



Résumily | Université de Boumerdes

Document

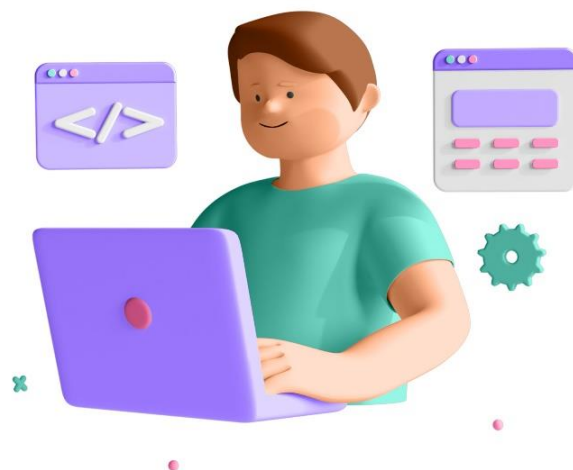
Solution Série n°1 2022/2023

Module

Programmation Orientée Objet

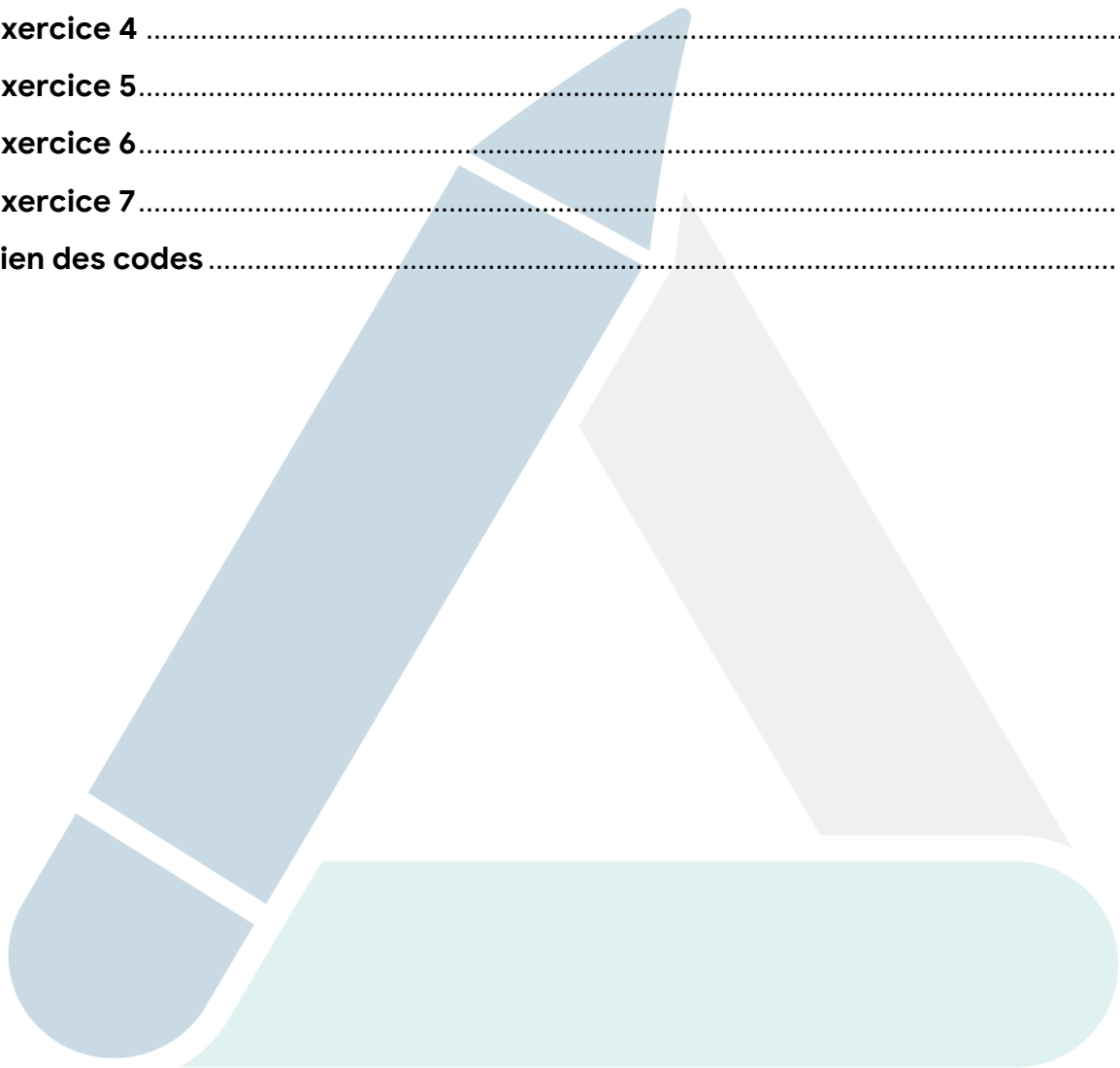
Réalisé par

Résumily



SOMMAIRE

Introduction	3
Exercice 1	4
Exercice 2	6
Exercice 3	7
Exercice 4	8
Exercice 5	10
Exercice 6	12
Exercice 7	14
Lien des codes	16



INTRODUCTION

Bienvenue dans ce document qui contient la solution détaillée de la série 1 du module de Programmation Orientée Objet de l'année universitaire 2022/2023. Ce document a été créé par l'équipe **Résumily** afin de vous aider à mieux comprendre les concepts fondamentaux de la programmation orientée objet et de vous fournir des solutions claires et détaillées pour les exercices proposés dans cette série.

Nous espérons que ce document vous sera utile dans votre apprentissage de la programmation orientée objet et nous vous souhaitons bonne lecture !

[Énoncé de la Série n°1](#)



Cliquez sur l'icône

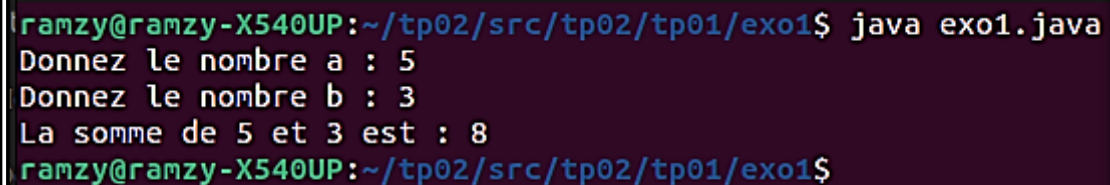
EXERCICE 1

```
import java.util.Scanner;

public class exo1 {

    public static void main(String[] args) {
        int a,b,s;
        Scanner scan = new Scanner(System.in);
        System.out.print("Donnez le nombre a : ");
        a = scan.nextInt();
        System.out.print("Donnez le nombre b : ");
        b = scan.nextInt();
        s = a+b;
        System.out.println("La somme de "+a+" et "+b+" est : "+s);
    }
}
```

Output :



```
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo1$ java exo1.java
Donnez le nombre a : 5
Donnez le nombre b : 3
La somme de 5 et 3 est : 8
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo1$
```

Ce programme permet à l'utilisateur de saisir deux nombres entiers, calcule leur somme, et affiche le résultat à l'écran.

Pour ce faire, le programme crée trois variables entières **a**, **b** et **s**, où **a** et **b** sont initialisées avec les valeurs saisies par l'utilisateur à l'aide de l'objet **Scanner**. La somme est ensuite calculée en ajoutant les valeurs de **a** et **b**, puis stockée dans **s**.

Enfin, le résultat est affiché à l'écran à l'aide de la méthode **println** de l'objet **System.out**.

Tips and tricks :

1. Utilisez `import java.util.Scanner;` pour importer la classe **Scanner** : Cela permet d'utiliser les fonctionnalités de la classe **Scanner** pour lire les entrées utilisateur à partir de la console.
2. Créez un objet **Scanner** pour lire les entrées utilisateur : Vous pouvez créer un objet **Scanner** en utilisant le constructeur `Scanner(System.in)`. Cet objet peut être utilisé pour lire les entrées utilisateur à partir de la console.
3. Utilisez `nextInt()` pour lire des entiers : Dans le code que vous avez fourni, la méthode `nextInt()` de la classe **Scanner** est utilisée pour lire les entiers entrés par l'utilisateur.



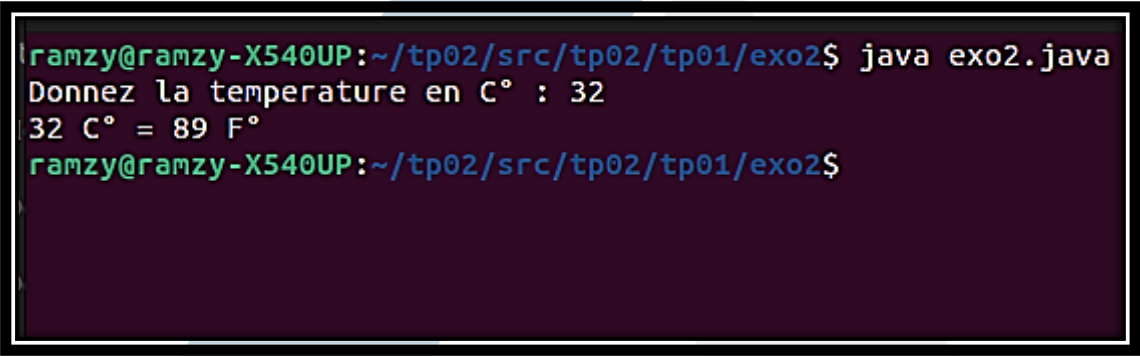
EXERCICE 2

```
import java.util.Scanner;

public class exo2 {

    public static void main(String[] args) {
        int C,F;
        Scanner scan = new Scanner(System.in);
        System.out.print("Donnez la temperature en C° : ");
        C = scan.nextInt();
        F = (C*9/5)+32;
        System.out.println(C+" C° = "+F+" F°");
    }
}
```

Output :



```
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo2$ java exo2.java
Donnez la temperature en C° : 32
32 C° = 89 F°
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo2$
```

Le programme demande à l'utilisateur de saisir une température en degrés Celsius. Ensuite, il utilise la formule de conversion pour calculer la température en degrés Fahrenheit. Enfin, il affiche la température en degrés Fahrenheit à l'écran.

Tips and tricks :

Voici 3 méthodes principales pour formater la sortie en Java :

- `System.out.print()` : affiche du texte à l'écran sans saut de ligne
- `System.out.println()` : affiche du texte à l'écran avec un saut de ligne à la fin
- `System.out.printf()` : permet d'afficher du texte formaté en utilisant des spécificateurs de format. **Exemple :**
`System.out.printf("%d C° = %d F°", C, F);`

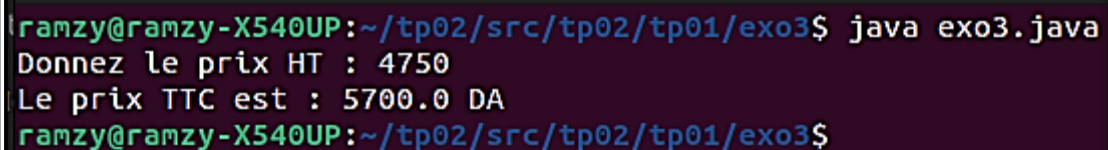
EXERCICE 3

```
import java.util.Scanner;

public class exo3 {

    public static void main(String[] args) {
        final float TVA = 0.2f;
        float HT,TTC;
        Scanner scan = new Scanner(System.in);
        System.out.print("Donnez le prix HT : ");
        HT = scan.nextFloat();
        TTC = HT + (HT*TVA);
        System.out.println("Le prix TTC est : "+TTC+" DA");
    }
}
```

Output :



```
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo3$ java exo3.java
Donnez le prix HT : 4750
Le prix TTC est : 5700.0 DA
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo3$
```

Ce programme permet de calculer le prix toutes taxes comprises (**TTC**) en fonction du prix hors taxes (**HT**) et d'un taux de **TVA** fixé à 20%. Il utilise une variable constante **TVA** pour stocker la valeur fixe du taux de **TVA** et demande à l'utilisateur d'entrer le prix **HT**. Le programme calcule ensuite le prix **TTC** en ajoutant au prix **HT** le produit du prix **HT** par le taux de **TVA**. Enfin, il affiche le résultat à l'écran avec une unité monétaire (DA).

Tips and tricks :

- Le type **float** est utilisé pour stocker des nombres à virgule flottante en simple précision.
- Pour déclarer une variable **float**, on doit ajouter le suffixe **f** (**Exemple : 10.5f**) à la fin de la valeur assignée.
- Pour saisir une valeur **float** depuis l'entrée utilisateur, on utilise la méthode `nextFloat()` de l'objet **Scanner**. Cette méthode lit la prochaine valeur entrée par l'utilisateur en tant que **float** et la retourne.
- **final** permet de déclarer une constante (**final type nomCte = valeur;**)
Exemple : final float TVA = 0.2f;

EXERCICE 4

Éliminez les parenthèses inutiles :

- $a = (b+5);$ -> parenthèses inutiles, donc : $a=b+5;$
- $a = (b=s) + 2;$
 - **Cas 1 :** en prenant en compte seulement **a**
 - ✓ On n'aura pas besoin des parenthèses car peu importe **b** et **s** le résultat de **a** sera le même avec ou sans parenthèses
-> $a= b=s + 2;$
 - **Cas 2 :** en prenant en compte **b** et **s** :
 - ✓ Les parenthèses seront utiles car la variable **b** changera de valeur

Exemple :

Variables	Valeur initiale	Valeur finale	
		Avec les ()	Sans les ()
a	/	3	3
b	2	1	3
s	1	1	1

- **boolean** $n = (a==b);$ -> parenthèses inutiles donc : **boolean** $n = a==b;$
- $n = (a<b) \&\& (b<s);$ -> parenthèses inutiles donc : $a = a<b \&\& b<s;$
- $a = (a++)*(b+s);$ (Les parenthèses autour de **b+s** sont nécessaires car l'opérateur « + » a une priorité inférieure à celle de la multiplication)

Tips and tricks :

- Il est important de comprendre la priorité des opérateurs et de bien placer les parenthèses pour éviter des erreurs de calcul et rendre le code plus lisible et facile à comprendre.

Exemple : $a = (a++)*(b+s);$ Dans cette expression, les parenthèses sont nécessaires pour que l'opérateur **++** soit évalué avant la multiplication. Sans les parenthèses, l'opération serait évaluée comme $(a * b) + s$, ce qui n'est pas l'intention du code. En ajoutant les parenthèses, l'opération est évaluée comme $(a++) * (b+s)$, ce qui est le résultat attendu.

- En Java, les opérateurs sont évalués en fonction de leur ordre de priorité. Voici l'ordre de priorité des opérateurs Java, du plus élevé au moins élevé :

1. **Opérateurs unaires :** ++, --, +(signe), -(signe), !(NON logique), ~(NON binaire)
2. **Opérateurs arithmétiques :** *, /, % (modulo), +, -

3. **Opérateurs de décalage** : << (décalage à gauche), >> (décalage à droite), >>> (décalage à droite sans signe)
4. **Opérateurs de comparaison**: <, <=, >, >=, instanceof
5. **Opérateurs d'égalité**: ==, !=
6. **Opérateurs logiques** : &(ET binaire), ^(XOR « OU exclusif » binaire), |(OU inclusif binaire)
7. **Opérateurs conditionnels** : &&(ET logique), ||(OU logique)
8. **Opérateurs de l'affectation** : =, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=

Notez que l'ordre des opérations peut être modifié en utilisant des parenthèses pour grouper les opérations à effectuer en premier.

Si les opérateurs ont la même priorité, la priorité sera de gauche à droite.

EXERCICE 5

```
import java.util.Scanner;

public class exo5 {

    public static void main(String[] args) {
        int i, j, n ;
        i = 0 ; n = i++ ;
        System.out.println("A : i=" + i + " n=" + n ) ;
        i = 10 ; n = ++i ;
        System.out.println("B : i=" + i + " n=" + n ) ;
        i = 20 ; j = 5 ; n = i++ * ++j ;
        System.out.println("C : i=" + i + " j=" + j + " n=" + n ) ;
        i = 15 ; n = i += 3 ;
        System.out.println("D : i=" + i + " n=" + n ) ;
        i = 3 ; j = 5 ; n = i *= --j ;
        System.out.println("E : i=" + i + " j=" + j + " n=" + n ) ;
    }
}
```

- Dans le premier bloc, la valeur de **i** est initialisée à 0 puis est assignée à **n** qui prend donc la valeur 0. Après cela, la valeur de **i** est incrémentée de 1. Donc, à la fin de ce bloc, **i=1** et **n=0**.
- Dans le deuxième bloc, la valeur de **i** est d'abord incrémentée de 1 (donc **i=11**). Ensuite, la nouvelle valeur de **i** est assignée à **n** qui prend donc la valeur de 11. Donc, à la fin de ce bloc, **i=11** et **n=11**.
- Dans le troisième bloc, la valeur de **i** est initialisée à 20 et la valeur de **j** à 5. Ensuite, **j** est incrémentée de 1 (donc **j=6**) et la multiplication de **i** (20) et de **j** (6) est assignée à **n**, donc **n=120**. Après cela, la valeur de **i** est incrémentée de 1, donc **i=21**. Donc, à la fin de ce bloc, **i=21**, **j=6** et **n=120**.
- Dans le quatrième bloc, la valeur de **i** est initialisée à 15. Ensuite, la valeur 3 est ajoutée à **i** (donc **i=18**) et la nouvelle valeur de **i** est assignée à **n** qui prend donc la valeur de 18. Donc, à la fin de ce bloc, **i=18** et **n=18**.
- Dans le cinquième bloc, la valeur de **i** est initialisée à 3 et la valeur de **j** à 5. Ensuite, **j** est décrémentée de 1 (donc **j=4**) et la multiplication de **i** (3) et **j** (4) est assignée à **n**, donc **n=12**. Après cela, la valeur de **i** est assignée à la nouvelle valeur de **i** multipliée par **j**, donc **i=12**. Donc, à la fin de ce bloc, **i=12**, **j=4** et **n=12**.

Lorsque le code est compilé et exécuté, il affiche :

```
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo5$ java exo5.java
A : i=1 n=0
B : i=11 n=11
C : i=21 j=6 n=120
D : i=18 n=18
E : i=12 j=4 n=12
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo5$
```

Tips and tricks :

- L'opérateur **++** ajoute 1 à une variable, tandis que l'opérateur **--** soustrait 1. L'emplacement de l'opérateur **++** ou **--** avant ou après la variable affecte le moment où l'incrément ou la décrémentation est effectuée.
- **Post-incrémentation** : Si l'opérateur **++** ou **--** est après la variable, la variable sera d'abord utilisée dans l'expression, puis incrémentée ou décrémentée.
- **Pré-incrémentation** : Si l'opérateur **++** ou **--** est avant la variable, la variable sera d'abord incrémentée ou décrémentée, puis utilisée dans l'expression.
- Les opérateurs raccourcis (**+=**, **-=**, ***=**, **/=**) permettent d'effectuer une opération arithmétique et d'affecter le résultat à la variable. Par exemple, **i += 3** est équivalent à **i = i + 3**.

Déroulement : (important)

- La priorité des expressions dans Java est évaluée de **gauche à droite** (Dans les autres langages comme « C » la priorité est évaluée de manière non spécifiée), **par exemple** :

Soit l'expression : `(int a = 5 ; int b= ++a + a++ + a++ + a;)`

Dans Java : le résultat sera **27**

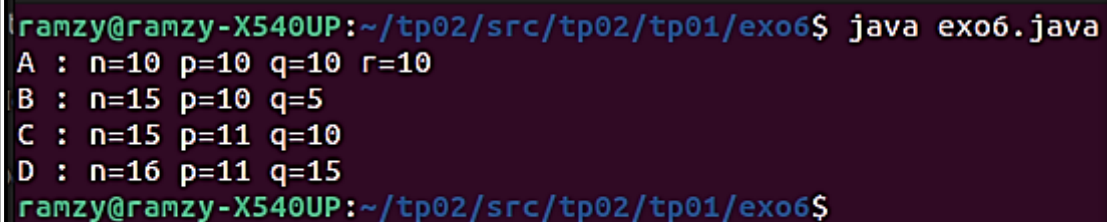
Expression	Valeur en mémoire
<code>int a=5 ;</code>	a=5
<code>int b= ++a</code> (« a » s'incrémente d'abord)	b=6 ; a=6
<code>b= 6 + a++ = 6 + 6</code> (« a » s'incrémente après le calcul)	b=12 ; a=7
<code>b= 12 + a++ = 12 + 7</code>	b=19 ; a=8
<code>b= 19 + a = 19 + 8</code>	b=27 ; a=8

Dans C : le résultat sera 28 (évaluation non spécifiée)

EXERCICE 6

```
public class exo6 {  
  
    public static void main(String[] args) {  
        int n=10, p=5, q=10, r;  
        r = (int) (p = q) ;  
        System.out.println("A : n=" + n + " p=" + p + " q=" + q + " r=" + r ) ;  
        n = p = q = 5 ;  
        n += p += q ;  
        System.out.println( "B : n=" + n + " p=" + p + " q=" + q );  
        q = n < p ? n++ : p++ ;  
        System.out.println("C : n=" + n + " p=" + p + " q=" + q ) ;  
        q = n > p ? n++ : p++ ;  
        System.out.println( "D : n="+ n + " p=" + p + " q=" + q ) ;  
    }  
}
```

Output :



```
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo6$ java exo6.java  
A : n=10 p=10 q=10 r=10  
B : n=15 p=10 q=5  
C : n=15 p=11 q=10  
D : n=16 p=11 q=15  
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo6$
```

Les variables **n**, **p** et **q** sont initialisées respectivement à 10, 5 et 10.

A : La variable **r** est initialisée à la valeur de **p** après que **p** a été affecté à la valeur de **q** grâce à l'opérateur d'assignation **=**. Le résultat de l'opération est ensuite converti en un entier à l'aide de la notation (**int**). Ainsi, **r** a la valeur **10**, tout comme **q** et **p**.

B : Les variables **n**, **p** et **q** sont toutes initialisées à 5 à l'aide de l'opérateur d'assignation **=**. Ensuite, la valeur de **q** est ajoutée à **p**, puis le résultat est ajouté à **n** grâce à l'opérateur d'addition composée **+=**. Ainsi, **n** a la valeur 15, tandis que **p** prend la valeur 10, et **q** prend la valeur 5.

C : La valeur de **q** est affectée à **n++** si **n** est inférieur à **p**, sinon la valeur de **p++** est affectée à **q**. Dans ce cas, **n** est supérieur à **p**, donc la valeur de **p++** (qui est 10) est affectée à **q**. La valeur de **n** et de **p** est augmentée de 1. Ainsi, **n** a la valeur 15, **p** a la valeur 11 et **q** a la valeur 10.

D : La valeur de **q** est affectée à **n++** si **n** est supérieur à **p**, sinon la valeur de **p++** est affectée à **q**. Dans ce cas, **n** est supérieur à **p**, donc la valeur de **n++** (qui est 16) est affectée à **q**. La valeur de **n** est augmentée de 1. Ainsi, **n** a la valeur 16, **p** a la valeur 11 et **q** a la valeur 15.

Tips and tricks :

- L'utilisation du ternaire peut être utile pour simplifier les instructions **if-else**. Cela peut rendre le code plus concis et plus facile à lire. Par exemple, la ligne `q = n < p ? n++ : p++` (signification : si **n** est inférieure à **p** alors **n++** sinon **p++**)
Syntaxe : **condition ? expression1 (si vrai) : expression2 (si faux)**



EXERCICE 7

```
public class exo7 {

    public static void main(String[] args) {
        int n, p, q ; boolean k;
        n = 5 ; p = 2 ; /* cas 1 */
        k = n++ > p || p++ != 3 ;
        System.out.println( "A : n = " + n + " p = " + p + " k = " + k ) ;
        n = 5 ; p = 2 ; /* cas 2 */
        k = n++ < p || p++ != 3 ;
        System.out.println( "B : n = " + n + " p = " + p + " k = " + k ) ;
        n = 5 ; p = 2 ; /* cas 3 */
        k = ++n == 3 && ++p == 3 ;
        System.out.println( "C : n = " + n + " p = " + p + " k = " + k ) ;
        n = 5 ; p = 2 ; /* cas 4 */
        k = ++n == 6 && ++p == 3 ;
        System.out.println( "D : n = " + n + " p = " + p + " k = " + k ) ;
    }
}
```

Output :

```
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo7$ java exo7.java
A : n = 6 p = 2 k = true
B : n = 6 p = 3 k = true
C : n = 6 p = 2 k = false
D : n = 6 p = 3 k = true
ramzy@ramzy-X540UP:~/tp02/src/tp02/tp01/exo7$
```

Le code utilise les opérateurs logiques || et && pour effectuer des tests de condition sur les variables n, p et k. Voici les résultats attendus pour chaque cas :

Cas 1 : Dans ce cas, la première condition est vraie ($5 > 2$) donc le programme n'a pas besoin de vérifier la seconde condition ($p++ \neq 3$), qui ne sera donc pas exécutée. La variable n est ensuite incrémentée ($n = 6$) et la variable k est vraie car la première condition est vraie. La sortie sera : "A : n = 6 p = 2 k = true".

Cas 2 : La première condition est fausse (5 n'est pas inférieur à 2), donc le programme doit vérifier la seconde condition ($p++ \neq 3$). Dans ce cas, p sera incrémenté ($p = 3$) et la seconde condition est vraie car p est différent de 3. La variable k sera donc vraie. La sortie sera : "B : n = 6 p = 3 k = true".

Cas 3 : Dans ce cas, la première condition est fausse (n est égal à 6 après avoir été préalablement incrémenté), donc le programme n'a pas besoin de vérifier la

seconde condition ($p++ == 3$), qui ne sera donc pas exécutée. La variable k sera fausse car la première condition est fausse. La sortie sera : "C : $n = 6$ $p = 2$ $k = false$ ".

Cas 4 : Dans ce cas, les deux conditions sont vraies, donc les variables n et p seront incrémentées ($n = 6$ et $p = 3$). La variable k sera vraie car les deux conditions sont vraies. La sortie sera : "D : $n = 6$ $p = 3$ $k = true$ ".

Tips and tricks :

- Les expressions booléennes peuvent être simplifiées en utilisant les opérateurs logiques **&&** (« **et** » **logique**) et **||** (« **ou** » **logique**). Par exemple, si vous avez une instruction **if** qui vérifie deux conditions, vous pouvez utiliser l'opérateur **&&** pour les combiner en une seule expression. Cela peut rendre le code plus facile à lire et à comprendre.
- Il est important de bien indenter le code à l'intérieur des blocs **if-else**. Cela aide à rendre le code plus lisible et facilite la compréhension de la logique du programme.
- Il est également important d'éviter les conditions inutiles. Si vous utilisez plusieurs **if-else** pour vérifier la même condition, vous pouvez réduire le nombre de conditions en utilisant un seul **if** avec plusieurs instructions **else if**. Cela peut rendre le code plus efficace et plus facile à lire.

LIEN DES CODES

Vous trouverez tous les codes (.java) dans notre **Github**

github.com/Resumily/poo-tp1



Cliquez sur l'icône

Bonne révision !

-L'équipe de Résumily-



Liens / Résumily

Cliquez sur les icônes