Université Paris Cité **Programmation Orientée Objet**
L2 **Année 2024-2025**

TD - Séance n°1

Révisions – Classes

> N'oubliez pas de vous inscrire sur le Moodle du cours ! Les groupes sont créés, pensez par conséquent à vous inscrire dans *VOTRE* groupe. Si vous n'avez pas le même groupe de TD et TP, inscrivez-vous dans les deux.

Lisez attentivement le sujet, les exercices sont toujours plus faciles à faire quand on a *bien* lu l'énoncé. Parfois, il peut être utile de lire les questions suivantes : on sait où l'exercice veut en venir !

Les exercices de la partie obligatoire du TD doivent tous être traités avant la semaine prochaine. Finissez ceux que vous n'avez pas eu le temps de faire en TD chez vous. Les exercices de la seconde partie sont à faire si vous vous sentez à l'aise.

1 Exercices obligatoires

Exercice 1 *Classes et accessibilité.*
On considère les deux classes suivantes écrites dans les fichiers `Personne.java` et `Test.java`.

```java
public class Personne {
    private String nom;
    private String prenom;
    public int age;

    public Personne(String nom, String prenom, int age){
        this.nom = nom;
        this.prenom = prenom;
        this.age = age;
    }

    public void setPrenom(String p){
        this.prenom = p;
    }

    public void anniversaire(){
        this.age ++;
    }

    public String toString(){
        return "Je m'appelle : " + this.prenom
            + " " + this.nom + ". J'ai " + this.age + " ans.";
    }
}
```

```java
public class Test {

    public static void main(String[] args){
        Personne tony = new Personne("Parker","Tony",29);
        System.out.println(tony);
        Personne mickael = tony;
        mickael.setPrenom("Mickael");
        // équivalent à  System.out.println(tony.toString());
        System.out.println(tony);
        //Voir remarque
    }
}
```

Remarque Chaque objet, quelle que soit sa classe, dispose implicitement d'une méthode `toString()` qui permet de convertir une référence vers cet objet en chaîne de caractères. Cette méthode est appelée implicitement lorsqu'une référence est argument de `System.out.println(...)` : les instructions `System.out.println(o)` et `System.out.println(o.toString())` sont équivalentes.

Par défaut, cette conversion produit une chaîne formée du nom de la classe de l'objet et de son adresse (virtuelle) en mémoire. Si une classe spécifie explicitement une méthode `toString()`, qui doit dans ce cas être publique (les raisons de cette contrainte vous seront expliquées plus tard dans l'année), c'est cette nouvelle implémentation qui sera invoquée pour effectuer la conversion.

1. Qu'obtient-on dans le terminal à l'exécution de `Test` ?
2. Peut-on exécuter les lignes suivantes dans le `main` pour changer le nom d'une Personne ?

   ```
   mickael.nom = "Gelabale";
   System.out.println(mickael);
   ```

   Sinon, proposez une façon de faire sans modifier les visibilités des attributs de la classe. Étant donnée la méthode `anniversaire`, est-il utile que l'attribut `age` soit public ?
3. Si le `main` avait été écrit dans la classe `Personne` et non dans la classe `Test` aurait-on eu le droit d'écrire `mickael.nom = "Gelabale";` ?

Exercice 2 *Petites manipulations*

1. Modifiez la classe `Personne` de l'exercice précédent en ajoutant un champ représentant la quantité de monnaie en centimes d'euros que la personne possède.
2. Les deux méthodes suivantes (redondantes) doivent permettre à deux personnes de se transmettre une certaine somme. Elles retournent un booléen. Celui-ci vaut `true` si le payement s'est bien déroulé, c'est-à-dire si la totalité de la somme a bien pu été transférée.

   ```java
   public static boolean donne(Personne p1, Personne p2, int montant)
       {...}
   public boolean donne(Personne p, int montant){...}
   ```

   Écrivez ces méthodes. Donnez un exemple d'appel pour chacune.
3. Laquelle de ces deux méthodes vous semble la plus judicieuse, et pourquoi ?

Exercice 3 *Évaluation de code*

Qu'affiche le code suivant ?

```java
public class A {
  private int attr;

  public A(int value_attr) {
    this.attr = value_attr;
  }
  public boolean egal(A b) {
    return (this.attr == b.attr);
  }
  public int getAttr() {
    return this.attr;
  }
  public String toString(){
    return "attribut:"+attr+" ";
  }
  public static void main(String[] args) {
    A obj  = new A(2);
    A obj2 = obj;
    A obj3 = new A(2);

    if (obj.egal(obj2))                                              // (1)
      System.out.println("Egal");
    else System.out.println("Different");
    System.out.println((obj.egal(obj2)) ? "Egal" : "Different"); // (2)
    System.out.println((obj2.egal(obj3)) ? "Egal" : "Different");// (3)
    System.out.println((obj.egal(obj3)) ? "Egal" : "Different"); // (4)
    System.out.println((obj == obj2) ? "Egal" : "Different");    // (5)
    System.out.println((obj == obj3) ? "Egal" : "Different");    // (6)
    System.out.println((obj2 == obj3) ? "Egal" : "Different");   // (7)
    System.out.println(obj.toString());                          // (8)
    System.out.println(obj);                                     // (9)

  }
}
```

Exercice 4

1. Peut-on écrire une expression qui compare la valeur d'une variable de type `int` à la valeur `null` ?
2. Comment déclarer un tableau de taille 0 ?
3. Ajoutez un constructeur `public` sans argument dans la classe `A` qui fait initialiser `attr` à 0 (vous pouvez le faire de deux façons).
4. Déclarez une variable tableau `t` d'objets de la classe `A` précédente. Instanciez le plus simplement possible `t` par un tableau de taille 10. Qu'obtient-on si on exécute :
   `for(int i = 0; i < t.length; i++) System.out.println(t[i])`

   Que se passe-t-il si on rend explicite l'appel à `toString()` ?

Exercice 5 *Questions de cours*

Dès votre première prise en main de Java vous avez fait appel à des variables ou à des méthodes qui sont de classes ou d'instances. À la lumière des rappels de cours, précisez dans le code suivant celles qui relèvent d'une classe ou d'une instance. Puis, remplissez le code selon les commentaires.

FIGURE 1 – Un jeu de Scrabble (source : `https://fr.wikipedia.org/wiki/Scrabble`)

```java
public class Exo {
    private static int a = 1;
    private int b = 2;
    private final int c = 3;

    public static void main(String [] args){
        System.out.println("Hello");
        // Ecrivez ici, lorsque c'est possible
        // des exemples qui modifient des valeurs de a, b et c
    }
}
```

Exercice 6 *Modélisation à faire collectivement*

Voici les règles générales, incomplètes et légèrement simplifiées :
(Voir `https://fr.wikipedia.org/wiki/Scrabble` pour les règles complètes).

Grands principes. Le Scrabble (marque déposée) se joue sur un plateau de 15×15 cases. On joue avec des jetons sur lesquels sont inscrites des lettres. Au début, ces jetons sont placés dans un sac puis chaque joueur prend sept jetons et les place sur un chevalet (support qui lui permet de cacher ses lettres aux autres tout en les voyant).

Lorsqu'un joueur joue son tour, il place sur le plateau tout ou partie de ses jetons pour former ou compléter un mot verticalement ou horizontalement. Ces lettres peuvent également créer ou compléter d'autres mots dans l'autre direction. Tous les mots doivent figurer dans le dictionnaire. Le joueur pioche ensuite dans le sac pour avoir à nouveau sept jetons, sauf si le sac ne contient plus assez de lettres.

Calcul des points. Les jetons comportent non seulement des lettres, mais aussi des chiffres indiquant la valeur de chaque lettre. De plus, certaines cases sont colorées pour indiquer que la

valeur de la lettre posée dessus est doublée ou triplée, ou que la valeur du mot est doublée ou triplée. Le joueur gagne la somme des valeurs des lettres du nouveau mot ou du mot complété (y compris celles déjà posées sur la grille), en tenant compte des cases colorées.

Modélisation. Nous allons modéliser ce jeu, c'est-à-dire trouver quelles classes il faut créer avec quels attributs et quelles méthodes. Le but final de cette modélisation est un programme qui simule un plateau de jeu de Scrabble, maintient l'état du jeu (état de remplissage du plateau, score des joueurs, etc), interagit avec des joueurs humains pour exécuter leur choix de jeu sur le plateau virtuel, notifie la fin du jeu et proclame le vainqueur. Cependant il n'est pas demandé d'implémenter ce programme en détail. Il suffit d'identifier les classes nécessaires, leurs champs et les méthodes qu'elles doivent fournir (sans les implémenter).

On définira par exemple une classe `Case` et une classe `Jeton` pour commencer, quels attributs faut-il leur donner ? Quelles autres classes faut-il créer ? On rappelle qu'on peut faire des tableaux d'objets.

Écrivez les attributs et éventuellement des constructeurs de certaines classes choisis par votre enseignant ou enseignante.

2 Si vous avez du temps...

Exercice 7 *Encapsulation*
Dans cet exercice, nous allons définir un compte en banque. Un compte en banque se caractérise par un solde et par un titulaire (en l'occurrence une `personne`). On considère que les découverts ne sont pas autorisés, c'est-à-dire que le solde doit toujours être positif ou nul.

1. Écrivez une classe `Compte`, ainsi qu'un constructeur adapté.
2. Écrivez la méthode `getSolde` qui renvoie le montant présent sur un compte.
3. Écrivez la méthode `credite` qui crédite le compte d'un certain montant.
4. Écrivez la méthode `debite` qui débite le compte d'un certain montant. (Rappel : pas de découvert)
5. *** On se propose d'attribuer un numéro unique à chaque compte. Ainsi, le premier compte instancié aura pour numéro `0`, le suivant `1` et ainsi de suite. En ajoutant à la classe `Compte` un champ statique `nbComptes` et un champ d'instance `numero`, modifiez le constructeur pour qu'il garde trace du nombre de comptes instanciés jusque là et initialise l'identifiant unique `numero`.

Exercice 8 *Liens croisés*
Dans notre modélisation, un compte est lié à son titulaire, mais une personne ne l'est pas au compte qu'elle possède. Nous proposons ici une modification simple de ce modèle.

1. Modifiez la classe `Personne`, en ajoutant un champ de type `Compte[]` qui contiendra l'ensemble des comptes associés à une personne.
2. Modifiez le constructeur de `Personne`, en ajoutant un paramètre `int n`. Le constructeur se chargera de fabriquer `n` comptes différents de solde nul pour cette personne.
3. Écrivez dans votre `main` quelques manipulations de crédit sur ces différents comptes.
4. Quand on cherche à retirer une somme importante, il se peut que les fonds d'un seul compte ne suffisent pas. On peut alors vider chacun de nos comptes jusqu'à avoir atteint cette somme. Il se peut également que la somme de tous les fonds ne suffise pas. Écrivez la méthode `retrait` correspondante.