

Slide 1

## Les enregistrements

Slide 2

### 0. Résumé des épisodes précédents

Slide 3

Structures de contrôle :

déclaration, affectation, séquence, test, boucle, fonction,  
récursivité

Deux fragments : impératif et fonctionnel

État : e, m

Signification des programme : calcul de valeur et transformation  
de la mémoires ( $\Theta$ ,  $\Sigma$ )

Structures de données : entiers, flottants, booléens, caractères

Slide 4

## I. Les enregistrements

### Des données composites

Une variable contient **un** entier, **un** flottant, **un** booléen ou **un** caractère

Slide 5

Impossible de représenter un objet comme "Bonjour",  
(48.715, 2.208, 156.0),  $3x^2 + 4$  formé de **plusieurs**  
entiers, flottants, booléens ou caractères

**Enregistrements**, tableaux, modules, ...

Chaque élément : un champ

### Des n-uplets à champs nommés

Un n-uplet est une fonction de domaine  $0 \dots n-1$

Un **enregistrement** une fonction de domaine fini

Le domaine de la fonction : les **étiquettes**

Slide 6

Exemple :

```
{latitude = 48.715,  
 longitude = 2.208,  
 altitude = 156.0}
```

### La définition d'un type enregistrement

```
class Point {  
    final double latitude;  
    final double longitude;  
    final double altitude;}
```

Slide 7

Pour chaque champ :

son étiquette, son type, s'il est mutable ou final

### L'allocation d'un enregistrement

```
Point x;
```

```
e = [x = r] m = [r = null]
```

Slide 8

### L'allocation d'un enregistrement

```
Point x;
```

```
e = [x = r] m = [r = null]
```

```
new Point ()
```

```
e = [x = r] m = [r = null, r' = {latitude =  
0.0, longitude = 0.0, altitude = 0.0}]
```

Slide 9

### L'allocation d'un enregistrement

```
Point x;
```

```
e = [x = r] m = [r = null]
```

```
x = new Point();
```

```
e = [x = r] m = [r = r', r' = {latitude =  
0.0, longitude = 0.0, altitude = 0.0}]
```

Slide 10

### L'allocation d'un enregistrement

```
Point x;
```

```
e = [x = r] m = [r = null]
```

```
x = new Point();
```

```
e = [x = r] m = [r = r', r' = {latitude =  
0.0, longitude = 0.0, altitude = 0.0}]
```

```
Quid de final x = new Point();?
```

Slide 11

### L'accès à un champ

```
x.latitude
```

```
System.out.println(x.latitude);
```

```
e = [x = r] m = [r = r', r' = {latitude =  
0.0, longitude = 0.0, altitude = 0.0}]
```

Slide 12

### L'accès à un champ

```
x.latitude
```

```
System.out.println(x.latitude);
```

Slide 13

```
e = [x = r] m = [r = r', r' = {latitude = 0.0, longitude = 0.0, altitude = 0.0}]
```

Si la valeur de `t` est une référence `r'` associée dans `m` à `{latitude = v1, longitude = v2, altitude = v3}`

alors la valeur de `t.latitude` est `v1`

### L'affectation d'un champ

Champs mutables

```
class Point {  
    final double latitude;  
    final double longitude;  
    final double altitude;}
```

Slide 14

## L'affectation d'un champ

Champs mutables

```
class Point {  
    double latitude;  
    double longitude;  
    double altitude;}
```

Slide 15

Pas de référence supplémentaire : la référence `x` suffit à rendre les champs mutables

## L'affectation d'un champ

```
x.latitude = 48.715;  
x.longitude = 2.208;  
x.altitude = 156.0;
```

Slide 16

## Les constructeurs

```
class Point {
  double latitude;
  double longitude;
  double altitude;

  Point (double a, double b, double c) {
    this.latitude = a;
    this.longitude = b;
    this.altitude = c;}}

Point x = new Point(48.715,2.208,156.0) ;
```

Slide 17

## La fonction $\Sigma$ (en exercice)

On ajoute un argument qui contient la liste des types définis

$\Theta(t, e, m, G, C) \Sigma(p, e, m, G, C)$

Quatre opérations :

- définir un type enregistrement (`class`)
- allouer un enregistrement (`new`)
- accéder à un champ (`.`)
- modifier un champ (`. =`)

Slide 18

Slide 19

## II. Le partage

### Le partage

```
e = [x = r1, y = r2]
m = [r1 = r3, r2 = r4, r3 = {latitude =
48.715, longitude = 2.208, altitude =
156.0}, r4 = {latitude = 0.0, longitude =
0.0, altitude = 0.0}]
```

Slide 20

On exécute

```
y = x;
```

### Le partage

Slide 21

```
e = [r = r1, y = r2]
m = [r1 = r3, r2 = r4, r3 = {latitude =
48.715, longitude = 2.208, altitude =
156.0}, r4 = {latitude = 0.0, longitude =
0.0, altitude = 0.0}]
```

On exécute

```
y.latitude = x.latitude ;
y.longitude = x.longitude ;
y.altitude = x.altitude ;
```

Slide 22

Dans

```
e = [x = r1, y = r2]
m = [r1 = r3, r2 = r4, r3 = {latitude =
48.715, longitude = 2.208, altitude =
156.0}, r4 = {latitude = 48.715, longitude =
2.208, altitude = 156.0}]
```

On exécute

```
x.latitude = 23.45 ;
System.out.println(y.latitude) ;
```

Slide 23

Dans

```
e = [x = r1, y = r2]
```

```
m = [r1 = r3, r2 = r3, r3 = {latitude =  
48.715, longitude = 2.208, altitude =  
156.0}]
```

On exécute

```
x.latitude = 23.45 ;  
System.out.println(y.latitude) ;
```

### L'égalité

Slide 24

`t` et `u` de type `Point`

La valeur de `(t == u)` est `true` si la valeur de `t` et celle de `u` sont **la même référence**

Slide 25

Dans

```
e = [x = r1, y = r2]
```

```
m = [r1 = r3, r2 = r3, r3 = {latitude =  
48.715, longitude = 2.208, altitude =  
156.0}]
```

On exécute

```
System.out.println(x == y);
```

Slide 26

Dans

```
e = [x = r1, y = r2]
```

```
m = [r1 = r3, r2 = r4, r3 = {latitude =  
48.715, longitude = 2.208, altitude =  
156.0}, r4 = {latitude = 48.715, longitude =  
2.208, altitude = 156.0}]
```

On exécute

```
System.out.println(x == y);
```

## L'égalité physique et l'égalité structurelle

Slide 27

```
static boolean equal(final Point x,
                    final Point y){
    return (x.latitude == y.latitude)
        && (x.longitude == y.longitude)
        && (x.altitude == y.altitude);}
```

## L'égalité physique, l'égalité structurelle et toutes les autres

$(90, 0, 0)$  et  $(90, 20, 0)$  sont égaux

$(0, 180, 0)$  et  $(0, -180, 0)$  sont égaux

Slide 28

### une autre égalité

L'égalité fait partie de la définition d'une structure (pas de quotients)

$(G, +, 0, -)$  v.s.  $(G, =, +, 0, -)$

$\mathbb{Q} = ((\mathbb{Z} \times \mathbb{Z}^*)/\mathbb{R}, +, 0, -, \times, 1, \mathbb{I})$

v.s.  $(\mathbb{Z} \times \mathbb{Z}^*, \mathbb{R}, +, 0, -, \times, 1, \mathbb{I})$

## Les types enveloppés

Type enregistrement à un seul champ

```
class Integer {int c;}
```

Slide 29

Idem `int`, mais autorise le partage. Dans

```
e = [x = r1, y = r2]    m = [r1 = r3, r2 =  
r4, r3 = {c = 4}, r4 = {c = 4}]
```

On exécute `x.c = 5; System.out.println(y.c);`

```
Et dans e = [x = r1, y = r2]    m = [r1 = r3, r2  
= r3, r3 = {c = 4}]
```

## Les types enveloppés et l'appel par référence

```
static void swap(Integer x, Integer y) {  
    int z = x.c; x.c = y.c; y.c = z;}  
}
```

Slide 30

```
public static void main (String [] args) {  
    a = new Integer(4);  
    b = new Integer(7);  
    swap (a,b);  
    System.out.println(a.c + " " + b.c);}  
}
```

Slide 31

### III. Caml et C

Quelles questions poser ?

Slide 32

Comment

- définir un type enregistrement ? (`class`)
- allouer un enregistrement ? (`new`)
- accéder à un champ ? (`.`)
- modifier un champ ? (`. =`)

## Caml

Slide 33

```
type point = {  
  mutable latitude : float;  
  mutable longitude : float;  
  mutable altitude : float;}  
  
let x = ref {latitude = 48.715;  
            longitude = 2.208;  
            altitude = 156.0;}  
  
!x.latitude  
  
!x.latitude <- 23.45
```

## Caml

Slide 34

```
{latitude = 48.715;  
 longitude = 2.208;  
 altitude = 156.0;}
```

Pas de `new({})`

Pas de définition de constructeur

Pas de `null`

Pas de valeur par défaut

C

Slide 35

```
struct Point {
    double latitude;
    double longitude;
    double altitude;};

struct Point x = {48.715,2.208,156.0};

x.latitude

x.latitude = 23.45;
```

C

Slide 36

Pas de `new`, pas d'allocation de cellules, pas de `null`

```
x = new Point (48.715,2.208,156.0);

e = [x = r] m = [r = r', r' = {latitude =
48.715, longitude = 2.208, altitude =
156.0}]

struct Point x = {48.715,2.208,156.0};
```

```
e = [x = r] m = [r = {latitude = 48.715,  
longitude = 2.208, altitude = 156.0}]
```

Slide 37

```
void tropic (struct Point y){  
    y.latitude = 23.45;}  
  
tropic(x);
```

recopie l'enregistrement champ à champ

ne modifie pas la valeur de `x`

pour modifier un champ : passage par référence

```
void tropic (struct Point* y){  
    (*y).latitude = 23.45;}  
  
tropic(&x);
```

Slide 38

## IV. Les tableaux

### Champs nommés et numérotés

Les quatre mêmes questions

Pas de **définition** de type `T []` (`int []`)

**Allocation** `new int [100]`

Taille (invariable) déterminée au moment de l'allocation

**Accès** : `t[50]`, `t[n]`

**Affectation d'un champ** : `t[50] = 10 ;`,

Slide 41

```
int c = 0;
for (int i = 0; i < 110; i = i + 1) {
    int s = x[i] + y[i] + c;
    z[i] = s % 10;
    c = s / 10;}
```

Slide 42

```
int c = 0;
for (int i = 0; i < 110; i = i + 1) {
    int s = x[i] + y[i] + c;
    z[i] = s % 10;
    c = s / 10;}
```

3.141 592 653 589 793 238 462 643 383 279 502 884 197 169  
399 375 105 820 974 944 592 307 816 406 286 208 998 628 034  
825 342 117 067 9

**Slide 43**

Exercices 4.2 et 4.3

**Slide 44**

La prochaine fois : les types de données dynamiques