Algorithmique
&
Programmation Orientée Objet

Travaux Dirigés

L3 MIAGE

Université Grenoble Alpes
Celine.Fouard@univ-grenoble-alpes.fr

Ce cours a été rédigé par Céline Fouard.
Il est très largement inspiré du cours d'Emmanuel Promayon, PhD donné à Polytech'Grenoble
Si vous souhaitez utiliser ce document, merci de contacter Celine.Fouard@univ-grenoble-alpes.fr

Table des matières

Déroulement des TDs

Objectifs pédagogiques des Tds

L'objectif premier du cours d'Algorithmique et Programmation Orientée Objet (APO) est d'une part de consolider et d'approfondir les prérequis en algorithmique et de voir et maîtriser les concepts clés de du paradigme de programmation orientée objet.

Un objectif sous-jascent, est le développement de la capacité de réflection et d'argumentation au sujet d'un code existant. Cet objectif sera celui de chacun des TDs qui auront pour but que vous soyez non seulement capable de proposer une solution à l'exercice mais également de construire un raisonnement et des critiques constructives sur une résolution que vous n'auriez pas proposé.

L'objectif à la fin des TDs est que vous soyez capable:

- de vous approprier/reforumler le problème pour pouvoir le résoudre
- de donner des exemples pour des tests en boîte noire
- produire une solution
- critiquer de manière constructive cette solution ainsi que d'autres solutions proposées
- d'itérer: la bonne solution n'est pas frocément atteinte du premier coup... Mais la solution doit converger vers un optimal.

Échelle de correction d'un code

1. ☐ le code compile

2. ☐ le code s'exécute sans crash

3. ☐ le code renvoie le résultat attendu

4. ☐ le code est lisible et conforme aux bonnes pratiques de programmation

5. ☐ le code est testé, commenté et versionné

Travail en groupe

Les sessions de travail se dérouleront en groupes de 4 à 5 étudiants. Dans chaque groue, il faudra élire/désigner

- un·e modérateur·trice dont le rôle ne sera pas d'être le·a *chef·fe* du groupe mais de
  - S'assurer du respect du timing des différentes phases (cf section suivante) du TD
  - S'assurer que toutes les discussions du groupe sont relatives à l'exercice

  - S'assurer que la prise de parole est équilibrée au sein du groupe : il n'y a pas un·e étudiant·e qui monopolise la parole ; **chaque** membre du groupe **doit** s'exprimer, même s'il·elle ne comprend pas / n'a pas de solution.

- un·e scribe qui sera chargé d'écrire la solution validée par le groupessur papier (et éventuellement au tableau).

Ces 2 rôles **doivent** absolument changer d'une fois sur l'autre.

Déroulement du TD

Le TD alternera des phases de travail individuel et de travail collectif.

1. constitution du groupe et nomination du/de la modérateur·trice et du/de la scribe. [moins de 2min]
2. phase individuelle: lecture et appropriation du problème, recherche d'exemples de test et recherche d'une solution. [entre 10 et 15min]
3. phase de discussion interne à chaque groupe [entre 15 et 20min]:
   - on compare les exemples, on les valide ou les invalide
   - on compare (ou construit) les solutions
   - on justifie les solutions avec un diagramme APO / on les teste
   - on se met d'accord sur une solution pour le groupe
4. phase de disscussion collective [entre 15 et 20min]:
   - On partage les solutions (4 personnes au tableau, ou bien on échange les feuilles avec des solutions).
   - chacun prend 3min pour lire la/les solutions
   - chacun évalue le code selon l'échelle de correction
   - chaque groupe donne des arguments / justifie sa notation
   - élaboration d'une solution optimale collective
5. On recommence sur les questions suivantes.

Pour que ce déroulement profite à chacun et vous permette de *muscler* vos argumentations et votre compréhension en profondeur de Java, il est bien évidemment indispensable que vous ayez lu le polycopier sur les prérequis et effectué les exercices d'entrainement sur Caseine **avant** de venir en TD.

Partie A

A Bases de la Programmation en Java

TD I

TD 01 : Classes, Instances et Diagrammes APO

I.1 Une Ampoule

Nous disposons de la classe **Ampoule** suivante:

```
1  class Ampoule {
2    boolean allumée;
3    Ampoule() {
4      allumée = false;
5    }
6    boolean estAllumee() {
7      return allumée;
8    }
9    void allumerAmpoule() {
10     allumée = true;
11   }
12   void eteindreAmpoule() {
13     allumée = false;
14   }
15 }
```

Question 1.1
Ce code contient-til (si oui, lesquelles, quelles lignes):
[] des variables ? _____
[] des paramètress ? _____
[] des attributs ? _____

Question 1.2
Parmis les éléments cités dans la question précédente, y a-t-il
[]des types primitifs ?
[]des types références ?

Question 1.3
A quelle(s) ligne(s) y a-t-il une(des)
[] déclaration: _____
[] instanciation: _____
[] initialisation: _____

Question 1.4
Où se trouve
[] le(s) constructeur(s) ?
[] une(des) méthode(s) sans retour ?
[] une(des) méthode(s) avec retour ?

Question 1.5
Dessiner le diagramme UML correspondant à la classe Ampoule.

On considère à présent le programme principal suivant:

```
1  public class Main {
2    public static void main(String[] args) {
3      Ampoule a1;
4      a1 = new Ampoule();
5      Ampoule a2;
6      a2 = new Ampoule();
7      Ampoule a3;
8      a3 = new Ampoule();
9      System.out.println("a1: " + a1.estAllumee() +
10             ", a2: " + a2.estAllumee() +
11             ", a3: " + a3.estAllumee());
12
13     a3.allumerAmpoule();
14     a1 = a3;
15     System.out.println("a1: " + a1.estAllumee() +
16             ", a2: " + a2.estAllumee() +
17             ", a3: " + a3.estAllumee());
18
19     a3 = a2;
20     System.out.println("a1: " + a1.estAllumee() +
21             ", a2: " + a2.estAllumee() +
22             ", a3: " + a3.estAllumee());
23
24     a3 = a2 = a1;
25     System.out.println("a1: " + a1.estAllumee() +
26             ", a2: " + a2.estAllumee() +
27             ", a3: " + a3.estAllumee());
28
29     a2 = new Ampoule();
30     System.out.println("a1: " + a1.estAllumee() +
31             ", a2: " + a2.estAllumee() +
32             ", a3: " + a3.estAllumee());
33
34     a2 = a1;
35     System.out.println("a1: " + a1.estAllumee() +
36             ", a2: " + a2.estAllumee() +
37             ", a3: " + a3.estAllumee());
38
39     a2.eteindreAmpoule();
40     System.out.println("a1: " + a1.estAllumee() +
41             ", a2: " + a2.estAllumee() +
42             ", a3: " + a3.estAllumee());
43
44     Ampoule a4 = a2;
45     System.out.println("a4: " + a4.estAllumee());
46   }
47 }
```

Question 1.6
Ce code contient-til (si oui, lesquelles, quelles lignes):
[] des variables ? _____
[] des paramètress ? _____
[] des attributs ? _____

Question 1.7
Parmis les éléments cités dans la question précédente, y a-t-il
[]des types primitifs ?
[]des types références ?

Question 1.8
A quelle(s) ligne(s) y a-t-il une(des)
[] déclaration: _____
[] instanciation: _____
[] initialisation: _____

Question 1.9
Quelle est la portée des données a1, a2 et a3 ?

Question 1.10
Faire le diagramme APO de ce programme étape par étape.

Question 1.11
En déduire l'affichage du programme.

I.2 Passage de paramètres...

Question 2.1
Dessinez le déroulement de ces programmes

Question 2.2
En déduire l'affichage

I.2.a Pour des entiers

On considère le code ci-dessous.

```
1  public class Incrementeur {
2      void incremente(int paramètre) {
3          System.out.println("  - À l'intérieur de menteur");
4          System.out.println("  - Avant incrémentation, paramètre = " + paramètre
               );
5          paramètre = paramètre + 1;
6          System.out.println("  - Après incrémentation, paramètre = " + paramètre
               );
7      }
8
9      public static void main(String[] args) {
10         int incré = 5;
11         Incrementeur menteur = new Incrementeur();
12         System.out.println("Avant incremente, incré = " + incré);
13         menteur.incremente(incré);
14         System.out.println("Après incremente, incré = " + incré);
15     }
16 }
```

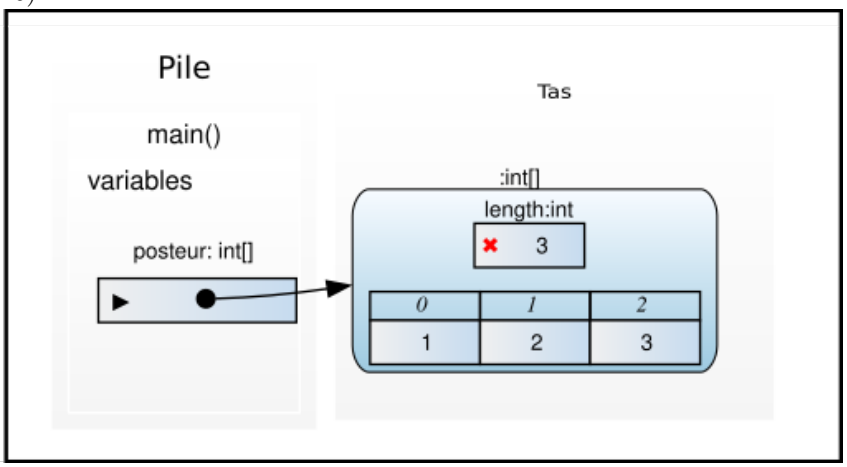
I.2.b Pour des String

```
1  public class Politicien {
2      void ticien(String paramètre) {
3          System.out.println("   - À l'intérieur de ticien");
4          System.out.println("   - avant toUpperCase : paramètre = " + paramètre)
               ;
5          paramètre = paramètre.toUpperCase();
6          System.out.println("   - après toUpperCase : paramètre = " + paramètre)
               ;
7      }
8
9      public static void main(String[] args) {
```

```
10          String str = "bonjour";
11          Politicien poli = new Politicien();
12          System.out.println("Avant poli.ticien, str = " + str);
13          poli.ticien(str);
14          System.out.println("Après poli.ticien, str = " + str);
15      }
16
17  }
```

I.2.c Pour des tableaux

Comme les tableaux n'ont pas encore été vus, on note qu'un tableau est une instance un peu particulière. Sur le diagramme APO, le tableau `int[] posteur = 1, 2, 3;` sera noté comme suit (cf cours chapitre 10):




```
1   public class Increposteur {
2       void im(int[] paramètre) {
3           System.out.println("    - À l'intérieur de im");
4           System.out.println("    - Avant incrémentation, paramètre = " + print(
                paramètre));
5           for (int p : paramètre) {
6               p = p + 1;
7           }
8           System.out.println("    - Après incrémentation, paramètre = " + print(
                paramètre));
9       }
10
11      void baratineur(int[] paramètre) {
12          System.out.println("    - À l'intérieur de baratineur");
13          System.out.println("    - Avant incrémentation, paramètre = " + print(
                paramètre));
14          int[] tmp = {2, 3, 4};
15          paramètre = tmp;
16          System.out.println("    - Après incrémentation, paramètre = " + print(
                paramètre));
17      }
18
19      void travailleur(int[] paramètre) {
20          System.out.println("    - À l'intérieur de travailleur");
21          System.out.println("    - Avant incrémentation, paramètre = " + print(
                paramètre));
22          for (int i = 0; i < paramètre.length; i++) {
23              paramètre[i] = paramètre[i] + 1;
24          }
25          System.out.println("    - Après incrémentation, paramètre = " + print(
                paramètre));
26
```

```
    }

    /* Cette méthode n'est pas à détailler dans le diagramme APO.
     * Elle écrit le tableau qui lui est entré en paramètre sous la forme [1,
         2, 3] entre crochets.*/
    String print(int[] tab){
        String s = "[";
        for (int i = 0; i < tab.length - 1; i++) {
            s += tab[i] + ", ";
        }
        if (tab.length > 0) {
            s += tab[tab.length - 1];
        }
        s += "]";
        return s;
    }

    public static void main(String[] args) {
        Increposteur incr = new Increposteur();
        int[] posteur = {1, 2, 3};
        System.out.println("Avant im, posteur = " + incr.print(posteur));
        incr.im(posteur);
        System.out.println("Après im, posteur = " + incr.print(posteur));

        System.out.println("Avant baratineur, posteur = " + incr.print(posteur)
            );
        incr.baratineur(posteur);
        System.out.println("Après im, baratineur = " + incr.print(posteur));

        System.out.println("Avant im, travailleur = " + incr.print(posteur));
        incr.travailleur(posteur);
        System.out.println("Après im, travailleur = " + incr.print(posteur));
    }
}
```

I.2.d Pour des classes

```
public class C {
    int a;

    C(int a) {
        this.a = a;
    }

    static void incremente(C paramètre) {
        paramètre.a += 1;
    }

    public static void main(String[] args) {
        C inst = new C(5);

        System.out.println("Avant incremente, inst.a = " + inst.a);
        incremente(inst);
        System.out.println("Après incremente, inst.a = " + inst.a);


    }
}
```

I.3 Algorithme Mystère

Question 3.1
Dessinez le déroulement de ces programmes

Question 3.2
En déduire l'affichage

```java
public class Mystere {
    public static void main(String[] args) {
        String mystere = "";
        for (int j = 0; j < 3; j++) {
            for (int i = 0; i < 3; i++) {
                if ((i == j) || ((i == 1) && (j == 2))) {
                    mystere += " 0 ";
                } else {
                    mystere += " X ";
                }
            }
            mystere += "\n";
        }

        System.out.println(mystere);
    }
}
```