3GIG - Informatique
Exercices sur les boucles

La plupart des exercices de ce recueil proviennent des sources suivantes :

- exercices 2B du LMRL ;
- cours Delphi de Fred Faber du LAM ;
- exercices supplémentaires pour la 3GIG de Pierre Dziadek et Jean-Marie Bourone du ALR ;

certains sont proposés par moi-même.

Georges Kugener, février 2021

Table des matières

Exercices à réaliser sous Unimozer

Pour chacun des exercices suivants, créez un projet Unimozer avec une classe **`Test`** comportant une méthode statique **`main`**. Cette méthode contiendra le code source du programme à réaliser.

Appeler les projets Unimozer **EX-LOOP-01**, **EX-LOOP-02**, …

EX-LOOP-1 Bonjour 20 fois

Écrivez un programme qui affiche 20 fois « Bonjour » dans la console.

```
Bonjour
Bonjour
Bonjour
Bonjour
:
```

EX-LOOP-2 Bonjour n fois

Écrivez un programme qui affiche **`n`** fois « Bonjour » dans la console. Le nombre **`n`** est introduit par l'utilisateur au clavier.

```
Entrez n: 4
Bonjour
Bonjour
Bonjour
Bonjour
```

EX-LOOP-3 Nombres entiers de 1 à 20

Écrivez un programme qui affiche dans la console les nombres entiers de 1 à 20. Le nombre n est introduit par l'utilisateur au clavier.

```
1
2
3
:
18
19
20
```

EX-LOOP-4 Nombres entiers de 1 à n

Écrivez un programme qui affiche dans la console les nombres entiers de 1 à **`n`**. Le nombre **`n`** est introduit par l'utilisateur au clavier.

```
Entrez n : 5
1
2
3
4
5
```

EX-LOOP-5 Nombres entiers de min à max

Écrivez un programme qui affiche dans la console les nombres entiers de **`min`** à **`max`**. Les nombres **`min`** et **`max`** sont introduits par l'utilisateur au clavier. On suppose que le nombre **`min`** est inférieur au nombre **`max`**.

```
Entrez min : -5
Entrez max : 3
-5
-4
-3
-2
-1
0
1
2
3
```

EX-LOOP-6 Nombres entiers de n à 1

Écrivez un programme qui affiche dans la console les nombres entiers de **n** à 1 par ordre décroissant. Le nombre **n** est introduit par l'utilisateur au clavier.

```
Entrez n : 5
5
4
3
2
1
```

EX-LOOP-7 Nombres entiers de max à min

Écrivez un programme qui affiche dans la console les nombres entiers de **max** à **min** par ordre décroissant. Les nombres **min** et **max** sont introduits par l'utilisateur au clavier. On suppose que le nombre **min** est inférieur au nombre **max**.

```
Entrez min : -3
Entrez max : 2
2
1
0
-1
-2
-3
```

EX-LOOP-8 Nombres entiers de n1 à n2

Écrivez un programme qui affiche dans la console les nombres entiers de **n1** à **n2**. Les nombres **n1** et **n2** sont introduits par l'utilisateur au clavier.

- Si n1 <= n2 alors les nombres sont affichés par ordre croissant
- Si n1 > n2 alors les nombres sont affichés par ordre décroissant

```
Entrez n1 : 2
Entrez n2 : 6
2
3
4
5
6
```

```
Entrez n1 : 12
Entrez n2 : 8
12
11
10
9
8
```

EX-LOOP-9 Numéroter Bonjour 1 à n

Écrivez un programme qui affiche **n** fois « Bonjour » dans la console. Le texte « Bonjour » est numéroté. Le nombre **n** est introduit par l'utilisateur au clavier.

```
Entrez n : 5
Bonjour 1
Bonjour 2
Bonjour 3
Bonjour 4
Bonjour 5
```

EX-LOOP-10 Table de multiplication

Écrivez un programme qui affiche les **n** premières lignes de la table de multiplication du nombre **x**. Les nombre **n** et **x** sont introduits par l'utilisateur au clavier.

```
Entrez n : 5
Entrez x : 7
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
```

EX-LOOP-11 Nombres pairs

Écrivez un programme qui affiche les nombre pairs entre 0 et **n**. Le nombre **n** est introduit par l'utilisateur au clavier.

```
Entrez n : 10
0
2
4
6
8
10
```

EX-LOOP-12 Nombres divisibles par 11

Écrivez un programme qui affiche les nombre divisibles par 11 entre 1 et **n**. Le nombre **n** est introduit par l'utilisateur au clavier.

```
Entrez n : 50
11
22
33
44
```

EX-LOOP-13 Nombres divisibles par 11 et 19

Écrivez un programme qui affiche les nombre divisibles par 11 et 19 entre 1 et **n**. Le nombre **n** est introduit par l'utilisateur au clavier.

```
Entrez n : 1000
209
418
627
836
```

EX-LOOP-14 Somme des nombres entiers de 1 à n

Écrivez un programme qui calcule la somme des nombres entiers de 1 à **n**. Le nombre **n** est introduit par l'utilisateur au clavier. Le programme affiche également les résultats intermédiaires.

Indication : Utilisez une variable pour sauvegarder la somme.

```
Entrez n : 8
1: somme = 1
2: somme = 3
3: somme = 6
4: somme = 10
5: somme = 15
6: somme = 21
7: somme = 28
8: somme = 36
La somme de 1 à 8 vaut 36
```

EX-LOOP-15 Somme des carrés

Écrivez un programme qui calcule la somme des carrés des nombres entiers de 1 à **n**. Le nombre **n** est introduit par l'utilisateur au clavier. Le programme affiche également les résultats intermédiaires.

Une factorielle par ligne

```
Entrez n : 6
1^2 = 1 : somme = 1
2^2 = 4 : somme = 5
3^2 = 9 : somme = 14
4^2 = 16 : somme = 30
5^2 = 25 : somme = 55
6^2 = 36 : somme = 91
La somme des carrés de 1 à 6 vaut 91
```

EX-LOOP-16 Factorielle

La factorielle d'un nombre n notée n ! est définie comme suit :

0 ! = 1

1 ! = 1

2 ! = 1 x 2 = 2

3 ! = 1 x 2 x 3 = 6

4 ! = 1 x 2 x 3 x 4 = 24

:

n ! = 1 x 2 x … x (n-1) x n

Écrivez un programme qui calcule et affiche les factorielles des nombres de 0 à **n**. Le nombre **n** est introduit par l'utilisateur au clavier.

```
Entrez n : 8
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
```

EX-LOOP-17 Diviseurs / nombre premier

Écrivez un programme affiche les diviseurs d'un nombre **n**. Le nombre **n** est introduit par l'utilisateur au clavier. Affichez à la fin le nombre de diviseurs et concluez si **n** est un nombre premier ou non (un nombre premier admet exactement 2 diviseurs).

```
Entrez n : 6
1 est diviseur de 6
2 est diviseur de 6
3 est diviseur de 6
6 est diviseur de 6
6 admet 4 diviseurs
6 n'est pas un nombre premier
```

```
Entrez n : 7
1 est diviseur de 7
7 est diviseur de 7
7 admet 2 diviseurs
7 est un nombre premier
```

EX-LOOP-18 n astérisques dans une ligne

Écrivez un programme qui affiche **n** astérisques dans une ligne. Le nombre **n** est introduit par l'utilisateur au clavier.

Essayez les méthodes suivantes :

- Utilisez la méthode System.out.print() qui n'effectue pas de saut de ligne après l'affichage
- Construisez d'abord dans une variable une chaîne de caractères composée de n astérisques et l'affichez ensuite.

```
Entrez n : 20
********************
```

EX-LOOP-19 Lancements d'un dé

Écrivez un programme qui simule **n** lancements de dé (Würfel) et qui calcule le total des points obtenus. Le nombre **n** est introduit par l'utilisateur au clavier.

```
Entrez n : 5
4 point(s)
1 point(s)
6 point(s)
4 point(s)
5 point(s)
Total = 20
```

EX-LOOP-20 Lancements d'un dé (* pour avancés)

Modifiez le programme précédent de manière à afficher le nombre de lancements consécutifs où le même nombre de points a été obtenu.

```
Entrez n : 10
1 point(s) 1 x
2 point(s) 2 x
5 point(s) 1 x
3 point(s) 1 x
2 point(s) 3 x
6 point(s) 1 x
3 point(s) 1 x
Total = 28
```

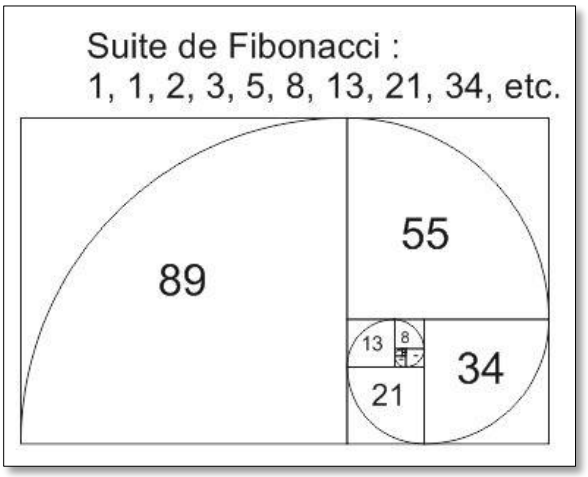
EX-LOOP-21 Suite de Fibonacci (* pour avancés)

Écrivez un programme qui affiche les **n** premiers termes de la suite de Fibonacci. Le nombre **n** est introduit par l'utilisateur au clavier.

En mathématiques, la **suite de Fibonacci** est une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent. Elle commence par les termes 0 et 1 si on part de l'indice 0.

Notée F_n, le nième terme est donc définie par :

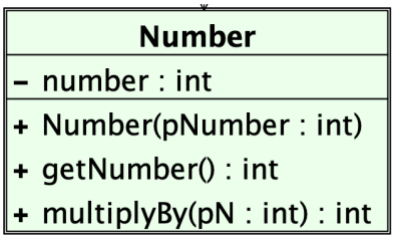
- $F_0 = 0$
- $F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$ pour $n \geqslant 2$




```
Entrez n : 11
F0 = 0
F1 = 1
F2 = 1
F3 = 2
F4 = 3
F5 = 5
F6 = 8
F7 = 13
F8 = 21
F9 = 34
F10 = 55
F11 = 89
```

EX-LOOP-22 Multiplication par addition

Créez la classe **`Number`**.



La méthode **`multiplyBy`** effectue la multiplication **`number`** x **`pN`** par additions successives et retourne le résultat.

`number` x **`pN`** = **`number`** + **`number`** + … + **`number`**

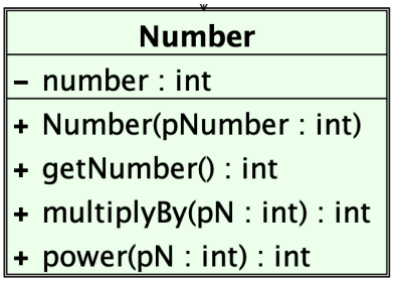
`pN` fois

Ajoutez un classe **`Test`** avec une méthode statique **`main`** qui permet de tester la méthode **`multiplyBy`**.

```
Entrez n : 12
Entrez m : 5
n x m = 60
```

EX-LOOP-23 Calcul d'une puissance par multiplication (* pour avancés)

Créez la classe **`Number`**.



Ajoutez à la classe **`Number`** de l'exercice précédent la méthode **`power`** qui calcule la puissance **`number ^ pN`** par multiplications successives et retourne le résultat.

`number ^ pN` = **`number`** x **`number`** x … x **`number`**

`pN` fois

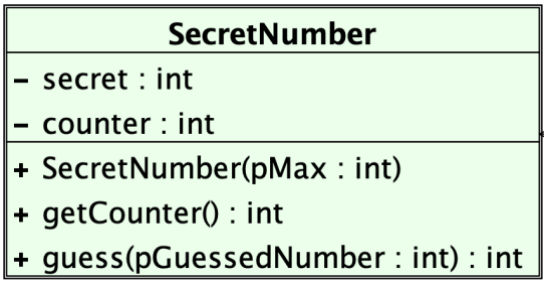
Pour effectuer les multiplications, vous devez utiliser la méthode **`multiplyBy`** et non pas l'opérateur **`*`**.

Ajoutez un classe **`Test`** avec une méthode statique **`main`** qui permet de tester la méthode **`power`**.

```
Entrez n : 2
Entrez m : 8
n ^ m = 256
```

EX-LOOP-24 Trouver le nombre secret, l'utilisateur joue

Adaptez la classe **`SecretNumber`** (voir exercice du cours) de la manière suivante.



Le constructeur initialise le nombre secret **`secret`** à un nombre entier aléatoire entre 1 et **`pMax`** (bornes incluses).

La méthode **`guess`** retourne :

- 0 si **`secret`** = **`pGuessedNumber`**
- -1 si **`secret`** < **`pGuessedNumber`**
- +1 si **`secret`** > **`pGuessedNumber`**

Ajoutez la classe **`UserPlays`** avec la méthode statique **`main`** qui permet à l'utlisateur de trouver le nombre secret par essais succesifs.

Lorsque l'utilisateur a trouvé le nombre secret alors le nombre d'essais est affiché.

```
Votre nombre: 50
Votre nombre est trop petit!
Votre nombre: 75
Votre nombre est trop petit!
Votre nombre: 88
Votre nombre est trop grand!
Votre nombre: 82
Votre nombre est trop petit!
Votre nombre: 86
Bravo!
Vous avez trouvé le nombre secret après 5 essais.
```

EX-LOOP-25 Nombre secret, l'ordinateur joue

Dans le projet de l'exercice précédent, remplacer la classe **`UserPlays`** par la classe **ComputerPlays** avec la méthode statique **`main`** qui trouve le nombre secret par une méthode naive en essayant tous les nombres à partir de 1 jusqu'à trouver le nombre secret.

```
L'ordinateur essaie le nombre : 1
Votre nombre est trop petit!
L'ordinateur essaie le nombre : 2
Votre nombre est trop petit!
L'ordinateur essaie le nombre : 3
Votre nombre est trop petit!
L'ordinateur essaie le nombre : 4
Votre nombre est trop petit!
L'ordinateur essaie le nombre : 5
Votre nombre est trop petit!
L'ordinateur essaie le nombre : 6
Bravo!
Vous avez trouvé le nombre secret après 6 essais.
```

EX-LOOP-26 Nombre secret, l'ordinateur joue mieux (* pour avancés)

Améliorez la stratégie de l'ordinateur en effectuant une recherche dichotomique.

La recherche dichomotomique consiste à délimiter le domaine de recherche par l'intervalle [min , max] et à essayer le nombre qui se trouve au milieu de ce domaine de recherche c'est-à-dire :

milieu = (max + min) / 2

3 cas peuvent alors se présenter :

- mileu = secret la recherche est terminée
- milieu < secret le domaine de recherche devient [x+1 , max]
- milieu > secret le domaine de recherche devient [min , x-1]

```
L'ordinateur essaie le nombre : 50
Votre nombre est trop grand!
L'ordinateur essaie le nombre : 25
Votre nombre est trop grand!
L'ordinateur essaie le nombre : 12
Votre nombre est trop grand!
L'ordinateur essaie le nombre : 6
Votre nombre est trop petit!
L'ordinateur essaie le nombre : 9
Bravo!
Vous avez trouvé le nombre secret après 5 essais.
```

EX-LOOP-27 Suite + 2 , x 2

Affichez la suite des termes qui sont inférieurs à une valeur plafond **`max`** saisie par l'utilisateur au clavier.

La suite est définie comme suit.

$u_0 = 1$

$u_1 = u_0 + 2 = 1 + 2 = 3$

$u_2 = u_1 \times 2 = 3 \times 2 = 6$

$u_3 = u_2 + 2 = 6 + 2 = 8$

:

en général :

- $u_i = u_{i-1} + 2$ si $i > 0$ et impair
- $ui = u_{i-1} \times 2$ si $i > 0$ et pair

```
Entrez la valeur plafond: 100
1
3
6
8
16
18
36
38
76
78
```

EX-LOOP-28 Décomposition d'un nombre

Affichez les chiffres composant un nombre entier positif **`n`** introduit au clavier en commencant avec le chiffre le moins significatif. Utilisez la méthode **`System.out.print()`** pour afficher plusieurs termes dans une ligne.

A la fin affichez le nombre de chiffres.

```
Entrez n: 12345
5 4 3 2 1
12345 est composé de 5 chiffres
```

EX-LOOP-29 Décomposition d'un nombre, ordre inverse

Même exercice que précédemment mais en commencant avec le chiffre le plus significatif. Utilisez une variable de type **`String`** dans laquelle vous ajoutez les chiffres.

```
Entrez n: 12345
1 2 3 4 5
12345 est composé de 5 chiffres
```

EX-LOOP-30 Somme des chiffres d'un nombre

Affichez la somme des chiffres (*Quersumme*) d'un nombre entier positif **`n`** introduit au clavier.

```
Entrez n: 12345
La somme des chiffres de 12345 est 15
```

EX-LOOP-31 Somme de chiffres d'un nombre, itérations (* pour avancés)

Affichez la somme des chiffres (*Quersumme*) d'un nombre entier positif **`n`** introduit au clavier. Ensuite continuez à calculer la somme des chiffres du résultat obtenu jusqu'à obtenir une somme inférieure à 10.

```
Entrez n: 999994
La somme des chiffres de 999994 est 49
La somme des chiffres de 49 est 13
La somme des chiffres de 13 est 4
```

EX-LOOP-32 Suite de Collatz

Ecrivez un programme qui affiche la suite de Collatz pour un entier naturel non nul donnée.

Cette suite est définie de la manière suivante :

> *On part d'un entier naturel non nul ; s'il est pair, on le divise par 2 ; s'il est impair, on le multiplie par 3 et on ajoute 1. En répétant l'opération, on obtient une suite récurrente, c.-à-d. une suite d'entiers positifs dont chacun ne dépend que de son prédécesseur. Par exemple, à partir de 14, on obtient la suite de Collatz suivante : 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1. La suite s'arrête si on tombe sur 1.*

Le programme devra aussi afficher le nombre de termes (18 dans l'exemple ci-dessus).

On suppose mais on n'a pas démontré jusqu'aujourd'hui que la suite de Collatz se termine pour chaque entier naturel.

Utilisez la méthode **`System.out.print()`** pour afficher plusieurs termes dans uen ligne.

```
Entrez n: 14
14 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
La suite de Collatz partant de 14 a 18 termes
```

EX-LOOP-33 Calcul de π, indication du nombre d'itérations

Le nombre π peut être calculé de manière approximative par la

Formule de Bernoulli: $$\frac{\pi^2}{6}=\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\ldots$$

ou: $$\pi=\sqrt{6\cdot\left(\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\ldots\right)}$$

On obtient la valeur exacte de π si le nombre de termes dans la formule est infini. Si le nombre de termes dans la formule est fini, on n'aboutit qu'à une approximation de la valeur exacte de π.

Exemples :

Avec 10 termes on obtient: 3.04936163598207
Avec 100 termes on obtient: 3.1320765318091053
Avec 1000 termes on obtient: 3.1406380562059946
Avec 10.000 termes on obtient: 3.1414971639472147

Complétez la méthode **`calculatePI`** qui calcule une approximation de π en utilisant les **`pN`** premiers termes de la formule de Bernoulli, **`pN`** étant fourni comme paramètre.

$$\pi=\sqrt{6\cdot\left(\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\ldots+\frac{1}{pN^2}\right)}$$

Développez un programme qui calcule une valeur approximative de π par la formule, où la valeur de n est entrée au clavier.

$$\pi=\sqrt{6\cdot\left(\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\ldots+\frac{1}{n^2}\right)}$$

Utilisez le type **`long`** pour obtenir une précision plus élevée.

```
Entrez n: 1000000
La valeur approximative de pi est 3.1415916986605086
```

EX-LOOP-34 Calcul de π, indication de la précision

Même exercice que l'exercice précédent. Mais au lieu d'indiquer le nombre d'itérations, l'utlisateur indique la précision voulue.

En notant π_i la valeur approximative de π trouvée après i itérations.

Le calul itératif s'arrête lorsque | **`Math.PI`** - π_i | < **ε**

où **ε** = 10^{-n} , **`n`** étant saisi au clavier par l'utilisateur

```
Entrez n: 6
La valeur calculée de pi est 3.1415916535905555
La valeur de Math.PI est     3.141592653589793
Le nombre d'itérations est   954931
```

EX-LOOP-35 Algorithme d'Héron

Écrivez un programme qui implémente l'algorithme d'Héron d'Alexandrie (ou méthode babylonienne) pour calculer $\sqrt{a}$.

L'utilisateur doit fournir :

- la valeur de *a* dont on veut calculer la racine carrée
- x_0 un réel > 0 quelconque, le premier terme de la suite d'Héron
- une précision ε > 0 avec laquelle on souhaite calculer $\sqrt{a}$, ε = 10^{-n} , **n** étant saisi au clavier par l'utilisateur

La suite d'Héron est définie récursivement par :

$$x_i = \begin{cases} x_0 & si\ i = 0 \\ \frac{1}{2}\left(x_{i-1} + \frac{a}{x_{i-1}}\right) & si\ i > 0 \end{cases}$$

On demande de calculer et d'afficher tous les termes de cette suite tant que l'écart (différence en valeur absolue) entre *a* et x_i soit $> \varepsilon$.

Exemple : pour calculer $\sqrt{2}$ en partant de $x_0 = 1$, on obtient $x_1 = \frac{3}{2} = 1{,}5$,

$x_2 = \frac{17}{12} = 1{,}41666\ldots$, $x_3 = \frac{577}{409} = 1{,}41422\ldots$ etc

```
Calculer la racine carrée de (a) : 2
Valeur de départ (x0)             : 1
Précision (n)                     : 15

epsilon = 1.0E-15

Math.sqrt( 2.0) = 1.4142135623730951

x0 = 1.0
x1 = 1.5
x2 = 1.4166666666666665
x3 = 1.4142156862745097
x4 = 1.4142135623746899
x5 = 1.414213562373095
```

EX-LOOP-36 Le plus grand d'une suite de nombres

Écrivez un programme qui calcule et affiche le plus grand d'une suite de nombres entiers positifs ou nuls introduits au clavier. La suite de ces nombres est terminée par l'introduction d'un nombre négatif.

```
Entrez le nombre 1 : 2
Entrez le nombre 2 : 9
Entrez le nombre 3 : 12
Entrez le nombre 4 : 5
Entrez le nombre 5 : -1
Le plus grand des 4 nombres est 12
```

EX-LOOP-37 Le plus petit d'une suite de nombres

Écrivez un programme qui calcule et affiche le plus petit d'une suite de nombres entiers positifs ou nuls introduits au clavier. La suite de ces nombres est terminée par l'introduction d'un nombre négatif.

```
Entrez le nombre 1 : 8
Entrez le nombre 2 : 5
Entrez le nombre 3 : 9
Entrez le nombre 4 : 3
Entrez le nombre 5 : -1
Le plus petit des 4 nombres est 3
```

EX-LOOP-38

Écrivez un programme qui renverse un nombre entier **n** entré au clavier. Vérifiez ensuite si le nombre n est un nombre palindrome. Un nombre est un palindrome s'il est symétrique.

Par exemple :

- 3, 12321, 99, 181 sont des nombres palindromes
- 122, 45, 12324 ne sont pas des nombres palindromes

```
Entrez n    : 123
n renversé : 321
```

```
Entrez n    : 12321
n renversé : 12321
12321 est un nombre palindrome
```

EX-LOOP-39 Triangles de chiffres

Écrivez un programme qui affiche les lignes suivantes dans la console :

a)

```
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

b)

```
1
12
123
1234
12345
123456
1234567
12345678
123456789
```

c)

```
999999999
88888888
7777777
666666
55555
4444
333
22
1
```

EX-LOOP-40 Motifs étoilés

a) Rectangles

```
Entrez le nombre de lignes : 5
Entrez le nombre de colonnes : 10
**********
**********
**********
**********
**********
```

b) Bordures

```
Entrez le nombre de lignes : 5
Entrez le nombre de colonnes : 10
**********
*        *
*        *
*        *
**********
```

c) Escalier descendant

```
Entrez le nombre de lignes : 10
*
**
***
****
*****
******
*******
********
*********
**********
```

d) Escalier ascendant

```
Entrez le nombre de lignes : 10
         *
        **
       ***
      ****
     *****
    ******
   *******
  ********
 *********
**********
```

e) Escalier double

```
Entrez le nombre de lignes : 8
        *
       ***
      *****
     *******
    *********
   ***********
  *************
***************
```

f) Croix

```
Entrez le nombre de lignes : 10
*         *
 *       *
  *     *
   *   *
    * *
     *
     *
    * *
   *   *
  *     *
 *       *
*         *
```

g) Damier 1

```
Entrez le nombre de lignes   : 5
Entrez le nombre de colonnes : 10
* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
```

h) Damier 2

```
Entrez le nombre de carrés dans une ligne : 3
Entrez la dimension d'un carré            : 5
***************
*     *     *     *
*     *     *     *
*     *     *     *
*     *     *     *
***************
*     *     *     *
*     *     *     *
*     *     *     *
*     *     *     *
***************
*     *     *     *
*     *     *     *
*     *     *     *
*     *     *     *
***************
```

i) Damier 3

```
Entrez le nombre de carrés dans une ligne : 4
Entrez la dimension d'un carré            : 5
**********************
*     *OOOO*     *OOOO*
*     *OOOO*     *OOOO*
*     *OOOO*     *OOOO*
*     *OOOO*     *OOOO*
**********************
*OOOO*     *OOOO*     *
*OOOO*     *OOOO*     *
*OOOO*     *OOOO*     *
*OOOO*     *OOOO*     *
**********************
*     *OOOO*     *OOOO*
*     *OOOO*     *OOOO*
*     *OOOO*     *OOOO*
*     *OOOO*     *OOOO*
**********************
*OOOO*     *OOOO*     *
*OOOO*     *OOOO*     *
*OOOO*     *OOOO*     *
*OOOO*     *OOOO*     *
**********************
```

j) Diamant

```
Entrez un nombre impair de lignes : 11
      *
     * *
    *   *
   *     *
  *       *
 *         *
*           *
 *         *
  *       *
   *     *
    *   *
     * *
      *
```

EX-LOOP-41 Factorisation première

Écrivez un programme qui effectue une factorisation première d'un nombre entier **n** entré au clavier.

```
Entrez n : 60
60 : 2
30 : 2
15 : 3
5 : 5
1
```

Exercices sur papier

Résolvez les exercices suivants sans utiliser Unimozer.

EX-LOOP-42 Indiquer le nombre d'astérisques affichés

Indiquez le nombre d'astérisques affichés dans la console.

```
for (int i = 1 ; i <= 10 ; i++)
{
     System.out.println("*");

}
```

```
for (int i = 0 ; i < 10 ; i++)
{
     System.out.println("*");

}
```

```
for (int i = -5 ; i <= 5 ; i++)
{
     System.out.println("*");

}
```

```
for (int i = 1 ; i <= 1 ; i++)
{
     System.out.println("*");

}
```

```
for (int i = 1 ; i < 1 ; i++)
{
     System.out.println("*");

}
```

```
for (int i = 0 ; i <= 1 ; i++)
{
     System.out.println("*");

}
```

EX-LOOP-43 Indiquer l'affichage dans la console

Indiquez ce qui est affiché dans la console.

```
for (int i = 1 ; i <= 3 ; i++)
{
     System.out.println(i + "");

}
```

Console

```
int s = 0;

for (int i = 1 ; i <= 3 ; i++)

{

     s = s + 2;

     System.out.println(i + "   " + s);

}
```

```
int s = 0;

for (int i = 1 ; i <= 3 ; i++)

{

     System.out.println(i + "   " + s);

     s = s + 2;

}
```

```
int s = 0;

for (int i = 1 ; i <= 3 ; i++)

{

     s = s + 2;

     System.out.println(i + "   " + s);

}
```

EX-LOOP-44 Boucle vide et infinie

Ecrivez un exemple d'une boucle **`for`** vide (où il n'y a aucune itération).

Ecrivez un exemple d'une boucle **`while`** vide (où il n'y a aucune itération).

Ecrivez un exemple d'une boucle **`while`** infinie (qui ne s'arrête jamais).

EX-LOOP-45 Transformer for en while et vice versa

Transformer la boucle **`for`** en boucle **`while`**

```
for (int i = 3 ; i <= 8 ; i++){
{
      System.out.println(i + "");
}
```

Transformer la boucle **`while`** en boucle **`for`**

```
int a = 10;
while (a > 0)
{
      System.out.println(a*a+"");
      a--;
}
```

Tableaux d'exécution (tableaux des variables)

EX-LOOP-46 Tableau d'exécution

```
1.    public int calculateSumEven(int pA , int pB)
2.    {
3.          int sum = 0;
4.          for (int i = pA ; i < pB; i++)
5.          {
6.                if (i % 2 == 0)
7.                {
8.                      sum = sum + i;
9.                }
10.         }
11.         return sum ;
12.   }
```

Faites un tableau des variables pour **pA** = 1 et **pB** = 4

				Valeur retournée :	

EX-LOOP-47 Tableau d'exécution

```
1.    public int gcdOptimizedEuclid(int pA, in pB)
2.    {
3.          int help = 0;
4.          while (pB != 0)
5.          {
6.                help = pA % pB;
7.                pA = pB;
8.                pB = help;
9.          }
10.         return pA;
11.   }
```

Faites un tableau des variables pour **pA** = 240 et **pB** = 64

				Valeur retournée :	

EX-LOOP-48 Tableau d'exécution

```
1.    public int lcmSearch(int pA, int pB)
2.    {
3.          int result = pB;
4.          while (result % pA != 0)
5.          {
6.               result = result + pB;
7.          }
8.          return result;
9.    }
```

Faites un tableau des variables pour **pA** = 7 et **pB** = 25

			Valeur retournée :		

EX-LOOP-49 Tableau d'exécution

```
1.    public String toBin(int pDec)
2.    {
3.          String result = "";
4.          if (pDec == 0)
5.          {
6.               result = "0";
7.          }
8.          else
9.          {
10.              while (pDec != 0)
11.              {
12.                   if (pDec %  2 == 0)
13.                   {
14.                        result = "0" + result;
15.                   }
16.                   else
17.                   {
18.                        result = "1" + result;
19.                   }
20.                   pDec = pDec / 2;
21.              }
22.         }
23.         return result;
24.   }
```

Faites un tableau des variables pour **pDec** = 0

				Valeur retournée :	

Faites un tableau des variables pour **pDEC** = 1, **pDec** = et **pDec** = 7

				Valeur retournée :	

Faites un tableau des variables pour **`pDec`** = 4

Valeur retournée :					

Faites un tableau des variables pour **`pDec`** = 7

Valeur retournée :					

Que fait cette méthode de manière générale ?

EX-LOOP-50 Tableau d'exécution

```
1.    public static void main(String[] args)
2.    {
3.          int x = 1;
4.          int y = 2;
5.          int z = 3;
6.          while (z != 1)
7.          {
8.                if (x == 1)
9.                {
10.                     x = y;
11.               }
12.               else
13.               {
14.                     if (y == 2)
15.                     {
16.                           y = z;
17.                     }
18.                     else
19.                     {
20.                           z = 1;
21.                     }
22.               }
23.         }
24.         System.out.println(x + ", " + y + ", " + z);
25.   }
```

Faites un tableau des variables.

Affichage dans la console :					