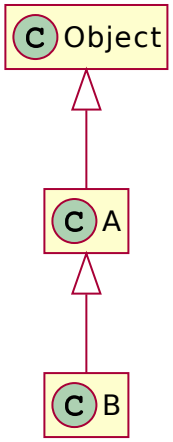
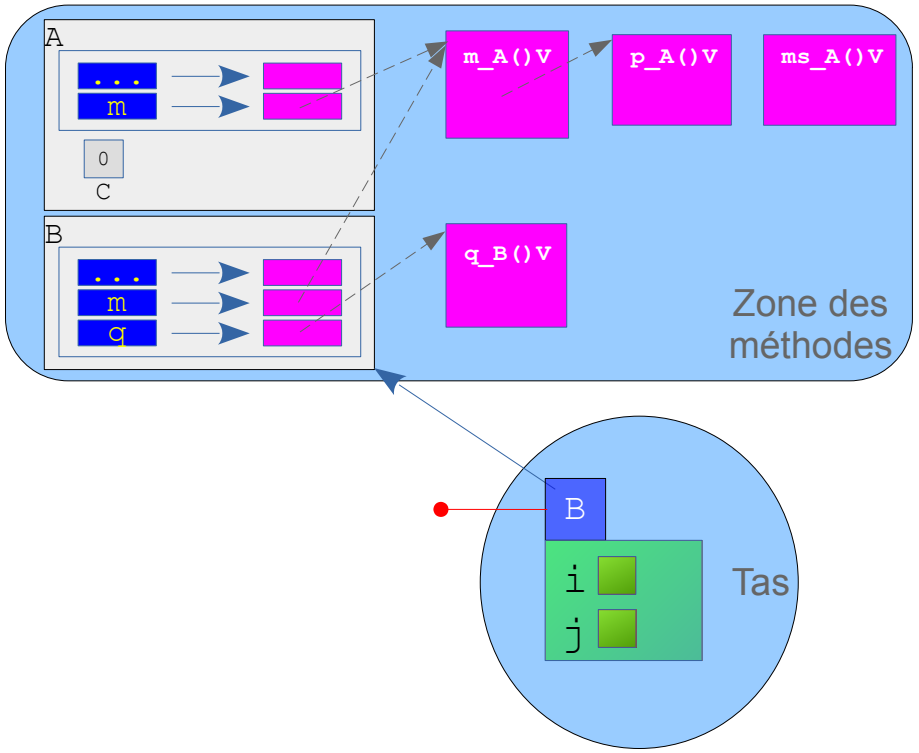
Rappels et compléments

La relation d'héritage est une relation entre classes.
Elle n'intervient pas au niveau des objets.

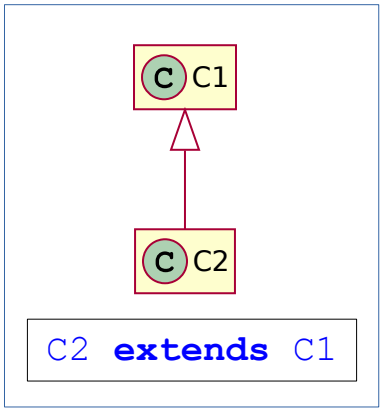
```
class A {
    public static final int C = 0;
    private int i;
    public void m() { p(); }
    private void p() { ... }
    public static void ms() { ... }
}
```

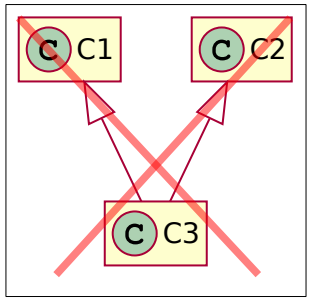
```
class B extends A {
    private int j;
    public void q() { ... }
}
```

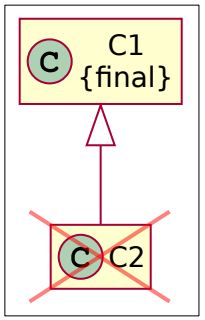


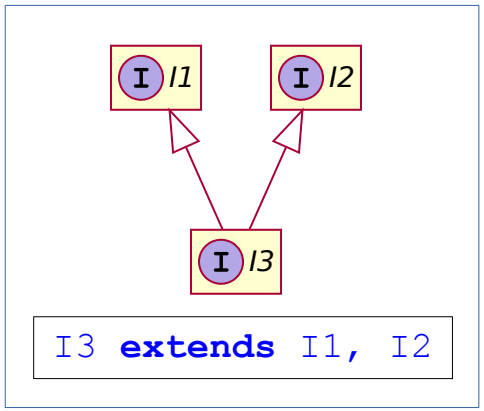


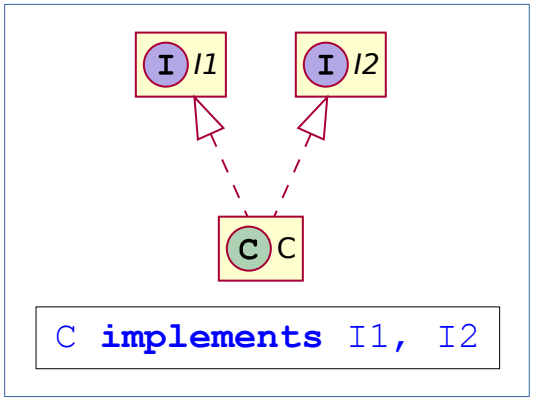
Rappels et compléments

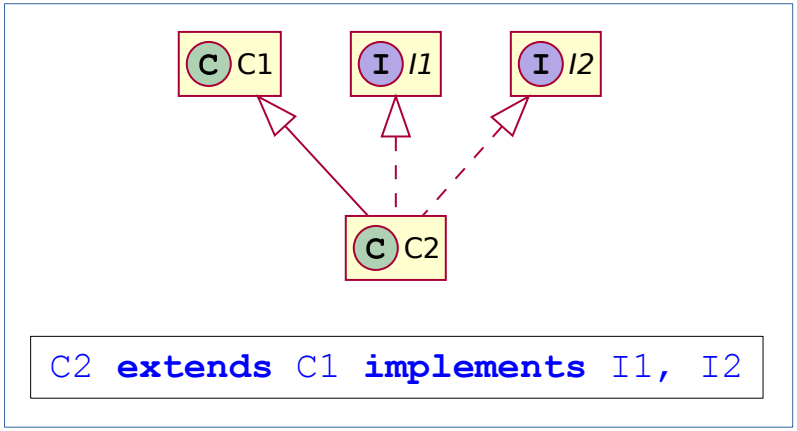


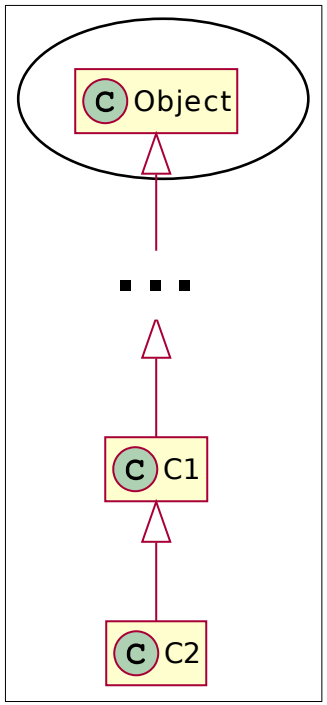










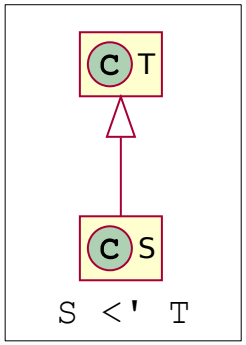


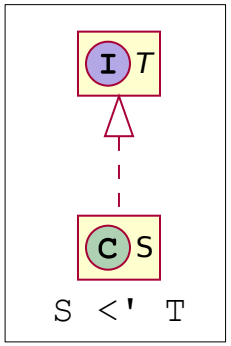
Rappels et compléments

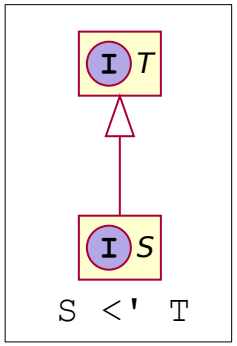
« *S est compatible avec T pour l'affectation* » signifie :
`v = e;` est possible quand `v` est de type T et `e` de type S.

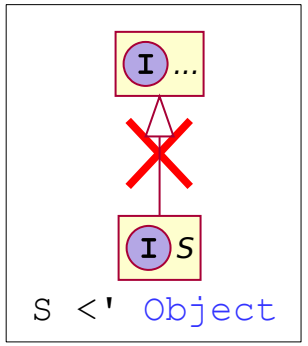
En Java, la compatibilité se confond avec le sous-typage

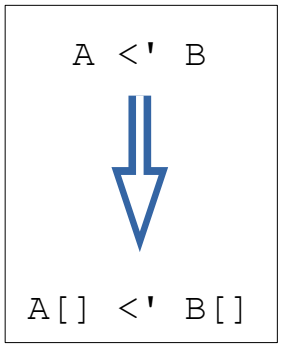
Sous-type Java direct S <' T
l'un des 8 cas de sous-typage Java présentés ci-dessous.



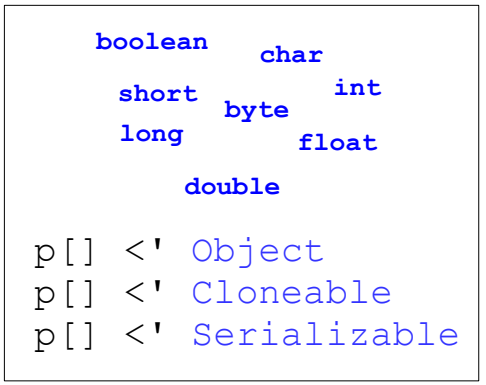









```
byte <' short
short <' int
int <' long
long <' float
float <' double
char <' int
```

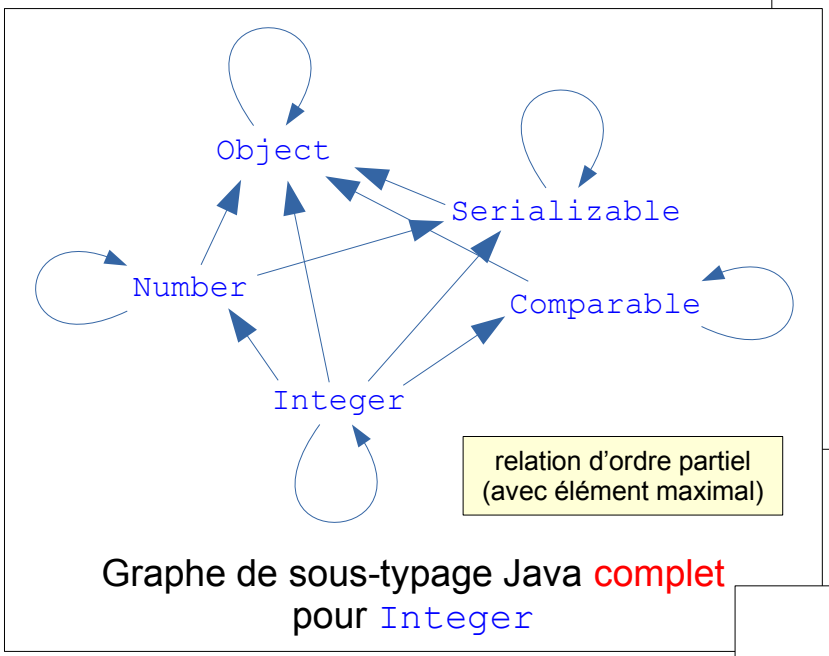



```
Object[] <' Object
Object[] <' Cloneable
Object[] <' Serializable
```

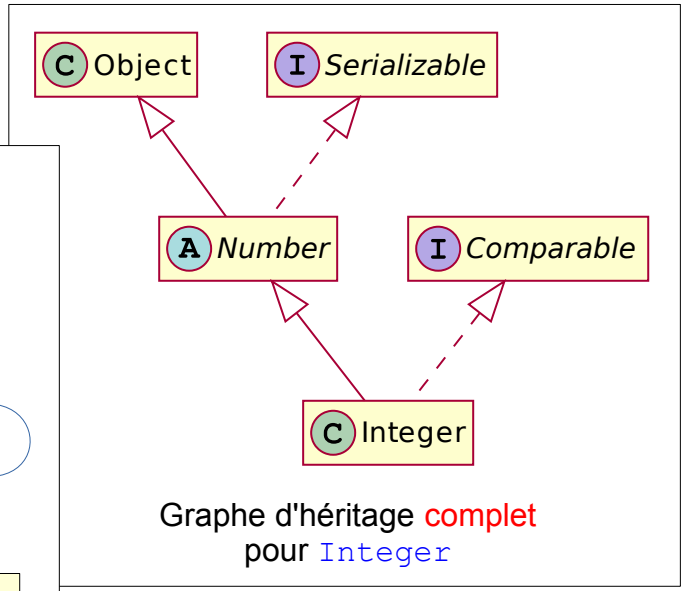
Rappels et compléments

Sous-type Java S <: T

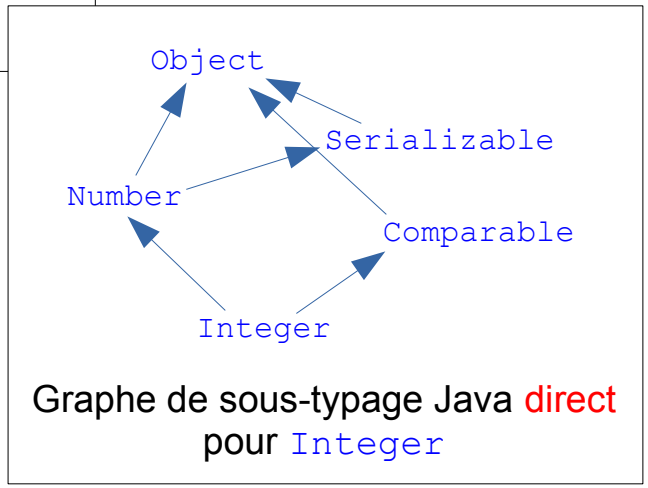
$(S = T) \vee (\exists Z)(S <' Z \wedge Z <: T)$



Graphe de sous-typage Java complet pour `Integer`



Graphe d'héritage complet pour `Integer`



Graphe de sous-typage Java direct pour `Integer`

Remarque : conséquences de `A <: B` ⇒ `A[] <: B[]`

```
void m() {
    Integer[] x = new Integer[1];
    Object[] y = x;
    y[0] = "Aïe";
    Integer n = x[0];
}
```

compilation → OK !

exécution → `ArrayStoreException`

compilation ↓

```
void m();
  Code:
    0: iconst_1
    1: anewarray #15    // class java/lang/Integer
    4: astore_1
    5: aload_1
    6: astore_2
    7: aload_2
    8: iconst_0
    9: ldc       #17    // String Aïe
   11: aastore
   12: aload_1
   13: iconst_0
   14: aaload
   15: astore_3
   16: return
```

`NullPointerException`	si `y` vaut **`null`**
`ArrayIndexOutOfBoundsException`	si indice hors domaine
`ArrayStoreException`	si valeur d'un type incompatible avec le type des éléments

`NullPointerException`	si `x` est **`null`**
`ArrayIndexOutOfBoundsException`	si indice hors domaine

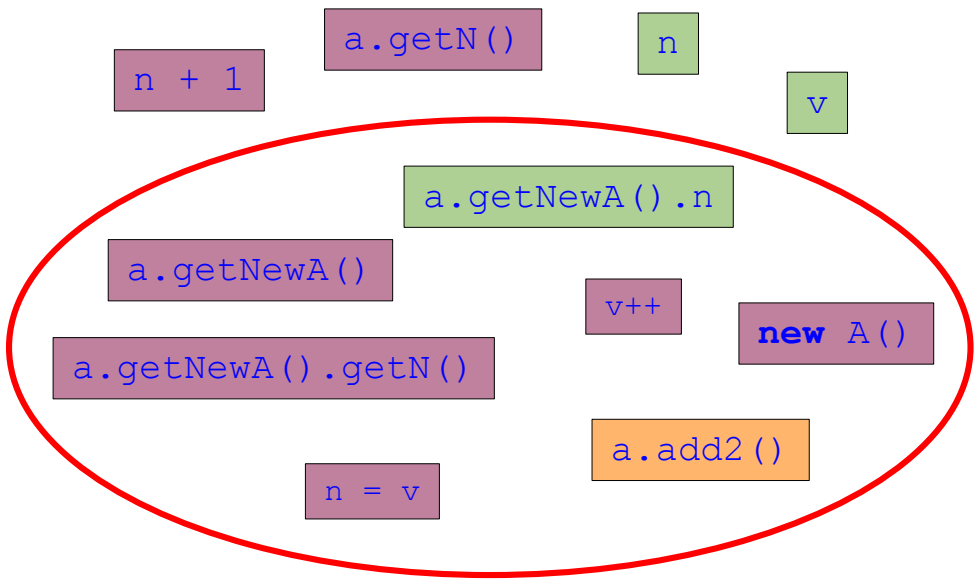
Expression : formule bien formée à partir de parenthèses, de littéraux et de noms (variables, opérateurs ou méthodes).

Évaluation d'une expression : calcul d'une expression par la *JVM* durant l'exécution d'un programme.

Ce calcul peut provoquer des *effets de bord*.

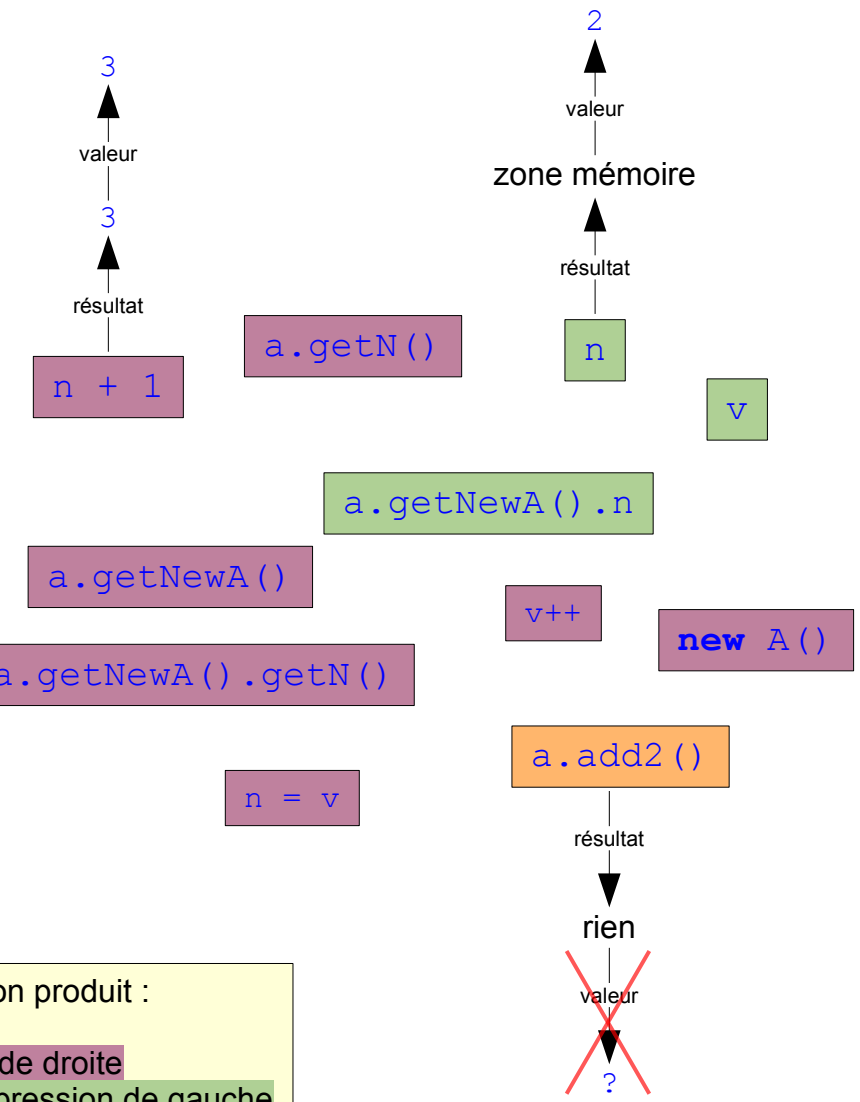
Le *résultat* de ce calcul peut représenter :
- une zone mémoire → expression pouvant aller à gauche ou à droite dans une affectation
- une valeur → expression ne pouvant aller qu'à droite dans une affectation
- rien → expression d'appel de méthode ne retournant pas de valeur (`void`)

```
class A {
    int n;
    int getN() {
        return n;
    }
    A getNewA() {
        return new A();
    }
    void add2() {
        int v = n + 1;
        v++;
        n = v;
    }
}
```



Valeur d'une expression : valeur du résultat (non vide) de son évaluation.

```
class A {
    int n;
    int getN() {
        return n;
    }
    A getNewA() {
        return new A();
    }
    void add2() {
        int v = n + 1;
        v++;
        n = v;
    }
}
```



E : « a.getNewA().n = v »

L'évaluation d'une expression d'affectation produit :
- un effet de bord
  par copie de la valeur de l'expression de droite
  dans le résultat de l'évaluation de l'expression de gauche
- et un résultat
  la valeur de l'expression de droite

La valeur d'une expression d'affectation
est la valeur de son expression de droite