

Algorithmique et Programmation

Introduction aux classes

IUT Informatique de Bordeaux

Plan du cours

- 1 Types de base : les limites
- 2 Déclaration et initialisation d'objets
 - Classes, objets, instances
 - Déclarer et initialiser un objet et ses attributs
 - Constructeurs maison
- 3 Références
- 4 Des objets comme attributs

Types primitifs : rappels

Jusqu'ici nous avons utilisé 4 types de base en Processing :

- `int`
- `float`
- `char`
- `boolean`

(et parfois `color`, qui encode un `int`)

Types primitifs : rappels

Jusqu'ici nous avons utilisé 4 types de base en Processing :

- `int`
- `float`
- `char`
- `boolean`

(et parfois `color`, qui encode un `int`)

Se limiter à ces types pose quelques problèmes.

Types primitifs : limites en Processing

Lisibilité

En Processing,

- beaucoup de variables accessibles dans tout le programme, et
- beaucoup de paramètres dans les fonctions.

Exemple

Tester si 2 rectangles se touchent : 8 paramètres.

```
1 boolean intersectionRect(int r1x, r1y, r1w, r1h, r2x, r2y, r2w, r2h) { ... }
```

Types primitifs : limites en Processing

Entrée/sortie

Le passage de paramètre par valeur permet de passer des variables

- en **entrée** (lecture),
- mais **pas** en **entrée/sortie** (lecture/écriture).

Rappelez-vous... (cf slide suivant)

La transmission par valeur

```
1 void changer(int first, int b)
2 {
3     int temp;
4     temp=first;
5     first=b;
6     b=temp;
7     println("Dedans : first vaut ", first, " et b vaut ", b);
8 }
9
10 void setup()
11 {
12     int first=3;
13     int last=5;
14     println("Avant : first vaut ", first, " et last vaut ", last);
15     changer(first, last);
16     println("Après : first vaut ", first, " et last vaut ", last);
17 }
```

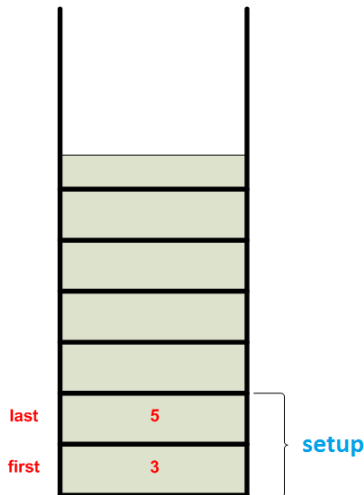
```
Avant : first vaut 3 et last vaut 5
Dedans : first vaut 5 et last vaut 3
Après : first vaut 3 et last vaut 5
```

La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```

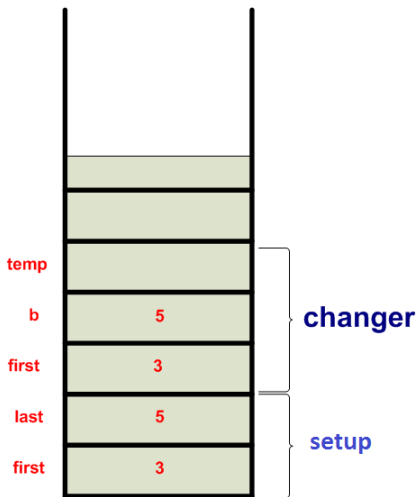


La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```

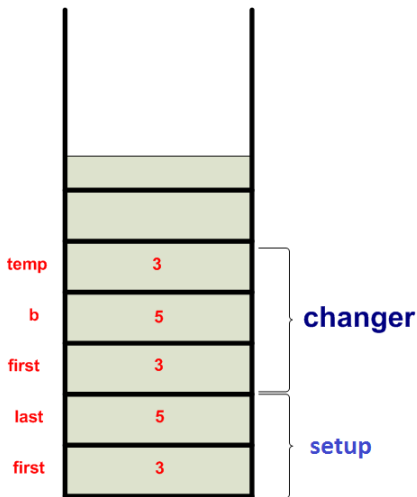


La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```

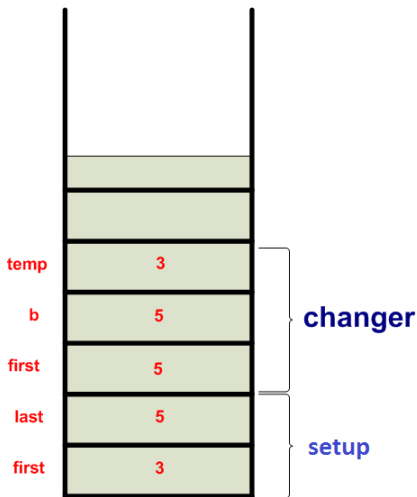


La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```

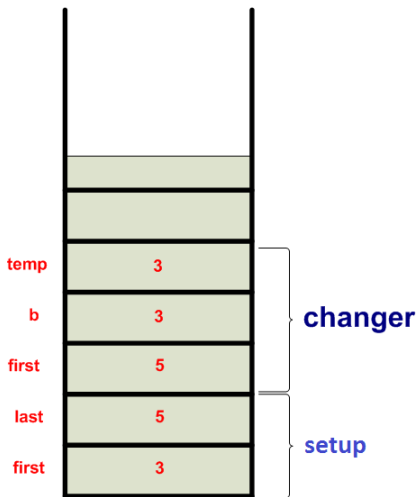


La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```

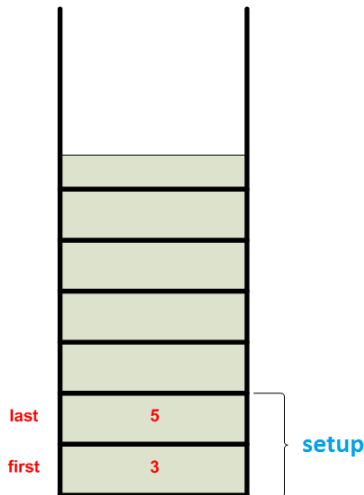


La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```



La transmission par valeur : exercice

Qu'affiche ce programme ?

```
1 void afficher(float moy, float min, float max) {
2     println("moy = " + moy + " / min = " + min + " / max = " + max);
3 }
4
5 void calculer(float a, float b, float moy, float min, float max) {
6     moy = (a+b) / 2;
7     if (a < b) {
8         min = a;
9         max = b;
10    } else {
11        min = b;
12        max = a;
13    }
14    afficher(moy, min, max);
15 }
16
17 float calculerMoy(float a, float b) {
18     float moy = (a + b) / 2;
19     afficher(moy, 0, 0);
20     return moy;
21 }
22
23 void setup() {
24     float a = 14.2;
25     float b = 15.8;
26     float moy = 0.;
27     float min = 0.;
28     float max = 0.;
29     calculer(a, b, moy, min, max);
30     afficher(moy, min, max);
31     moy = calculerMoy(a, b);
32     afficher(moy, min, max);
33 }
```

Classes : intérêt

Les classes vont nous permettre (notamment) de :

- **regrouper** les données, et
- les passer aux fonctions en **entrée/sortie**.

Plan du cours

- 1 Types de base : les limites
- 2 Déclaration et initialisation d'objets
 - Classes, objets, instances
 - Déclarer et initialiser un objet et ses attributs
 - Constructeurs maison
- 3 Références
- 4 Des objets comme attributs

Classe : définition

Une classe permet de regrouper des informations concernant une “entité” (exemple ici : une salle, ou un rectangle).

Définir une classe

```
1  class Salle {  
2      int capacite;  
3      boolean salleMachine;  
4      boolean libreService;  
5      int numero;  
6  }
```

```
1  class Rectangle {  
2      int x;  
3      int y;  
4      int largeur;  
5      int hauteur;  
6  }
```

Comme pour les fonctions, ces déclarations se font n'importe où dans le code, sauf à l'intérieur de fonctions / classes / etc.

Vocabulaire

Les champs `capacite`, `salleMachine`, `salleLibreService` et `numero` sont les **attributs** de la **classe** `Salle`.

Classe : définition

Une classe n'est pas un fourre-tout !

On y regroupe les variables qui concernent la même chose, pas "toutes les variables du programme".

Exercice

Une agence de voyage souhaite un logiciel sur mesure pour gérer ses clients et leurs voyages.

Chaque client est caractérisé par :

- un prénom
- un nom
- un numéro de client

Chaque destination est caractérisée par :

- un lieu
- une durée (en jours)
- une capacité (nb participants max)

Définissez une classe `Client` et une classe `Destination` correspondantes.

Classe : définition

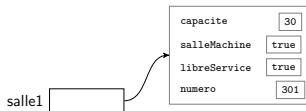
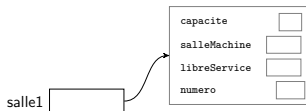
```
1  class Salle {  
2      int capacite;  
3      boolean salleMachine;  
4      boolean libreService;  
5      int numero;  
6  }
```

Une classe constitue un
nouveau type de données.

Nous allons voir comment :

- 1 *déclarer* une variable de type Salle
- 2 *initialiser* cette variable
- 3 *utiliser* ses attributs

salle1 → null



1. Déclarer une variable

Déclarer une variable

```
1 Salle salle1;
```

2. Initialiser une variable

Pour initialiser une variable, on utilise un **constructeur**.

Par défaut, il y a toujours un constructeur sans paramètre :

Constructeur par défaut

```
1 Salle salle1;  
2 Salle salle2;  
3 salle1 = new Salle();  
4 salle2 = new Salle();
```

ou

```
1 Salle salle1 = new Salle();  
2 Salle salle2 = new Salle();
```


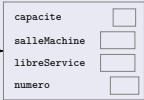
2. Initialiser une variable

Vocabulaire

- avant son initialisation, toute variable de type "objet" prend la valeur **null**.

1 `Salle salle1;` salle1  → null

- une fois initialisée, la variable `salle1` contient une **référence** vers une **instance** de la **classe** `Salle` (nous verrons pourquoi ensuite).

1 `salle1 = new Salle();` salle1  → 

On dit aussi que `salle1` est un **objet** de **type** `Salle`.

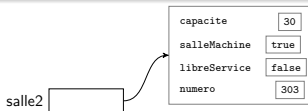
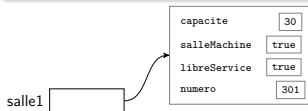
3. Accéder aux attributs d'une variable

Ensuite on accède à chaque attribut avec la notation `instance.attribut` :

Accès aux attributs

```
1  salle1.capacite = 30;  
2  salle1.salleMachine = true;  
3  salle1.libreService = true;  
4  salle1.numero = 213;
```

```
1  salle2.capacite = 30;  
2  salle2.salleMachine = true;  
3  salle2.libreService = false;  
4  salle2.numero = 206;
```




Exercice

En reprenant l'exercice sur l'agence de voyage, créez :

- un client nommé "Archibald Haddock", de numéro 17
- une destination nommée "Syldavie", durant 8 jours et pouvant accueillir 25 participants.

Constructeurs maison

Pour obtenir : salle1



capacite	30
salleMachine	true
libreService	true
numero	301

plutôt que de faire :

```
1 Salle salle1 = new Salle();  
2 salle1.capacite = 30;  
3 salle1.salleMachine = true;  
4 salle1.libreService = true;  
5 salle1.numero = 301;
```

on aimerait faire :

```
1 Salle salle1 = new Salle(30, true, true, 301);
```

→ Pour cela on peut définir ses propres constructeurs !

Constructeurs maison

Définir un nouveau constructeur

Déclaration du constructeur :

```
1 Salle(int uneCapacite, boolean estUneSalleMachine,  
2     boolean estEnLibreService, int unNumero) {  
3     capacite = uneCapacite;  
4     salleMachine = estUneSalleMachine;  
5     libreService = estEnLibreService;  
6     numero = unNumero;  
7 }
```

Utilisation du constructeur :

```
1 Salle salle1 = new Salle(30, true, true, 301);
```

Redéfinir le constructeur par défaut (si on veut)

Si, par exemple, on connaît des valeurs par défaut pour tous les attributs :

```
1 Salle() {  
2     capacite = 30;  
3     salleMachine = true;  
4     libreService = false;  
5     numero = 0;  
6 }
```

```
1 Salle salle1 = new Salle();  
2 salle1.numero = 301; // on ecrase
```

Constructeurs maison

Où placer un constructeur ?

→ dans la classe !

```
1  class Salle {  
2  
3      // Attributs  
4      int    capacite;  
5      boolean salleMachine;  
6      boolean libreService;  
7      int    numero;  
8  
9      // Un premier constructeur  
10     Salle() {  
11         capacite = 30;  
12         salleMachine = true;  
13         libreService = false;  
14         numero = 0;  
15     }  
16  
17     // Un deuxieme constructeur  
18     Salle(int uneCapacite, boolean estUneSalleMachine,  
19           boolean estEnLibreService, int unNumero) {  
20         capacite = uneCapacite;  
21         salleMachine = estUneSalleMachine;  
22         libreService = estEnLibreService;  
23         numero = unNumero;  
24     }  
25 }
```

Exercice

En reprenant l'exercice sur l'agence de voyage, créez :

- un constructeur pour la classe `Client`, prenant en paramètre tous ses attributs
- un constructeur pour la classe `Destination`, prenant uniquement le lieu en paramètre, et mettant par défaut 8 jours comme durée, et 25 participants comme capacité.
- une destination "Bordurie" utilisant ce nouveau constructeur
- un client "Tryphon Tournesol" de numéro 23.

Indiquez où se placent ces lignes de code par rapport à votre code précédent.

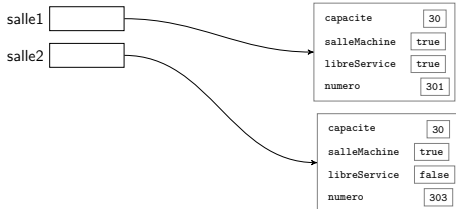
Plan du cours

- 1 Types de base : les limites
- 2 Déclaration et initialisation d'objets
 - Classes, objets, instances
 - Déclarer et initialiser un objet et ses attributs
 - Constructeurs maison
- 3 **Références**
- 4 Des objets comme attributs

Références

En Java (et donc Processing), on manipule toujours des **références** vers les objets (instances) :

```
1 Salle salle1 = new Salle( 30, true, true, 301);  
2 Salle salle2 = new Salle( 30, true, false, 303);
```

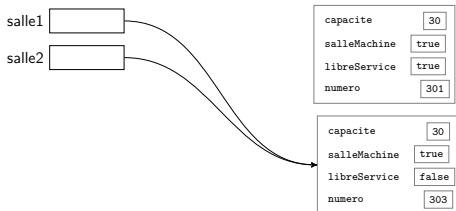


Références

```
1 Salle salle1 = new Salle( 30, true, true, 301);  
2 Salle salle2 = new Salle( 30, true, false, 303);
```



```
1 salle1 = salle2;
```



Passage de référence en paramètre

Une conséquence est que les **objets passés en paramètre** d'une fonction peuvent être **modifiés** (lecture / écriture).

```
1  /**
2   * Demenage une salle d'un etage vers un nouvel etage.
3   */
4  void demenage(Salle s, int etageAvant, int etageApres) {
5      // si la salle est a l'etage concerne
6      if (s.numero / 100 == etageAvant) {
7          // on demenage ! 301 devient 201
8          s.numero = (s.numero % 100) + (etageApres * 100);
9      }
10 }
11 void setup() {
12     Salle salle1 = new Salle( 30, true, true, 301);
13     Salle salle2 = new Salle( 30, true, false, 303);
14     int etageSrc = 3;
15     int etageDst = 2;
16     println("Numero avant :", salle1.numero);
17     demenage(salle1, etageSrc, etageDst);
18     println("Numero apres :", salle1.numero);
19 }
```

```
Numero avant : 301
Numero apres : 201
```



Console



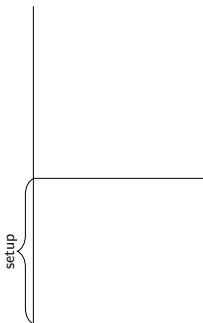
Erreurs

→ pourquoi ?

Passage de référence en paramètre

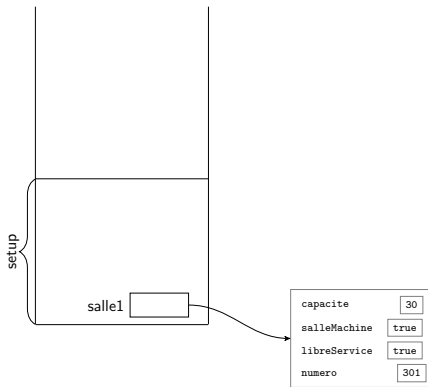
```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}
```

```
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



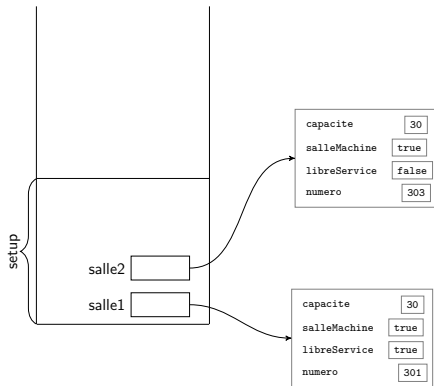
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



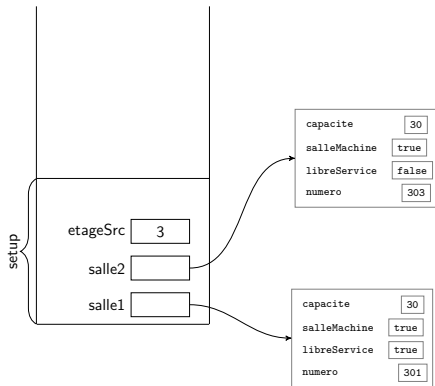
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



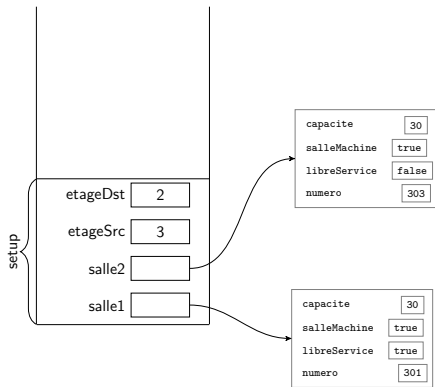
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```

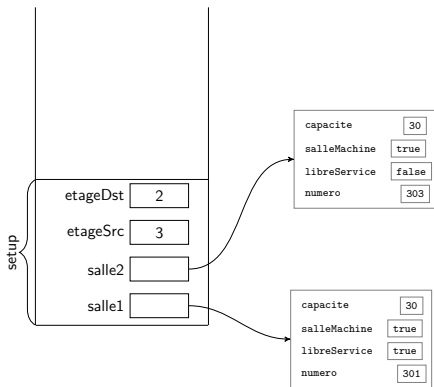
Numéro avant : 301



Console

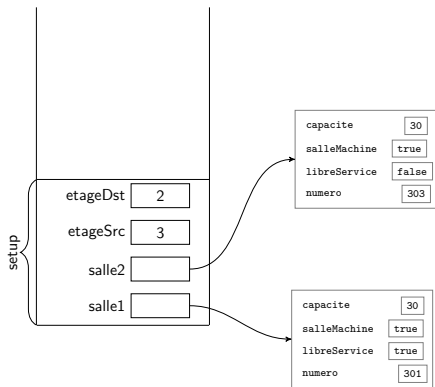


Erreurs



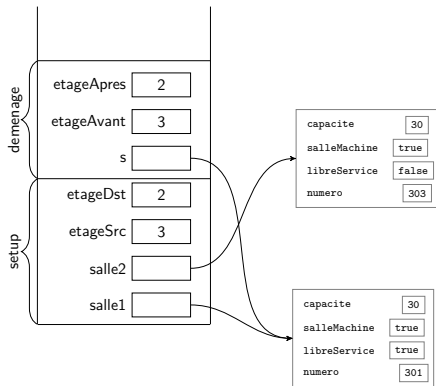
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



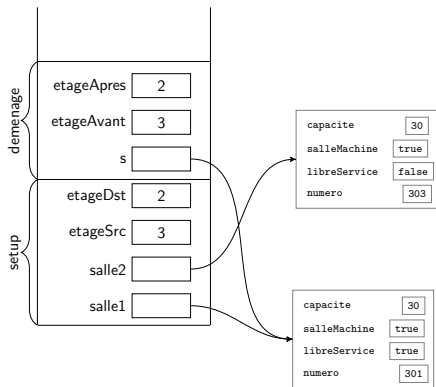
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



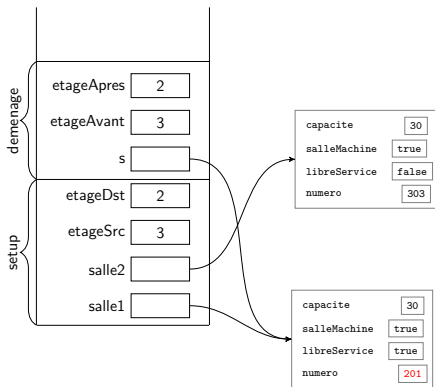
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```

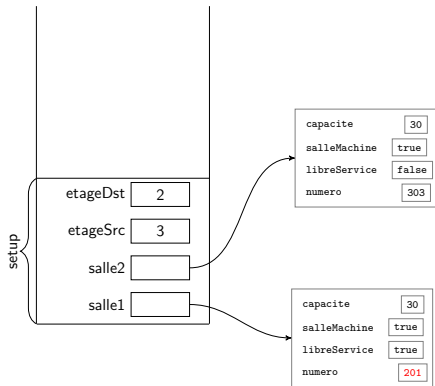


Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}
```

```
}
```

```
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = new Salle( 30, true, true, 301);  
    Salle salle2 = new Salle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```

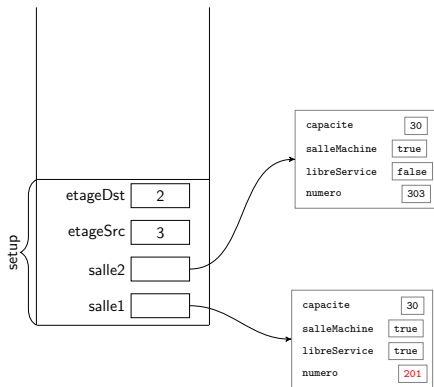
```
Numéro avant : 301  
Numéro après : 201
```



Console



Erreurs



Exercice

Suite de l'exercice sur l'agence de voyage :

Écrire une action permettant de modifier la durée d'une destination.

Plan du cours

- 1 Types de base : les limites
- 2 Déclaration et initialisation d'objets
 - Classes, objets, instances
 - Déclarer et initialiser un objet et ses attributs
 - Constructeurs maison
- 3 Références
- 4 Des objets comme attributs

Un attribut peut être de n'importe quel type...

... y compris un objet, un tableau d'objets, etc.

```
1  class Salle {
2      int         capacite;
3      boolean     salleMachine;
4      boolean     libreService;
5      int         numero;
6      Ordinateur  ordiVideoProj; // l'ordinateur connecte au video-projecteur
7      Ordinateur[] ordisEtudiants; // tableau d'ordinateurs
8
9      public static void main(String[] args) {
10         salle301 = new Salle();
11         salle301.ordiVideoProj = new Ordinateur();
12         salle301.ordisEtudiants = new Ordinateur[30];
13         salle301.ordisEtudiants[0] = new Ordinateur();
14         ...
15     }
16 }
```

```
1  class Ordinateur {
2      int     modele;
3      String  nomHote;
4      OS      systeme;
5  }
```

```
1  class OS {
2      ...
3  }
```


Exercice

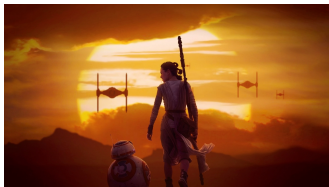
Suite de l'exercice sur l'agence de voyage :

Un voyage consiste en :

- une destination
 - des participants (tous sont clients de l'agence)
 - une date de début (stockée dans un entier, pour simplifier)
-
- Étendez le code précédent pour inclure la notion de voyage.
 - Écrivez une fonction `afficherVoyage` affichant toutes les informations d'un voyage.
 - Créez un voyage en Bordurie avec pour participants Haddock et Tournesol, dont la date vaut 74.
 - Affichez ce voyage.

Conclusion

That's all, folks !



Enfin, ce n'est que le début...

Pour l'instant nous n'utiliserons que des objets :

- avec uniquement des **attributs** et des **constructeurs**,
- **que nous définirons**, et pas ceux déjà implémentés dans Java/Processing.

Nous verrons plus tard, et surtout au S2, ce que permet la **programmation objet**.