

## Généricité

## III. Généricité contrainte

1. Contrainte générique
2. Bornes multiples

## IV. Joker

1. Définition
2. Bornes du joker
3. Types à joker borné
4. Sous-typage
5. Instanciation des types paramétrés

## V. Méthodes génériques

1. Motivation
2. Définition
3. Inférence des paramètres de types

## i. Algorithme d'inférence

## ii. Exemple 1

## iii. Exemple 2

## iv. Exemple 3

## 4. Capture du joker

## i. Motivation et définition

## ii. Exemples

## 5. Règles méthodologiques

## i. Méthode générique ou à joker ?

## ii. Utilisation des jokers bornés

## a. Règles 1 &amp; 2

## b. Règles 3 &amp; 4

## VI. Compléments

## 1. Tableaux et généricité

## 2. Exceptions et généricité

## 3. Classe Class&lt;E&gt;

## 4. Types énumérés

## i. Classe Enum&lt;E&gt;

## ii. Entête de la classe Enum

## iii. Exemple élaboré

**Inférence de type** : le compilateur sait inférer (= calculer) les valeurs des paramètres de type lors d'un appel de méthode générique.

```
<T> void addAll(T[] t, Collection<T> c)
```

```
Integer[] ia = new Integer[] { 1, 2 };
List<Number> nl = new ArrayList<Number>();
obj.<Number>addAll(ia, nl);
```

```
Integer[] ia = new Integer[] { 1, 2 };
List<Number> nl = new ArrayList<Number>();
obj.addAll(ia, nl);
```

le compilateur  
sait calculer  
 $T ::= \text{Number}$

$DT_i \rightarrow T_i \text{ extends } U_{i,1} \ \& \ \dots \ \& \ U_{i,h_i}$

Entête :  $\langle DT_1, \dots, DT_n \rangle R p(F_1 f_1, \dots, F_m f_m) \ (n > 0, m \geq 0)$

Appel :  $\text{obj}.p(e_1, \dots, e_m) \text{ avec } E_j = CT(e_j)$

algorithme d'inférence

Résultat :  $\langle Z_1, \dots, Z_n \rangle = \text{le plus spécifique des } \langle X_1, \dots, X_n \rangle$   
sous les contraintes  $E_j <: F_j[T_1|X_1, \dots, T_n|X_n] \ (1 \leq j \leq m)$

s'il reste des indices  $i$  tq  $Z_i$  n'a pas pu être déterminé ( $i \in I$ )

poursuite du calcul avec les contraintes

-  $X_i = Z_i \ (i \notin I)$

-  $\forall i \in I, X_i <: U_{i,h_i}[T_k | (k \in I ? X_k : Z_k)] \ (1 \leq h \leq h_i)$

-  $R[T_k | (k \in I ? X_k : Z_k)] <: S$

pour déterminer les  $Z_i$  manquants

$S$  est déterminé par  
le contexte d'appel

Entête :  $\langle T \rangle \text{ void addAll}(T[] t, \text{Collection}<T> c)$

Appel :  $\text{obj}.addAll(ia, nl)$

$(F_1, F_2) = (CT(t), CT(c)) = (T[], \text{Collection}<T>)$

$(E_1, E_2) = (CT(ia), CT(nl)) = (\text{Integer}[], \text{List}<\text{Number}>)$

algorithme d'inférence

résultat :  $Z$ , le plus spécifique des  $x$  tels que  
 $\text{Integer}[] <: X[]$   
 $\text{List}<\text{Number}> <: \text{Collection}<X>$

$T ::= Z$



## Généricité

### III. Généricité contrainte

1. Contrainte générique
2. Bornes multiples

### IV. Joker

1. Définition
2. Bornes du joker
3. Types à joker borné
4. Sous-typage
5. Instanciation des types paramétrés

### V. Méthodes génériques

1. Motivation
  2. Définition
  3. Inférence des paramètres de types
    - i. Algorithme d'inférence
    - ii. Exemple 1
    - iii. Exemple 2
    - iv. Exemple 3
  4. Capture du joker
    - i. Motivation et définition
    - ii. Exemples
  5. Règles méthodologiques
    - i. Méthode générique ou à joker ?
    - ii. Utilisation des jokers bornés
      - a. Règles 1 & 2
      - b. Règles 3 & 4
- ### VI. Compléments
1. Tableaux et généricité
  2. Exceptions et généricité
  3. Classe Class<E>
  4. Types énumérés
    - i. Classe Enum<E>
    - ii. Entête de la classe Enum
    - iii. Exemple élaboré

```
<T> void addAll(T[] t, Collection<T> c)
```

```
Integer[] ia = new Integer[] { 1, 2 };
List<Number> nl = new ArrayList<Number>();
obj.addAll(ia, nl);
```

$DT_i \rightarrow T_i$  **extends**  $U_{i,1} \& \dots \& U_{i,h_i}$

$n = 1, h_1 = 0, DT_1 \rightarrow T_1$   
( $T_1$  est noté  $T$  par la suite)

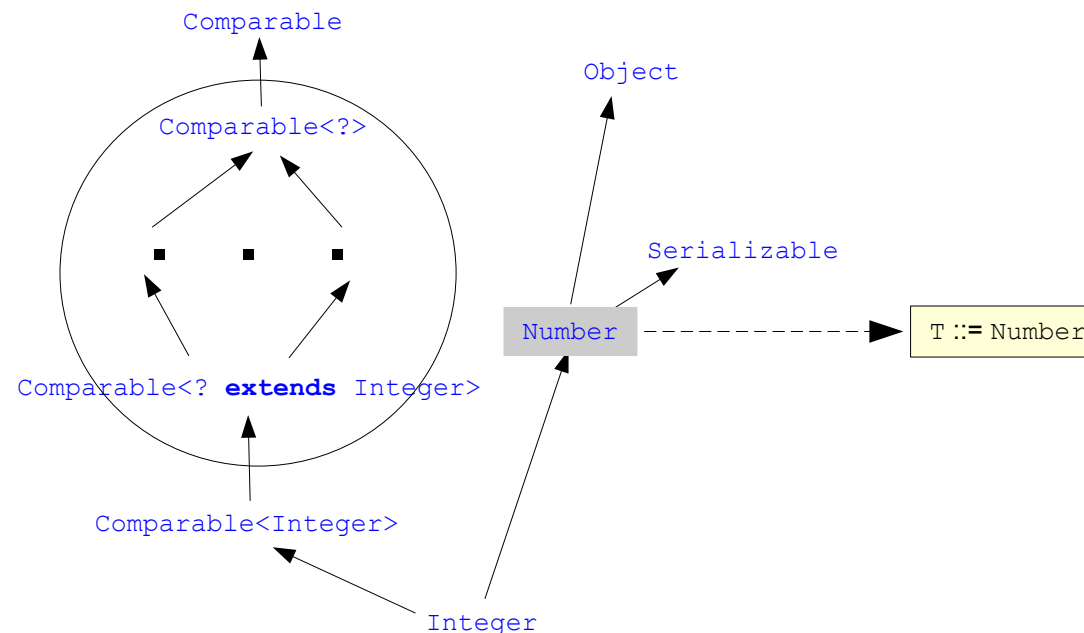
Entête :  $\langle DT_1, \dots, DT_n \rangle R p(F_1 f_1, \dots, F_m f_m) (n > 0, m \geq 0)$

Appel :  $obj.p(e_1, \dots, e_m)$  avec  $E_j = CT(e_j)$

$m = 2$   
 $(F_1, F_2) = (T[], Collection<T>)$   
 $(E_1, E_2) = (Integer[], List<Number>)$

Résultat :  $\langle Z_1, \dots, Z_n \rangle =$  le plus spécifique des  $\langle X_1, \dots, X_n \rangle$   
sous les contraintes  $E_j \prec: F_j[T_1 | X_1, \dots, T_n | X_n] (1 \leq j \leq m)$

Contraintes :  
 $Integer[] \prec: X[]$   
 $List<Number> \prec: Collection<X>$



## Généricité

### III. Généricité contrainte

1. Contrainte générique
2. Bornes multiples

### IV. Joker

1. Définition
2. Bornes du joker
3. Types à joker borné
4. Sous-typage
5. Instanciation des types paramétrés

### V. Méthodes génériques

1. Motivation
2. Définition
3. Inférence des paramètres de types
  - i. Algorithme d'inférence
  - ii. Exemple 1
  - iii. Exemple 2
  - iv. Exemple 3
4. Capture du joker
  - i. Motivation et définition
  - ii. Exemples
5. Règles méthodologiques
  - i. Méthode générique ou à joker ?
  - ii. Utilisation des jokers bornés
    - a. Règles 1 & 2
    - b. Règles 3 & 4
- VI. Compléments
  1. Tableaux et généricité
  2. Exceptions et généricité
  3. Classe Class<E>
  4. Types énumérés
    - i. Classe Enum<E>
    - ii. Entête de la classe Enum
    - iii. Exemple élaboré

```
<T extends Number> List<T> emptyList()
```

```
List<Integer> il = obj.emptyList();
```

$DT_i \rightarrow T_i \text{ extends } U_{i,1} \ \& \ \dots \ \& \ U_{i,h_i}$

$n = 1, h_1 = 1, DT_1 \rightarrow T_1 \text{ extends } \text{Number}$   
( $T_1$  est noté  $T$  par la suite)

Entête :  $\langle DT_1, \dots, DT_n \rangle R \ p(F_1 \ f_1, \dots, F_m \ f_m) \ (n > 0, m \geq 0)$   
Appel :  $\text{obj.p}(e_1, \dots, e_m) \text{ avec } E_j = CT(e_j)$

$m = 0$

Résultat :  $\langle Z_1, \dots, Z_n \rangle = \text{le plus spécifique des } \langle X_1, \dots, X_n \rangle$   
sous les contraintes  $E_j \prec: F_j[T_1|X_1, \dots, T_n|X_n] \ (1 \leq j \leq m)$

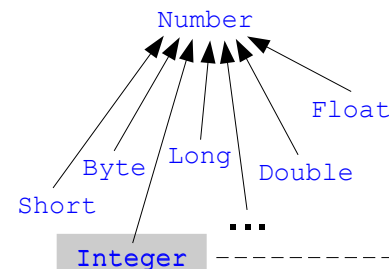
Aucune contrainte :  
 $Z$  reste indéterminé

poursuite du calcul avec les contraintes

- $X_i = Z_i \ (i \notin I)$
  - $\forall i \in I, X_i \prec: U_{i,h_i}[T_k | (k \in I?X_k:Z_k)] \ (1 \leq h \leq h_i)$
  - $R[T_k | (k \in I?X_k:Z_k)] \prec: S$
- pour déterminer les  $Z_i$  manquants

Nouvelles contraintes :

- $X \prec: \text{Number}$
- $\text{List}\langle X \rangle \prec: \text{List}\langle \text{Integer} \rangle$



## Généricité

## III. Généricité contrainte

1. Contrainte générique
2. Bornes multiples

## IV. Joker

1. Définition
2. Bornes du joker
3. Types à joker borné
4. Sous-typage
5. Instanciation des types paramétrés

## V. Méthodes génériques

1. Motivation
2. Définition
3. Inférence des paramètres de types

- i. Algorithme d'inférence
- ii. Exemple 1
- iii. Exemple 2
- iv. Exemple 3

## 4. Capture du joker

- i. Motivation et définition
- ii. Exemples

## 5. Règles méthodologiques

- i. Méthode générique ou à joker ?
- ii. Utilisation des jokers bornés
  - a. Règles 1 & 2
  - b. Règles 3 & 4

## VI. Compléments

1. Tableaux et généricité
2. Exceptions et généricité
3. Classe Class<E>
4. Types énumérés
  - i. Classe Enum<E>
  - ii. Entête de la classe Enum
  - iii. Exemple élaboré

```
<T> List<T> toList(T a, T b)
```

```
List<Object> ol = obj.toList(5, "5");
```

```
n = 1, h1 = 0, DT1 → T1
(T1 est noté T par la suite)
```

```
DTi → Ti extends Ui,1 & ... & Ui,hi
```

```
Entête : <DT1, ..., DTn> R p(F1 f1, ..., Fm fm) (n > 0, m ≥ 0)
```

```
Appel : obj.p(e1, ..., em) avec Ej = CT(ej)
```

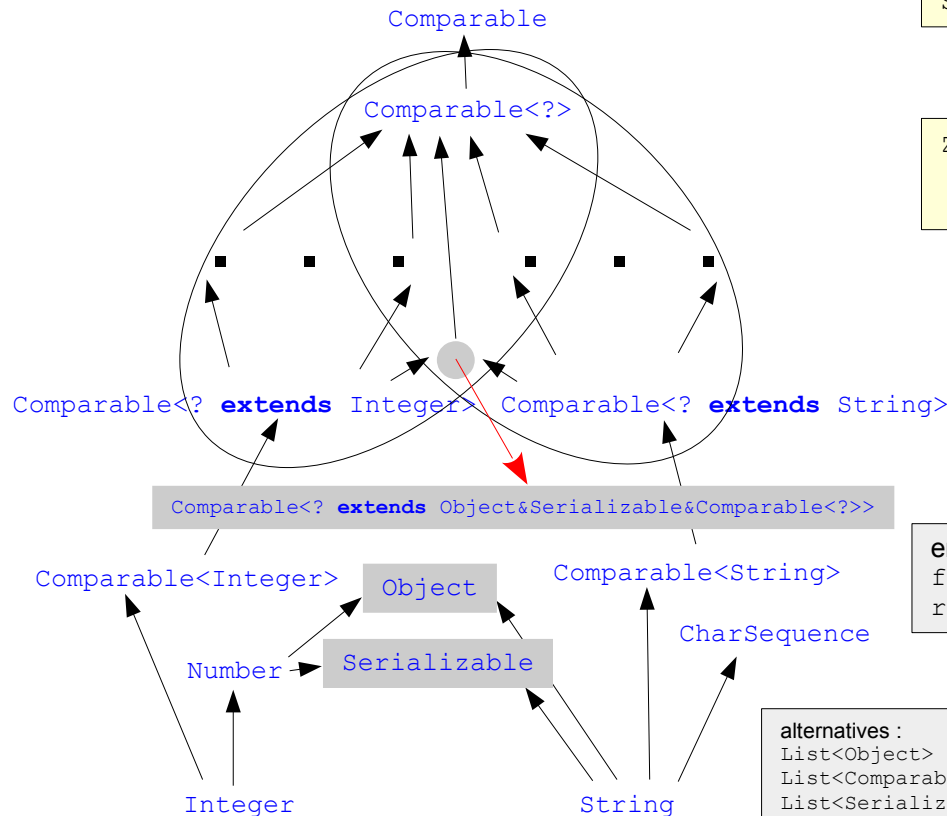
```
m = 2
(F1, F2) = (T, T)
(E1, E2) = (Integer, String)
```

```
Résultat : <Z1, ..., Zn> = le plus spécifique des <X1, ..., Xn>
```

```
sous les contraintes Ej <: Fj [T1 | X1, ..., Tn | Xn] (1 ≤ j ≤ m)
```

```
Contraintes :
Integer <: X
String <: X
```

```
Z = Object
  & Serializable
  & Comparable<?> ...>
```



```
List<Z> <: List<Object>
```

```
erreur à la compilation :
found : List<Object&Serializable&...>
required : List<Object>
```

```
alternatives :
List<Object> x = obj.<Object>toList(5, "5");
List<Comparable<?>> x = obj.<Comparable<?>>toList(5, "5");
List<Serializable> x = obj.<Serializable>toList(5, "5");
List<?> x = obj.toList(5, "5");
```

## Généricité

### III. Généricité contrainte

1. Contrainte générique
2. Bornes multiples

### IV. Joker

1. Définition
2. Bornes du joker
3. Types à joker borné
4. Sous-typage
5. Instanciation des types paramétrés

### V. Méthodes génériques

1. Motivation
2. Définition
3. Inférence des paramètres de types

- i. Algorithme d'inférence
- ii. Exemple 1
- iii. Exemple 2
- iv. Exemple 3

#### 4. Capture du joker

- i. Motivation et définition
- ii. Exemples

#### 5. Règles méthodologiques

- i. Méthode générique ou à joker ?
- ii. Utilisation des jokers bornés
  - a. Règles 1 & 2
  - b. Règles 3 & 4

### VI. Compléments

1. Tableaux et généricité
2. Exceptions et généricité
3. Classe `Class<E>`
4. Types énumérés
  - i. Classe `Enum<E>`
  - ii. Entête de la classe `Enum`
  - iii. Exemple élaboré

### *Capture de joker (wildcard capture) :*

lors d'un appel de méthode générique, pendant l'inférence de types, si  $E$  est de la forme  $H<?>$  et si  $F$  est de la forme  $G<T>$  alors le joker est remplacé par un nouveau nom de type de la forme `capture#n` et alors  $T ::= \text{capture}\#n$ .

```
void permuteHead(List<?> list) {  
    Object a = list.get(0);  
    Object b = list.get(1);  
    list.set(0, b);  
    list.set(1, a);  
}
```

erreur :  
impossible d'ajouter a ou b dans list

```
<T> void permuteHead(List<T> list) {  
    T a = list.get(0);  
    T b = list.get(1);  
    list.set(0, b);  
    list.set(1, a);  
}
```

correct car a, b, et list  
sont typés avec T

*capture de joker*  
durant l'inférence de types  
 $F = \text{List}<T>$   
 $E = \text{List}<?>$

```
public void permuteHead(List<?> list) {  
    pH(list);  
}  
  
private <T> void pH(List<T> list) {  
    T a = list.get(0);  
    T b = list.get(1);  
    list.set(0, b);  
    list.set(1, a);  
}
```

① ? est remplacé momentanément par  
le nom `capture#1`

② `z = capture#1`

③ `pH(list) → this.<capture#1>pH(list)`