d'un **Objet « de type primitif »**

 On utilise la classe enveloppe (wrapper) qui lui correspond (elle encapsule les données du type primitif)

| Type primitif | Classe enveloppe | |
|---------------|----------------------------|--|
| boolean | B oolean | |
| char | C har a cter | |
| byte | B yte | sous-classes de la classe Number |
| short | S hort | |
| int | I nteger | |
| long | L ong | |
| float | F loat | |
| double | D ouble | |

 On utilise une instance de classe enveloppe comme une instance d'objet ordinaire

La documentation


Boolean

 <http://docs.oracle.com/javase/7/docs/api/java/lang/Boolean.html>

Character

 <http://docs.oracle.com/javase/7/docs/api/java/lang/Character.html>

ClasseY

 <http://docs.oracle.com/javase/7/docs/api/java/lang/ClasseY.html>

Autoboxing

 conversion automatique

 type primitif \rightarrow classe enveloppe correspondante

 Exemples

1. `Character c = 'a';` // `char` \rightarrow `Character`

2. `Integer i = 12;` // `int` \rightarrow `Integer`

Unboxing

Conversion automatique


 classe enveloppe \rightarrow type primitif

Exemples

1. `Integer iInteger = new Integer(-8);`

2. `int iInt = iInteger; // Integer \rightarrow int`

Notion de classe immuable

 Une classe est dite **immuable** si l'état (valeur des attributs) d'un objet qui l'instancie ne peut pas être modifié après sa création

 Les classes enveloppes et la classe `String` sont **immuables**

 Exemple (tiré de http://fr.wikipedia.org/wiki/Objet_immuable)

```
String s = "ABC";  
s.toLowerCase();  
  
s = s.toLowerCase();
```

// ne modifie pas s mais produit un
// nouvel objet de la classe String
// s référence un nouvel objet qui
// contient "abc"
// l'objet qui contient "ABC" n'est
// plus accessible !

Note

Rien dans la déclaration d'un objet de classe `String` ne le contraint à être immuable : c'est plutôt qu'aucune des méthodes associées à la classe `String` n'affecte jamais la valeur d'un tel objet, ce qui le rend de fait immuable

EXEMPLE RÉCAPITULATIF

Code exemple

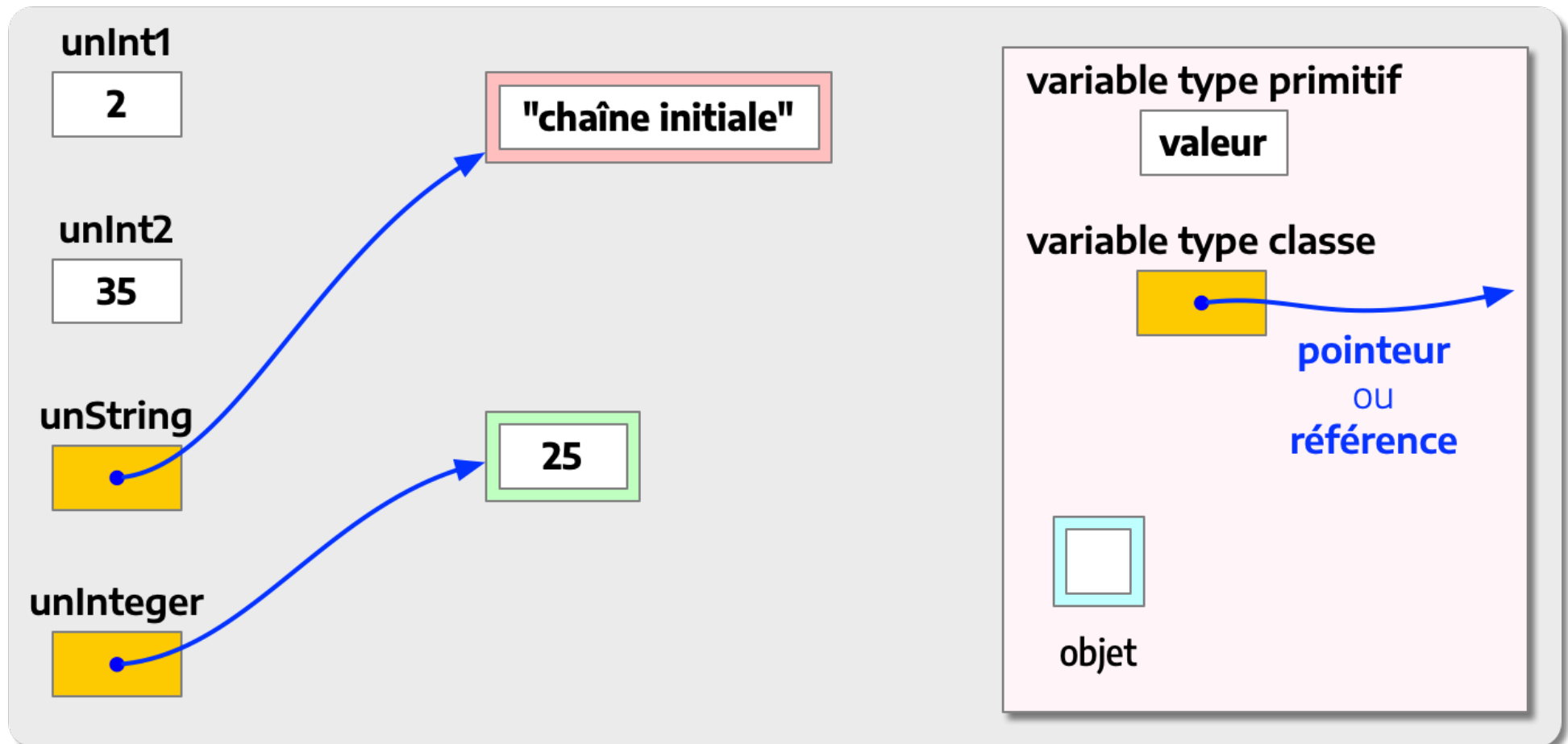
```
1 public class PrimitifsEnveloppesString {
2
3     public static void main(String[] args) {
4         int unInt1 = 2;
5         int unInt2 = 35;
6         Integer unInteger = 25;           // autoboxing int -> Integer
7         String unString = "chaîne initiale";
8
9         // autour de la variable int1 de type primitif
10        System.out.println("unInt1 tel qu'initialisé : " + unInt1);
11        unInt1 = unInteger;                // unboxing Integer -> int
12        System.out.println("unInt1 après changement de valeur : " + unInt1);
13
14        // autour de la variable unInteger de type Integer (enveloppe)
15        System.out.println("unInteger tel qu'initialisé : " + unInteger);
16        unInteger = unInt2;                // autoboxing int -> Integer
17        System.out.println("unInteger après changement de valeur : " + unInteger);
18
19        // autour de la variable unString de type String
20        System.out.println("unString tel qu'initialisé : " + unString);
21        unString = unString.toUpperCase(); // toUpperCase est une fonction
22        System.out.println("unString après passage en majuscules : " + unString);
23    }
24 }
```

Trace

| | |
|----|--|
| 10 | unInt1 tel qu'initialisé : 2 |
| 12 | unInt1 après changement de valeur : 25 |
| 15 | unInteger tel qu'initialisé : 25 |
| 17 | unInteger après changement de valeur : 35 |
| 20 | unString tel qu'initialisé : chaîne initiale |
| 22 | unString après passage en majuscules : CHÂÎNE INITIALE |

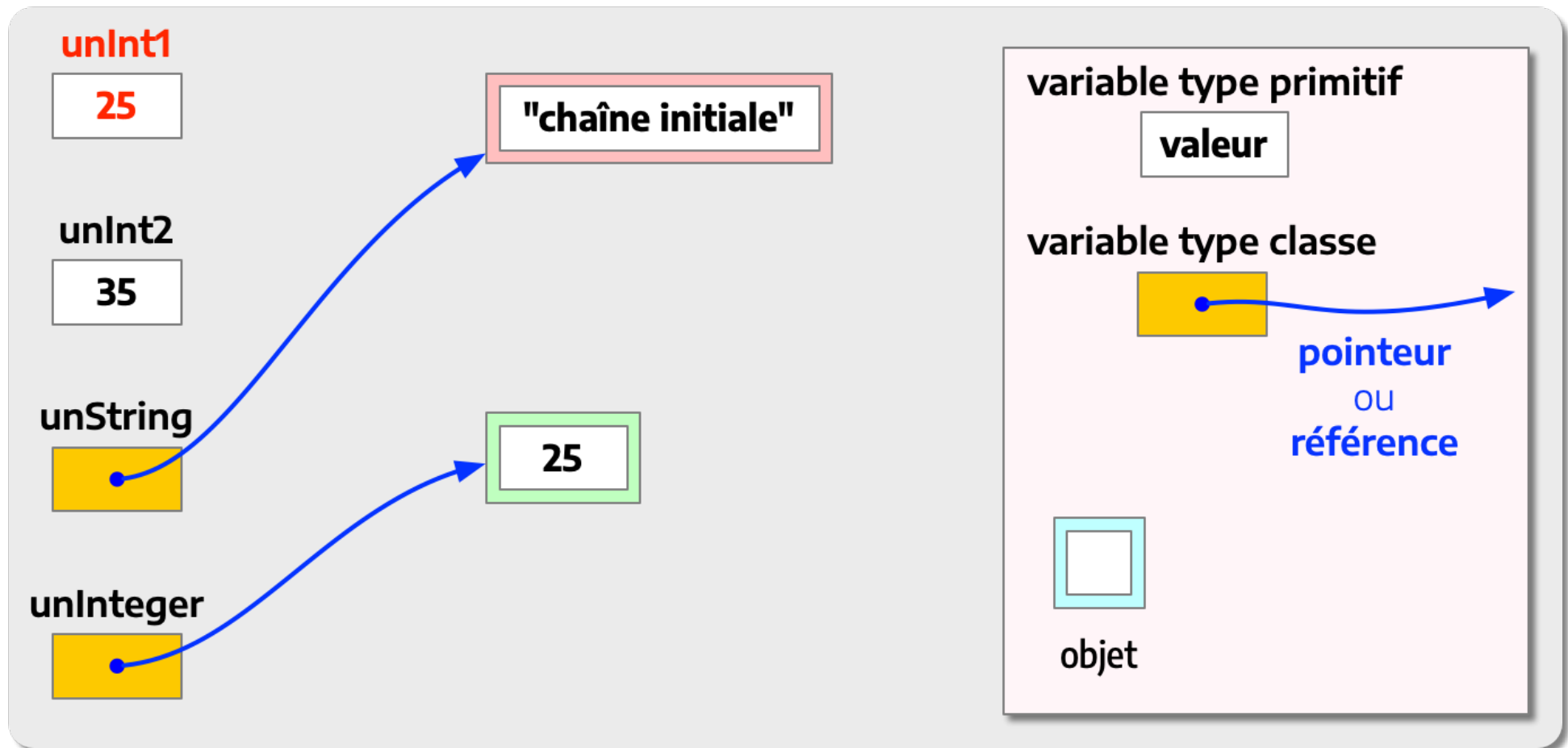
État de la mémoire (PrimitifsEnveloppesString)

```
4 int unInt1 = 2;
5 int unInt2 = 35;
6 Integer unInteger = 25; // autoboxing int -> Integer
7 String unString = "chaîne initiale";
```



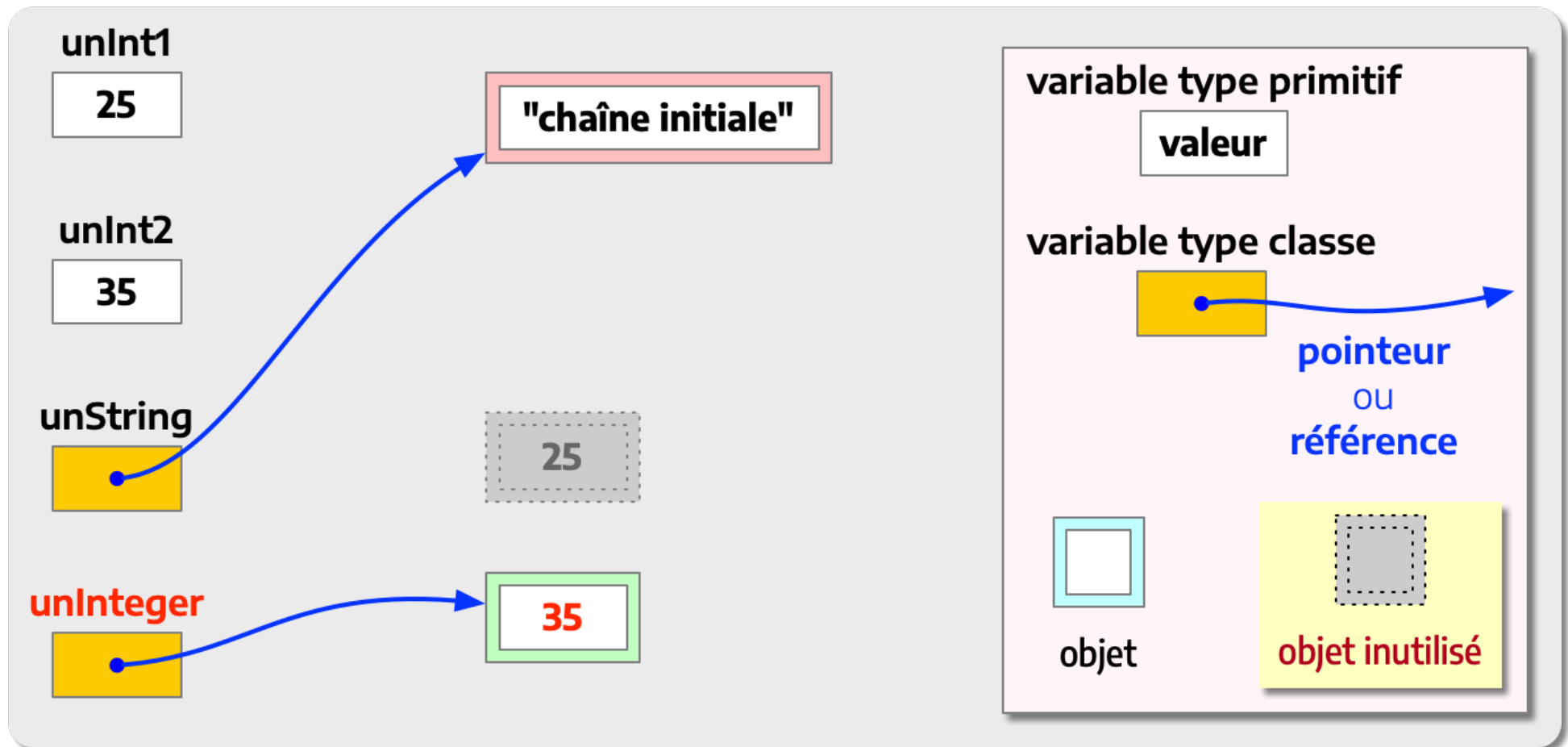
État de la mémoire (PrimitifsEnveloppesString)

```
9 // autour de la variable int1 de type primitif
10 System.out.println("unInt1 tel qu'initialisé : " + unInt1);
11 unInt1 = unInteger; // unboxing Integer -> int
12 System.out.println("unInt1 après changement de valeur : " + unInt1);
```



État de la mémoire (PrimitifsEnveloppesString)

```
14 // autour de la variable unInteger de type Integer (enveloppe)
15 System.out.println("unInteger tel qu'initialisé : " + unInteger);
16 unInteger = unInt2; // autoboxing int -> integer
17 System.out.println("unInteger après changement de valeur : " + unInteger);
```



État de la mémoire (PrimitifsEnveloppesString)

```
19 // autour de la variable unString de type String
20 System.out.println("unString tel qu'initialisé : " + unString);
21 unString = unString.toUpperCase();
22 System.out.println("unString après passage en majuscules : " + unString);
```

