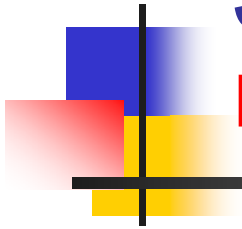
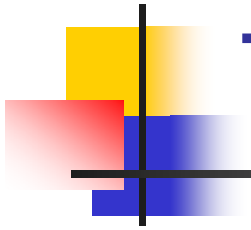


Java

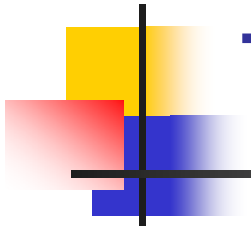
Licence Professionnelle CISII, 2009-10

Cours 3 : Autres types de données



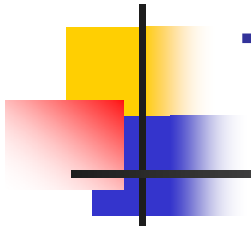
Types de données en Java

- Les constantes
  - Contiennent une valeur fixe et sont caractérisées par un identificateur et un type de données à l'instar des variables
  - Les constantes peuvent être déclarées dans les classes ou dans les méthodes en utilisant le modificateur *final*
    - final String COULEUR_NOIRE = "#000000";
    - final int CODE = 1537503;
    - final double PI = 3.14;
  - Par convention, le nom des constantes est toujours en majuscules afin de les distinguer sans équivoques des variables
  - La tentative de modifier une constante dans le programme entraînera une erreur lors de la compilation



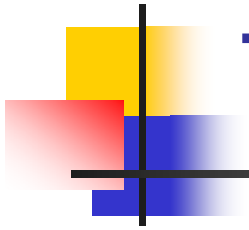
Types de données en Java

- Constantes nombres
  - Une constante « entière » est de type **long** si elle est suffixée par « L » et de type **int** sinon
  - Une constante « flottante » est de type **float** si elle est suffixée par « F » et de type **double**
  - Exemples:
    - 35
    - 2589L // constante de type long
    - 4.567e2 // 456,7 de type double
    - .123587E-25F // de type float
    - 012 // 12 en octal = 10 en décimal
    - 0xA7 // A7 en hexa = 167 en décimal



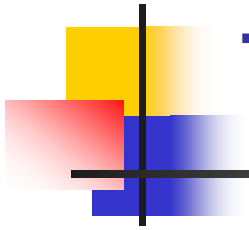
Types de données en Java

- Constantes de type caractère : Précautions
  - Un caractère Unicode entouré par « ’ »
  - CR et LF interdits (caractères de fin de ligne) **’A’**
  - **’\t’ ’\n’ ’\r’ ’\\’ ’\’’ ’\"’**
  - **’\u03a9’** (**\u** suivi du code hexadécimal d’un caractère Unicode)



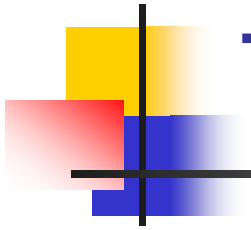
Types de données en Java

- Autres constantes
  - Type booléen
    - **false**
    - **true**
  - Référence inexistante (indique qu'une variable de type non primitif ne référence rien); convient pour *tous les types non primitifs*
    - **null**



Types de données en Java

- Valeurs par défaut
  - Si elles ne sont pas initialisées, les variables d'instance ou de classe (pas les variables locales d'une méthode) reçoivent par défaut les valeurs suivantes :
    - boolean — false
    - char — '\u0000'
    - Entier (byte short int long) — 0 0L
    - Flottant (float double) — 0.0F 0.0D
    - Référence d'objet — null

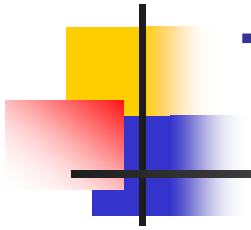


Types de données en Java

- Variable de classe **final**
  - Une variable de classe **static final** est constante dans tout le programme
  - Exemple :

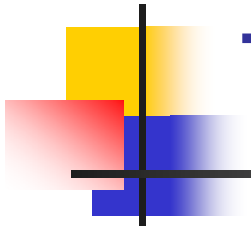
    ```
    static final double PI = 3.14;
    ```

  - Une variable de classe **static final** peut ne pas être initialisée à sa déclaration mais elle doit alors recevoir sa valeur dans un bloc d'initialisation **static**



Types de données en Java

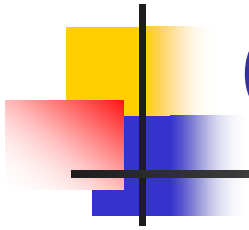
- Variable d'instance **final**
  - Une variable *d'instance* (pas **static**) **final** est constante pour chaque instance ; mais elle peut avoir 2 valeurs différentes pour 2 instances
  - Une variable d'instance **final** peut ne pas être initialisée à sa déclaration mais elle doit avoir une valeur à la sortie de tous les constructeurs



Types de données en Java

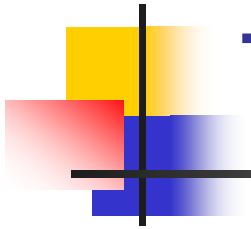
- Variable **final**
  - Si la variable est d'un type primitif, sa valeur ne peut changer
  - Si la variable référence un objet, elle ne pourra référencer un autre objet mais l'état de l'objet pourra être modifié

```
final Employe e = new Employe("Bibi");
. . .
e.nom = "Toto"; // Autorisé !
e.setSalaire(12000); // Autorisé !
e = new Employe("Bob"); // Interdit
```



Gestion de la mémoire

- L'environnement virtuel
  - Le programme en charge de la JVM créé un environnement d'exécution complet, prenant en charge la plupart des fonctionnalités d'un système d'exploitation
  - La complexité de cette émulation est répartie dans quatre unités liées :
    - les registres (*registers*)
    - la pile (*stack*)
    - le tas (*heap*)
    - et la zone de méthodes

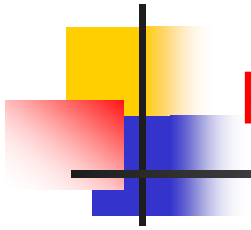


Types de données en Java

- La pile et le tas
  - L'espace mémoire alloué à une variable locale est situé dans la pile
  - Si la variable est d'un type primitif, sa valeur est placée dans la pile
  - Sinon la variable contient une référence à un objet
  - la valeur de la référence est placée dans la pile mais l'objet référencé est placé dans le tas
  - Lorsque l'objet n'est plus référencé, un « ramasse-miettes » (*garbage collector*, GC) libère la mémoire qui lui a été allouée

Types de données en Java

Pile et tas



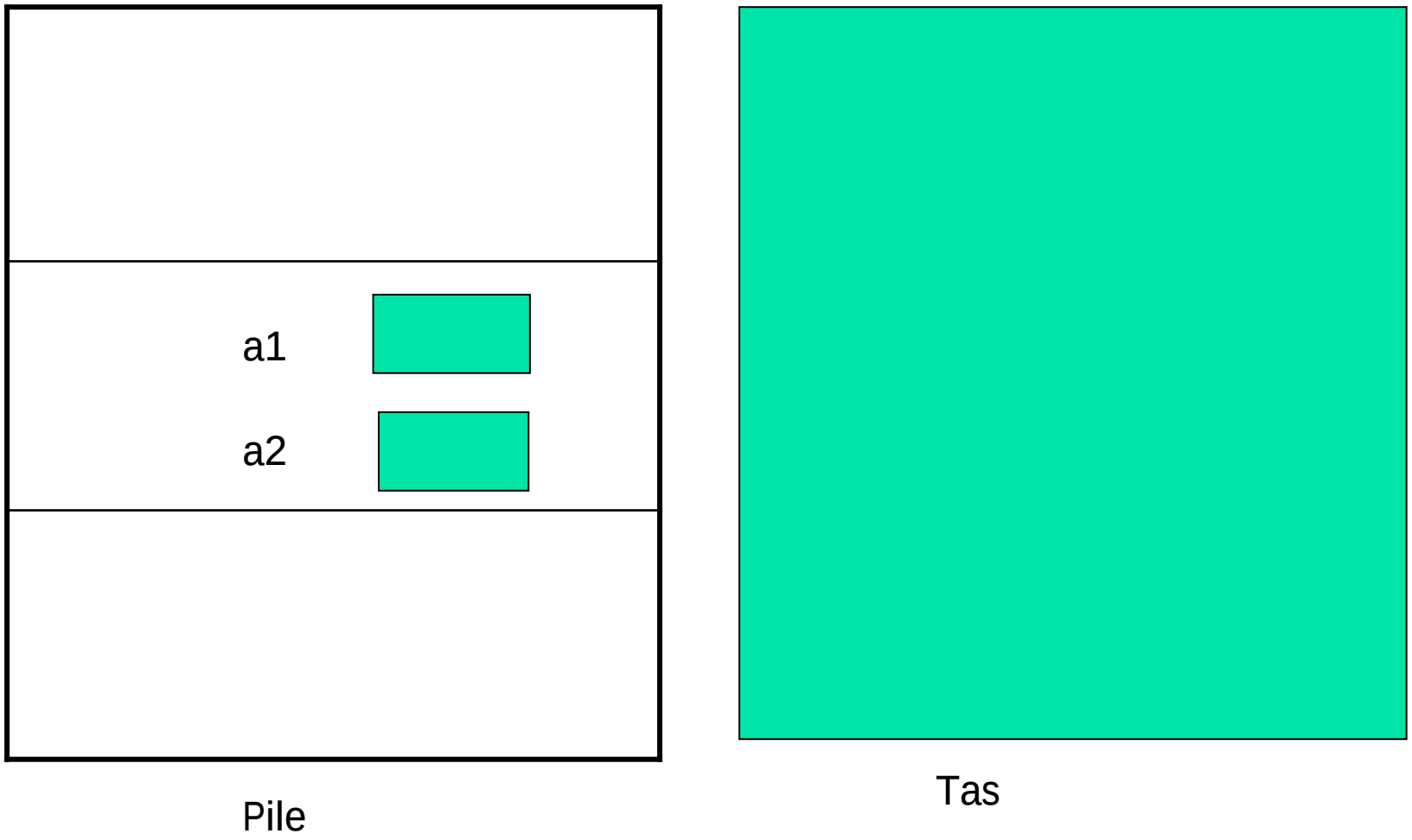
- Exemple d'utilisation des références

```
int m() {
   A a1, a2;
   a1 = new A();
   a2 = a1;
   ...
}
```

- Que se passe-t-il lorsque la méthode **m()** est appelée ?

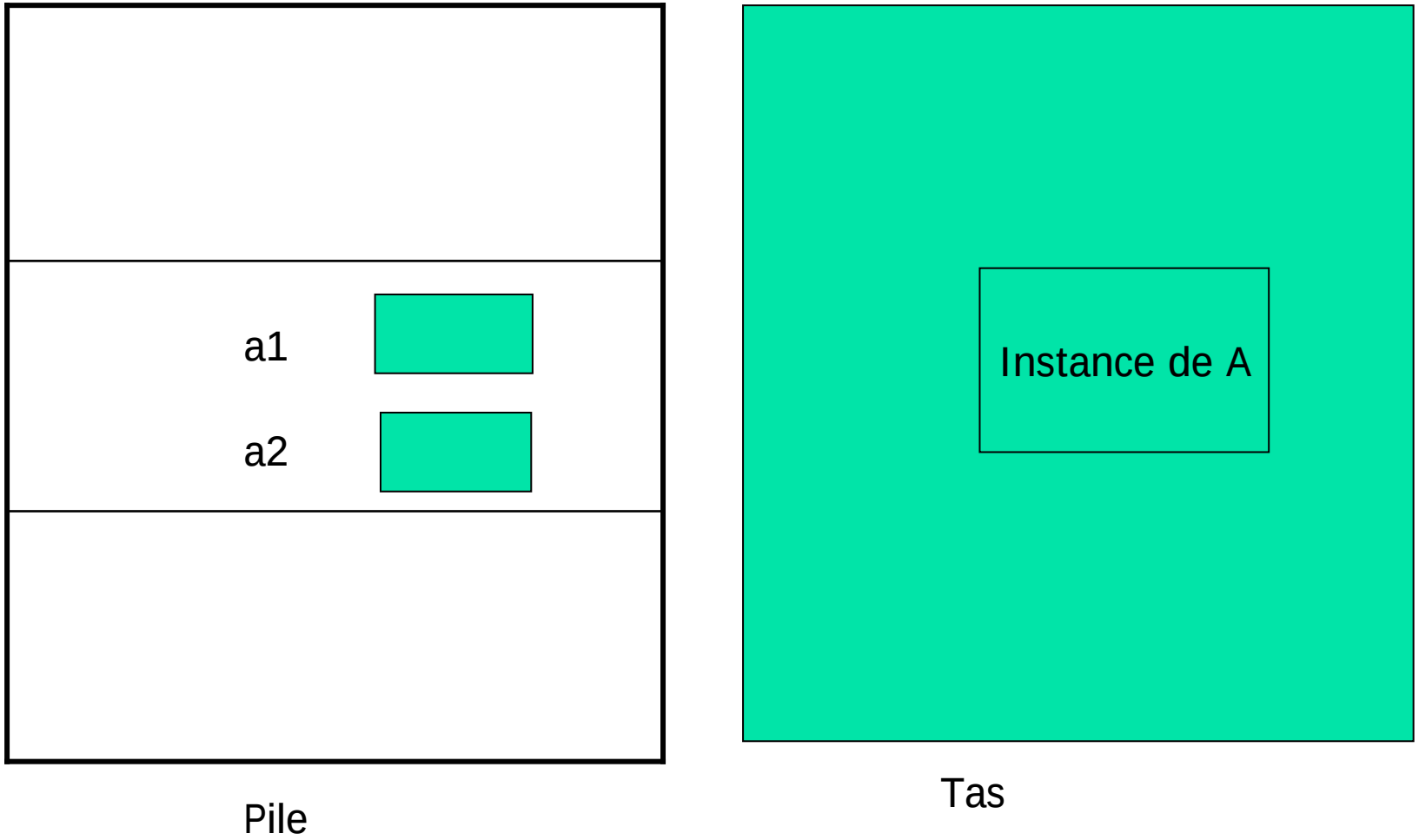
Références

```
int m() {
A a1 = new A();
A a2 = a1;
```



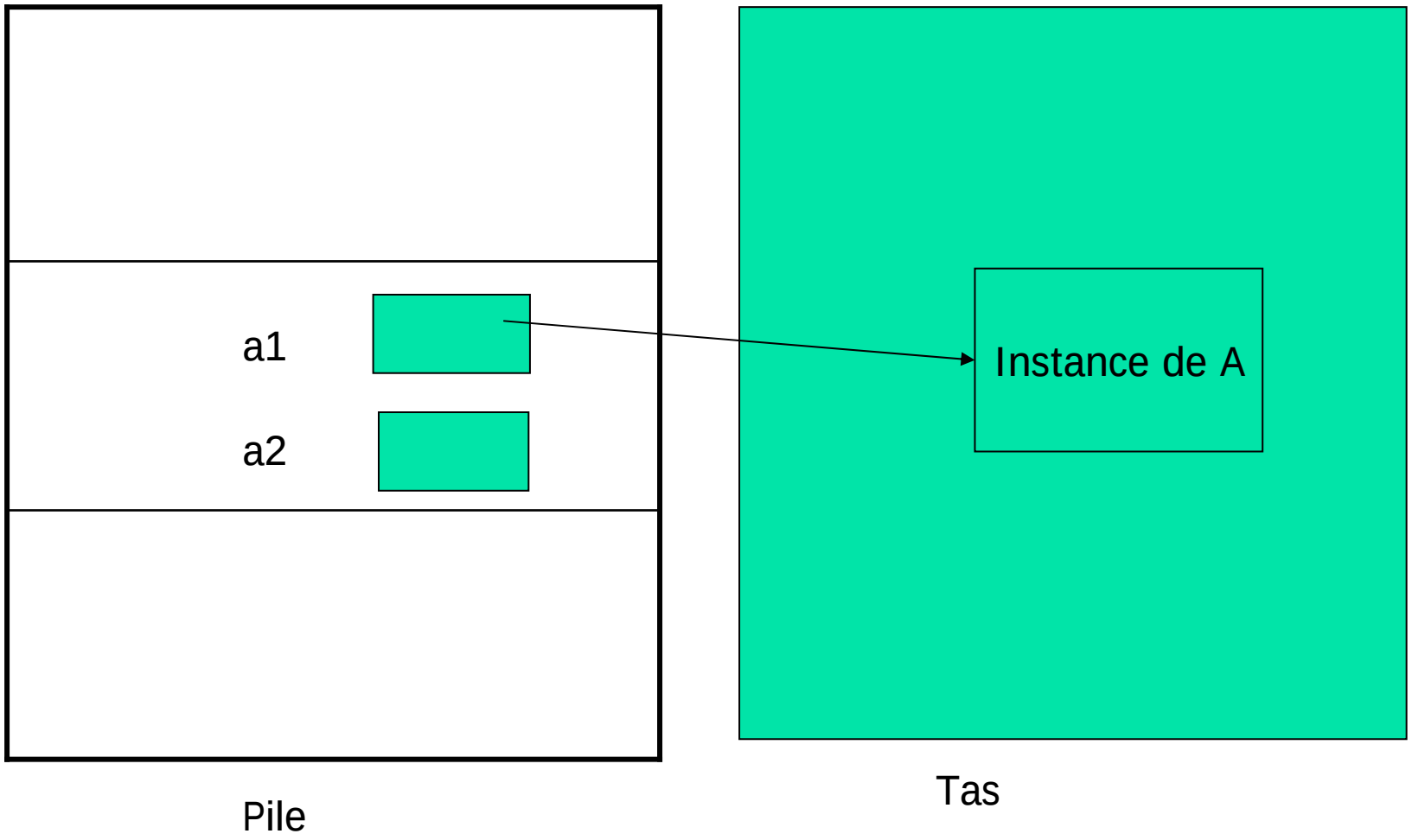
```
int m() {
A a1 = new A();
A a2 = a1;
```

Références



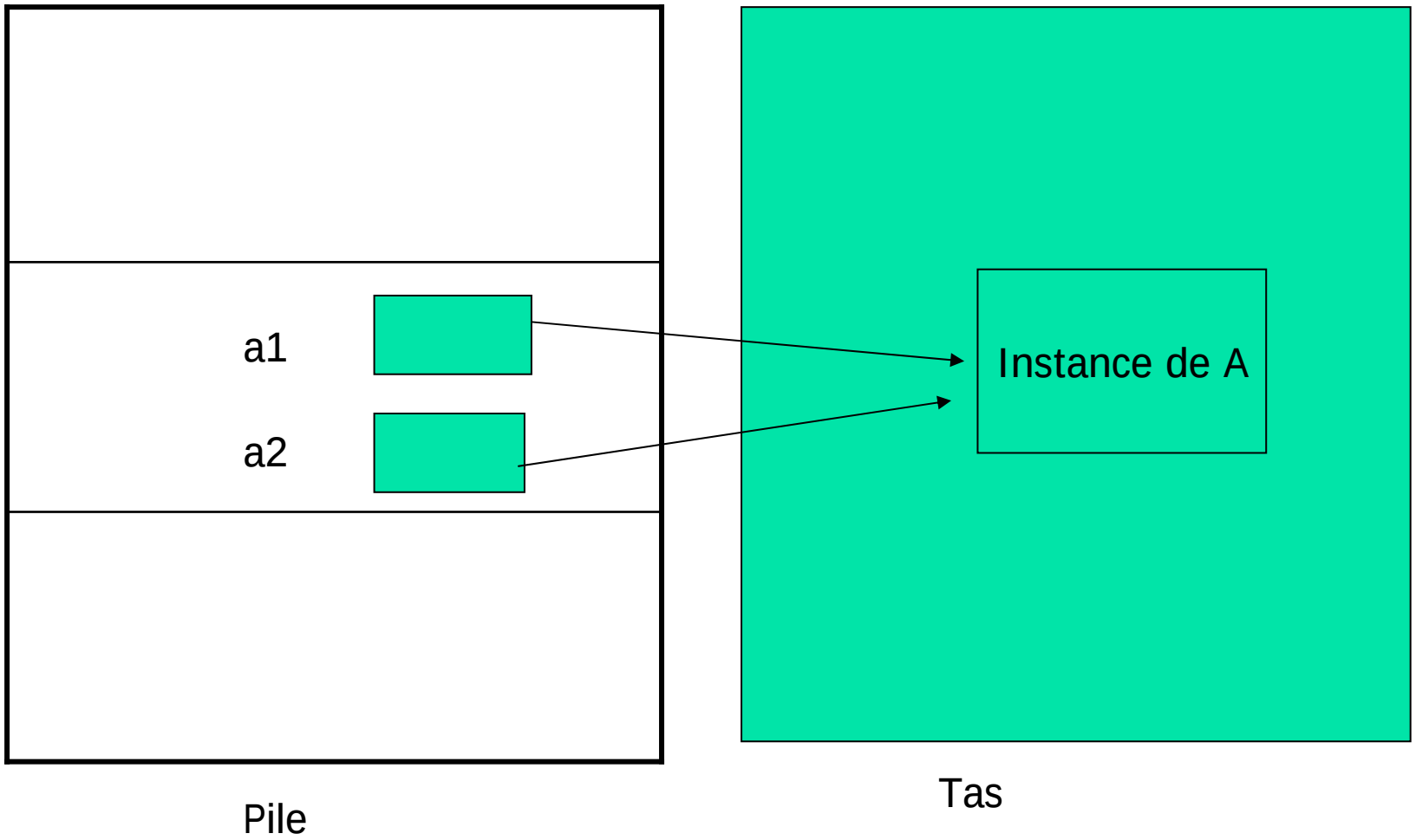
Références

```
int m() {
A a1 = new A();
A a2 = a1;
```



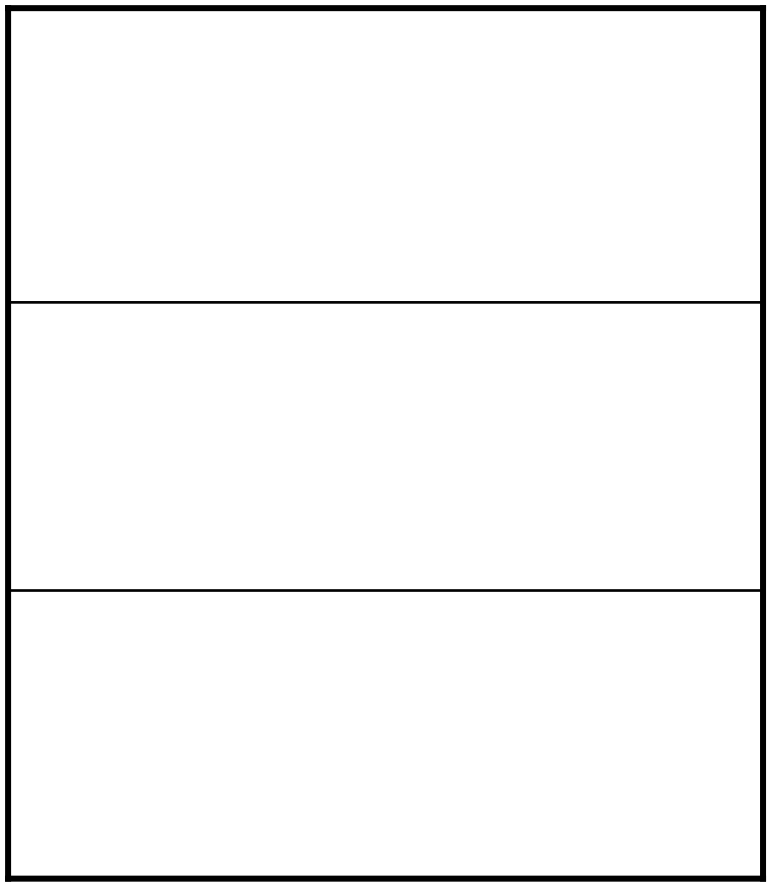
Références

```
int m() {
A a1 = new A();
A a2 = a1;
```

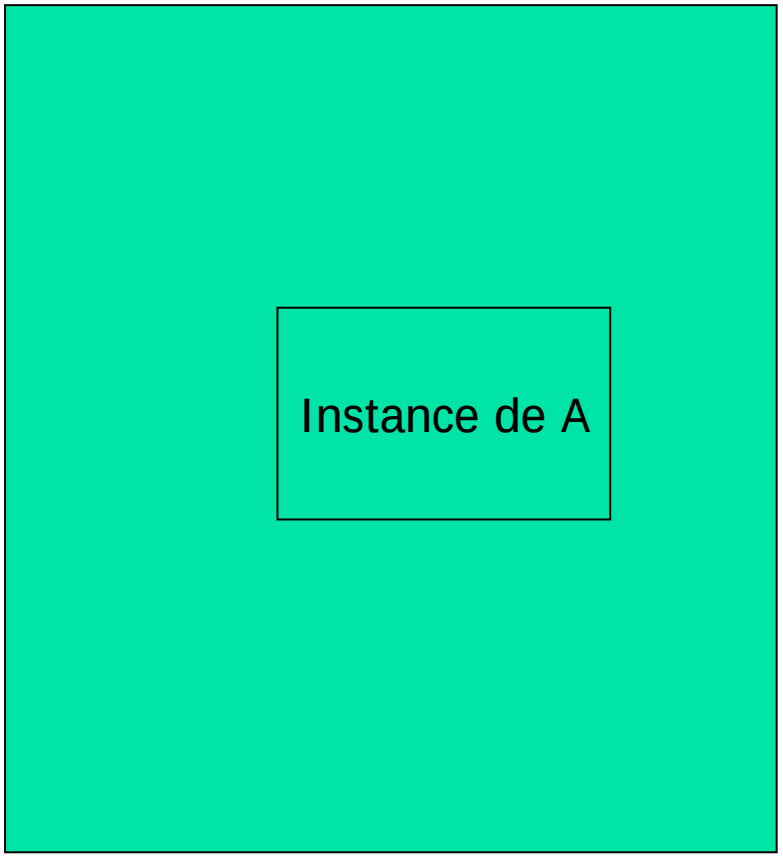


```
int m() {
A a1 = new A();
A a2 = a1;
```

Après l'exécution de la méthode m(), l'instance de **A** n'est plus référencée mais reste dans le tas

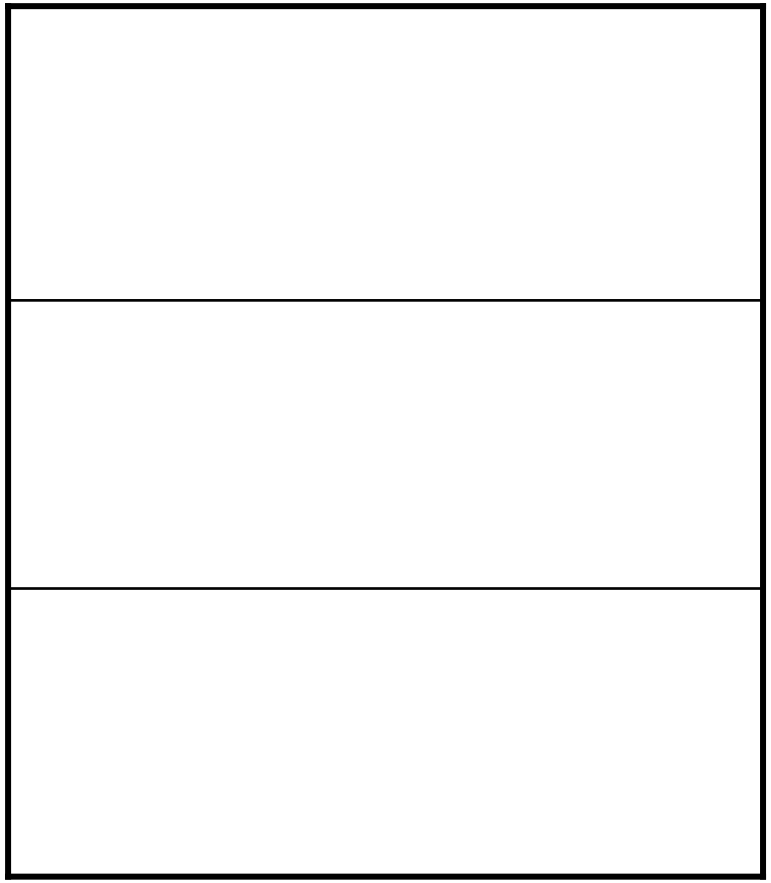


Pile

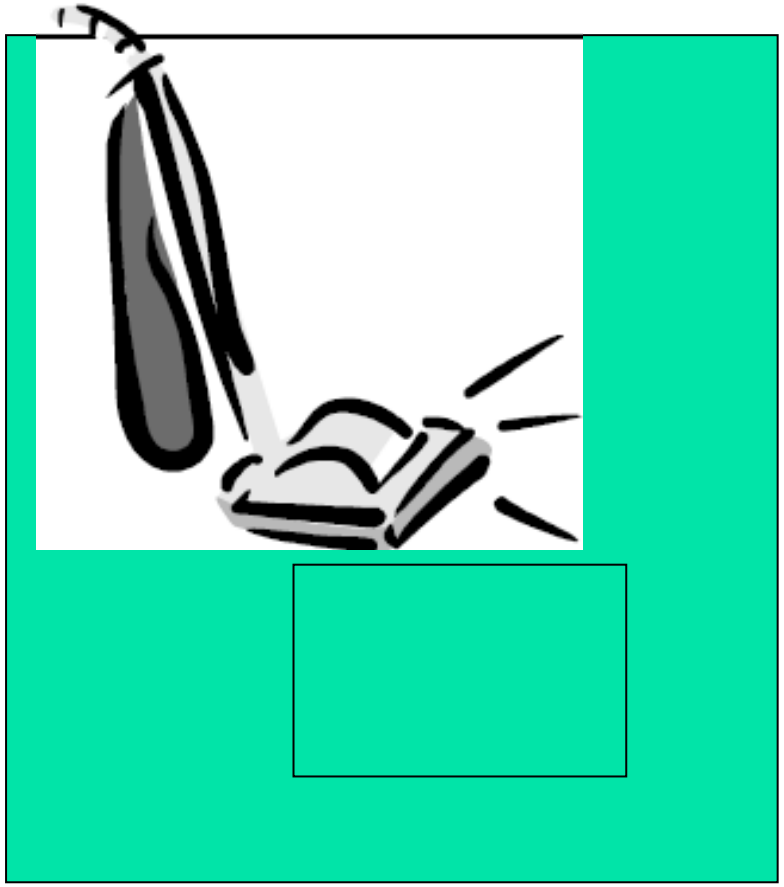


Tas

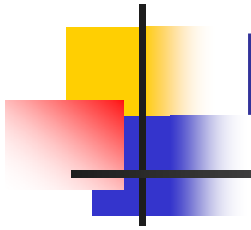
...le ramasse-miette interviendra à un moment aléatoire...



Pile

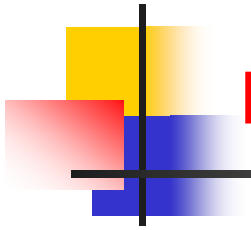


Tas



Ramasse-miettes

- Fonctionnement
  - Le ramasse-miettes (*garbage collector*) est une tâche qui
    - travaille en arrière-plan
    - libère la place occupée par les instances non référencées
    - compacte la mémoire occupée
  - Il intervient
    - quand le système a besoin de mémoire
    - ou, de temps en temps, avec une priorité faible



Types de données en Java

Les tableaux

- Préambule
  - Ils sont dérivés de la classe Object :
    - il faut utiliser des méthodes pour y accéder dont font partie des messages de la classe Object tel que equals() ou getClass()
  - Le premier élément d'un tableau possède l'indice 0

Types de données en Java

Les tableaux

- Déclaration
  - Java permet de placer les crochets après ou avant le nom du tableau dans la déclaration
  - La taille n'est pas fixée
    - int[] tabEntiers;
- Création
  - on doit donner la taille

    tabEntiers = new int[5];
    - Chaque élément du tableau reçoit la valeur par défaut du type de base du tableau
    - La taille ne pourra plus être modifiée par la suite

Types de données en Java

Les tableaux

- Exemple
  - int tableau[] = new int[50]; // déclaration et allocation

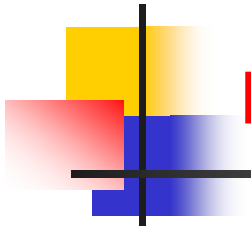
- OU
  - int[] tableau = new int[50];

- OU
  - int tab[]; // déclaration
  - tab = new int[50]; //allocation

Types de données en Java

Les tableaux

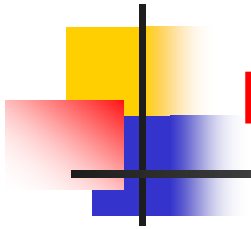
- Tableau à plusieurs dimensions
  - Java ne supporte pas directement les tableaux à plusieurs dimensions :
    - il faut déclarer un tableau de tableau
  - Exemple
    - float tableau[][] = new float[10][10];



Types de données en Java

Les tableaux

- Tableau à plusieurs dimensions
  - La taille des tableaux de la seconde dimension peut ne pas être identique pour chaque occurrence
  - Exemple
    - int dim1[][] = new int[3][];
    - dim1[0] = new int[4];
    - dim1[1] = new int[9];
    - dim1[2] = new int[2];



Types de données en Java

Les tableaux

- Initialisation
  - Chaque élément du tableau est initialisé selon son type par l'instruction new :
    - 0 pour les numériques,
    - '\0' pour les caractères,
    - false pour les booléens
    - et nil pour les chaînes de caractères et les autres objets

Types de données en Java

Les tableaux

- L'initialisation explicite d'un tableau
  - int tableau[5] = {10,20,30,40,50};
  - int tableau[3][2] = {{5,1},{6,2},{7,3}};
  - La taille du tableau n'est pas obligatoire si le tableau est initialisé à sa création
    - int tableau[] = {10,20,30,40,50};
  - Le nombre d'éléments de chaque ligne peut ne pas être identique :
    - int[][] tabEntiers = {{1,2,3,4,5,6}, {1,2,3,4}, {1,2,3,4,5,6,7,8,9}};

Types de données en Java

Les tableaux

- L'initialisation explicite d'un tableau
  - Autres exemples

```
int[] tabEntiers = {8, 2*8, 3, 5, 9};


Employe[] employes = {
new Employe("Dupond", "Sylvie"),
new Employe("Durand", "Patrick")
}
```

Types de données en Java

Les tableaux

- Parcours d'un tableau
  - for (int i = 0; i < tableau.length ; i ++) { ... }
- La variable length retourne le nombre d'éléments du tableau
- Pour passer un tableau en paramètre à une méthode, il suffit de déclarer le paramètre dans l'en tête de la méthode
- Exemple :
  - public void printArray(String texte[]){ ... }
- Les tableaux sont toujours transmis par référence puisque ce sont des objets
  - Un accès a un élément d'un tableau qui dépasse sa capacité, lève une exception du type java.lang.arrayIndexOutOfBoundsException.

- Exemple

```
public class Moyenne
{ public static void main (String args[])
  {  int i, nbEl, nbElSupMoy ;
     double somme ;
     double moyenne ;
     System.out.print ("Combien d'eleves ") ;
     nbEl = Clavier.lireInt();
     double notes[] = new double[nbEl] ;
     for (i=0 ; i<nbEl ; i++)
      { System.out.print ("donnez la note numero " + (i+1) + " : " ) ;
        notes[i] = Clavier.lireDouble() ;
      }
     for (i=0, somme=0 ; i<nbEl ; i++)  somme += notes[i] ;
     moyenne = somme / nbEl ;
     System.out.println ("\nmoyenne de la classe  " + moyenne) ;
     for (i=0, nbElSupMoy=0 ; i<nbEl ; i++ )
       if (notes[i] > moyenne) nbElSupMoy++ ;
     System.out.println (nbElSupMoy + " eleves ont plus de cette moyenne") ;
  }
}
```

Types de données en Java

Les tableaux

- Paramètres de la ligne de commande : exemple de tableau de chaînes

```
class Arguments {
   public static void main(String[] args) {
            for (int i=0; i < args.length; i++)
                     System.out.println(args[i]);
   }
}
```

java Arguments toto bibi

affichera

toto

bibi

Types de données en Java

Les tableaux

- Utilisation d'un tableau d'objets
  - Faute fréquente :
    - Utiliser les objets du tableau avant de les avoir créés

```
Employe[] personnel = new Employe[100];
personnel[0].setNom("Dupond");

Employe[] personnel = new Employe[100];
personnel[0] = new Employe();
personnel[0].setNom("Dupond");
```

Création du
Premier employé

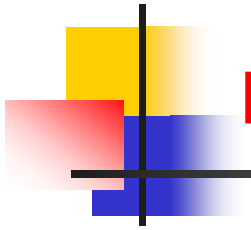
- Exemple : tableau de points

```
public class TabPoint
{ public static void main
    (String args[])
  { Point [] tp ;
    tp = new Point[3] ;
    tp[0] = new Point (1, 2)
    ;
    tp[1] = new Point (4, 5)
    ;
    tp[2] = new Point (8, 9)
    ;
    for (int i=0 ; i<tp.length
    ; i++)
      tp[i].affiche() ;
  }
}
```

```
class Point
{ public Point(int x, int y)
  { this.x = x ; this.y = y ;
  }
  public void affiche ()
  { System.out.println ("Point : " +
    x + ", " + y) ;
  }
  private int x, y ;
}
```

Types de données en Java

Les tableaux



- Comparer 2 tableaux
  - On peut comparer l'égalité de 2 tableaux (au sens où ils contiennent les mêmes valeurs) en comparant les éléments un à un
  - On peut aussi utiliser les méthodes à 2 arguments de type tableau de la classe **Arrays**

    java.util.Arrays.equals()
  - Exemple,

    java.util.Arrays.equals(double[] a, double[] a2)

Types de données en Java

Les tableaux

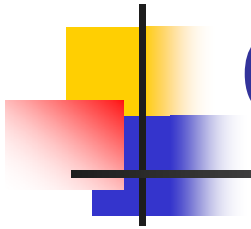
- Exemple

```
int[][] t;
t = new int[2][];
int[] t0 = {0, 1};
t[0] = t0;
t[1] = new int[] {2, 3, 4, 5};
for (int i = 0; i < t.length; i++) {
    for (int j = 0; j < t[i].length; j++) {
    System.out.print(t[i][j] + "; ");
    }
    System.out.println();
}
```

- Autre exemple : Util.java

```
class Util
{ static void raz (int t[] [])
  { int i, j ;
    for (i= 0 ; i<t.length ; i++)
      for (j=0 ; j<t[i].length ; j++)
        t[i] [j] = 0 ;
  }
  static void affiche (int t[] [])
  { int i, j ;
    for (i= 0 ; i<t.length ; i++)
      { System.out.print ("ligne de
     rang " + i + "= ") ;
        for (j=0 ; j<t[i].length ;
     j++)
          System.out.print (t[i] [j] +
     " ") ;
        System.out.println() ;
      }
  }
}
```

```
public class Tab2ind1
{ public static void main (String
    args[])
  { int t[] [] = { {1, 2, 3}, {11,
    12}, {21, 22, 23, 24} } ;
    System.out.println ("t avant raz :
    ") ;
    Util.affiche(t) ;
    Util.raz(t) ;
    System.out.println ("t apres raz :
    ") ;
    Util.affiche(t) ;
  }
}
```



Cours3-TD 3

- Exercice 1