

# DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE

**SIGLE DU COURS:** IFT 1010 (H2001)

**NOM DU PROFESSEUR:** Philippe Langlais

**TITRE DU COURS:** Programmation I

## EXAMEN FINAL

Date : jeudi 26 avril 2001

Heure : 15:30 - 18:30

Salle : B-245 Pavillon 3200 Jean-Brillant

## DIRECTIVES PÉDAGOGIQUES:

- Inscrivez tout de suite votre nom et code permanent dans l'encadré.
- Aucune documentation autorisée.
- Si vous avez besoin de brouillon, utilisez les feuilles à la fin du sujet (vous pouvez arracher ces feuilles).
- Répondez **sur le questionnaire** en utilisant l'espace libre qui suit chaque question. Si vous manquez de place, utilisez le verso en indiquant clairement que votre solution s'y trouve (une flèche ou tout autre marquage clair).
- Le barème est donné à titre indicatif et peut-être modifié lors de la correction.

## Barème indicatif:

Exercice 1:	/10			
Exercice 2:	/20	a)	b)	c)
Exercice 3:	/10	a)	b)	c)
Exercice 4:	/10			
Exercice 5:	/25			
Exercice 6:	/25			
<b>Total:</b>	<b>/100</b>			

Inscrivez votre nom et votre code permanent ici

### Exercice 1 (10 points) – Parcourir un tableau

Écrire une méthode `estTrie` qui prend la référence d'un tableau d'entiers en argument et qui retourne `true` si le tableau est trié en ordre croissant, `false` sinon.

Exemple:

```
int [] t1 = {1,2,5,7};  
int [] t2 = {7,2,4};
```

```
System.out.println(estTrie(t1)); // affichera true  
System.out.println(estTrie(t2)); // affichera false
```

**Votre réponse** (qui devrait être très courte):

## Exercice 2 (20 points) – Récursivité

Soit la méthode:

```
void methode (int n) {  
    if (n != 4) {  
        methode(n+1);  
        System.out.println(n);  
        methode(n+1);  
    }  
    else System.out.println("STOP");  
}
```

a) Qu'affiche à l'écran l'appel suivant: `methode(4)`;

b) Qu'affiche à l'écran l'appel suivant: `methode(2)`;

c) Qu'affiche à l'écran l'appel suivant: `methode(5)`;

### Exercice 3 (10 points) – Cherchez l’erreur

Dans tout cet exercice, on vous demande de retrouver des erreurs. Chaque erreur que vous devez trouver est indépendante de la suivante. En d’autres mots: si vous trouvez une erreur en ligne 3, la réponse consistant à dire: *il y a une erreur en ligne 3 donc les lignes suivantes ne sont pas compilables* est une mauvaise réponse.

Soit la classe `Fiche`:

```
class Fiche {
    private String nom;
    public int age;

    public Fiche(String n, int a) {nom = n; age = a;}
    private int getAge() {return age;}
    public String getName() {return nom;}
}
```

- a) Rayez (barrez/tracez une barre horizontale sur) les instructions qui ne sont pas compilables<sup>1</sup>. On suppose que la méthode `main` n’a pas de problème d’accès à la classe `Fiche`.

```
public static void main (String [] args) {
    String nom = "Jean";
    Fiche f1 = new Fiche(15,10);
    Fiche f2 = new Fiche("Jean",20);
    Fiche f3 = new Fiche(nom,20);
}
```

- b) Idem avec la méthode `main` suivante. On admettra ici que la première instruction est compilable.

```
public static void main (String [] args) {

    Fiche f = new Fiche("Pierre",20); // admettons que celle-ci soit compilable

    f.getName();
    f.getAge();

    f.age = 15;
    f.nom = "toto";
}
```

---

<sup>1</sup>Compiler  $\simeq$  `javac`

- c) Ce programme comporte une erreur logique rendant son exécution erronée. Décrivez clairement en quoi elle consiste.

```
public static void main (String [] args) {  
    Fiche f = null;  
    f.getName();  
}
```

#### Exercice 4 (10 points) – Booléens

Ces deux méthodes sont-elles identiques, c'est-à-dire produisent-elles le même affichage et ce quelque soient les valeurs d'appel données à a, b et c ? Justifiez **clairement** votre réponse<sup>2</sup>.

```
void m1(int a, int b, int c) {  
    if ((a == 2) && (b == 3)) {  
        if (c == 0) System.out.println("X");  
    }  
    else System.out.println("Y");  
}
```

```
void m2(int a, int b, int c) {  
    if ((a == 2) && (b == 3) && (c == 0))  
        System.out.println("X");  
    else  
        System.out.println("Y");  
}
```

**Votre réponse (qui doit commencer par l'un des mots OUI ou NON):**

---

<sup>2</sup>Si vous répondez correctement à la question mais que votre explication est fausse, partielle ou imprécise, votre réponse sera considérée comme mauvaise (aucun point).

## Exercice 5 (25 points) – POO

Dans cet exercice, vous devez écrire une classe `Polynome` qui permet de gérer des polynômes de degré 2.

$p(x)$  est un polynôme de degré 2 s'il est de la forme:  $p(x) = ax^2 + bx + c$ ; avec  $a$  un entier différent de zéro, et  $b$  et  $c$  deux entiers quelconques.

Votre classe polynôme doit contenir les éléments suivants:

- des variables de classe,
- deux constructeurs (voir l'exemple),
- une méthode `getValue(int x)` qui retourne la valeur d'un polynôme (appelant) pour une valeur de  $x$  donnée,
- une méthode `aDesRacines()` qui retourne `true` si le polynôme (appelant) a au moins une racine<sup>3</sup>, et `false` sinon,
- une méthode pour permettre l'affichage d'un polynôme sous sa forme canonique (voir l'exemple). Cette méthode devra permettre d'afficher le polynôme de manière adéquate: on n'écrit pas  $1x^2$  mais  $x^2$ ; on n'écrit pas  $3x^2+0x+2$  mais  $3x^2+2$ ; on n'écrit pas  $3x^2+-2x+2$  mais  $3x^2-2x+2$ ; etc.

La notation de cet exercice tiendra compte de la complétude de votre classe `Polynome`; c'est-à-dire, l'adéquation des modificateurs d'accès (`public/private`) aux membres de la classe, la qualité du codage de la classe, etc.

**Contraintes à respecter:** Voici un programme qui utilise la classe que vous devez écrire. Les commentaires reportés après chaque ligne illustrent ce que vous devez coder.

```
public static void main(String [] args) {

    Polynome p1 = new Polynome(2,3,1); // crée le polynôme 2x^2+3x+1
    Polynome p2 = new Polynome(); // crée le polynôme x^2+x+1
    Polynome p3 = new Polynome(1,-2,0); // crée le polynôme x^2-2x

    System.out.println("P1(3)=" + p1.getValue(3)); // affichera 28 (2x9+3x3+1)
    System.out.println("P1 a des racines ? " + p1.aDesRacines()); // affichera true
    System.out.println("P2 a des racines ? " + p2.aDesRacines()); // affichera false
    System.out.println("P3 a des racines ? " + p3.aDesRacines()); // affichera true

    System.out.println("p1(x)=" + p1); // affichera 2x^2+3x+1
    System.out.println("p2(x)=" + p2); // affichera x^2+x+1
    System.out.println("p3(x)=" + p3); // affichera x^2-2x
}
```

Reportez votre solution sur la page suivante.

---

<sup>3</sup>On rappelle qu'un polynôme de degré 2 a au moins une racine si  $\delta = b^2 - 4ac$  est supérieur ou égal à 0.



## Exercice 6 (25 points) – Algorithmique

Écrire une méthode `getNBest` qui prend en argument un tableau d'entiers  $t$  et un entier  $n$  et qui retourne une table contenant les  $n$  plus grands entiers de la table  $t$ . Ne faites aucune hypothèse particulière sur les valeurs de la table  $t$ . Si  $n$  est plus grand que le nombre d'éléments de  $t$  alors la méthode retourne un tableau de  $t.length$  éléments.

Exemple:

```
int [] t = {15, 3, 2, 5, 7, 5, 10, 15};
int [] r;

r = getNBest(t,1); // r pointe sur le tableau {15}
r = getNBest(t,3); // r pointe sur le tableau {15 15 10}
r = getNBest(t,5); // r pointe sur le tableau {15 15 10 7 5}
```

**Contraintes à respecter:** Votre méthode ne doit pas créer d'autre tableau que le tableau de  $n$  éléments retourné par la méthode et ne doit pas modifier la table  $t$ .

**Notes:** Un seul passage sur  $t$  suffit pour coder la solution (mais plusieurs sur le tableau résultat). Vous avez normalement programmé cette méthode dans le devoir 2.

**Votre réponse:**

**Brouillon 1/4**

**Brouillon 2/4**

**Brouillon 3/4**

**Brouillon 4/4**