

Programmation Orientée Objets avec Java

Chapitre 1 : les bases de Java

Stéphane Malandain / Yassin Rekik
Semestre d'automne 2024

1.1 Exercice

- Écrire un code qui simule le comportement d'une boucle `while` à l'aide de la boucle `do while`
- Écrire un code qui simule le comportement d'une boucle `do while` à l'aide de la boucle `while`

```
import java.util.Arrays;

public class Loop {
    public static void whileWithDoWhile(int i) {
        // essayez par exemple de reproduire l'équivalent suivant:
        while(i < 5) {
            System.out.println(i);
            i++;
        }
        // devient:
        if (i < 5) {
            do {
                System.out.println(i);
                i++;
            } while (i < 5);
        }
    }

    public static void doWhileWithWhile(int i) {
        // essayez par exemple de reproduire l'équivalent suivant:
        do {
            System.out.println(i);
            i++;
        } while (i < 5);

        // devient:
        System.out.println(i);
        i++;
        while(i < 5) {
            System.out.println(i);
            i++;
        }
    }
}
```

```

public static void main(String[] args) {

    //whileWithDoWhile(10);
    //doWhileWithWhile(10);

    for (String a: args) {
        System.out.println(a);
    }
    for (int i = 0; i < args.length; i += 1) {
        System.out.println( args[i] );
    }
    int i = 0;
    while (i < args.length) {
        System.out.println( args[i] );
        i += 1;
    }
    Arrays.asList(args).forEach( a -> System.out.println(a) );
}

```

1.2 Exercice

Indiquez quelles lignes ne compilent pas :

```

1  short s = 10;
2  byte b = s;
3  int i = s;
4  long l = s;
5

```

- 1 : ok
- 2 : non
- 3 : ok
- 4 : ok

1.3 Exercice

Indiquez quelles lignes ne compilent pas :

```

1  short s = 10;
2  s = s + s;
3  s = s * 2;

```

- 2 : non
- 3 : non

```
opExample.java:4: error: incompatible types: possible lossy conversion from int to short
    s = s + s;
      ^
opExample.java:5: error: incompatible types: possible lossy conversion from int to short
    s = s * 2;
      ^
```

1.4 Exercice

```
1    int i = s++;
```

Que vaut i à la fin de l'exécution ?

```
[jshell> short s = 10;
s ==> 10

[jshell> int i = s++;
i ==> 10
```

1.5 Exercice

On dispose des méthodes suivantes :

```
1    void g(int n, float x) { ... }
2    void h(short s) { ... }
```

Et des déclarations suivantes :

```
1    int i;
2    byte b;
3    float f;
4    double d;
```

Spécifiez si les appels suivants sont corrects ou non tout en justifiant vos réponses

```

1  g(i, f);
2  g(b+1, f);
3  g(b, f);
4  g(i, d);
5  g(i, i);
6  g(i, 2*f);
7  g(i, 2.0*f);
8  h(b);
9  h(b+1);
10 h(5);
11 h(5.0);

```

```

1  // oui
2  // oui
3  // oui
4  // non
5  // oui
6  // oui
7  // Non
8  // oui
9  // non
10 // non
11 // non

```

Pratique

1.6 Exercice (Triangle)

A l'aide d'un paramètre `n` indiquant la hauteur, écrivez une fonction `printTriangle(int n)` qui affiche un triangle sous cette forme (ici, `n = 4`) :

```

*
**
***
****

```

```

public class serie1_16 {

    static void printTriangle(int n) {

        for (int i = 0; i < n; i++) {
            for (int j = 0; j <= i; j++) System.out.print("*");
            System.out.println();
        }
    }

    public static void main(String[] args) {

        printTriangle(4);

    }

}

```

1.7 Exercice (Sapin)

A l'aide d'un paramètre `n` indiquant la hauteur, écrivez une fonction `printSapin(int n)` qui affiche un sapin sous cette forme (ici, `n = 4`) :

```
*  
***  
*****  
*****
```

```
public class serie1_17 {  
  
    static void printSapin(int n) {  
        int up = 1;  
        for (int i = 0; i < n; i++) {  
            for (int j = 1; j < n - i; j++) System.out.print(" ");  
            for (int k = 0; k < up; k++) System.out.print("*");  
            System.out.println();  
            up += 2;  
        }  
    }  
    public static void main(String[] args) {  
        printSapin(4);  
    }  
}
```

1.8 Exercice (Factorielle)

Réalisez une fonction permettant de calculer la factorielle de n .

$$n! = \prod_{i=1}^n i$$

```

1  import java.util.Scanner;
2
3  public class FactorialTest {
4
5      Run | Debug
6      public static void main(String[] args) {
7
8          // Initialization of variables
9          int n ;
10         n = scanSide() ;
11         System.out.println("Factorial of " + n + "in : "+ getFactorial(n));
12     }
13
14     private static int scanSide() {
15         Scanner s = new Scanner(System.in);
16         int x;
17         do {
18             System.out.println(x: "Please enter n : ");
19             x = Integer.parseInt(s.next());
20             } while (x > 20 || x < 0);
21         return x ;
22     }
23
24     private static int getFactorial(int n) {
25         int result = 1 ;
26         for (int i = 1 ; i <= n ; i++) {
27             result = result * i ;
28         }
29         return result ;
30     }
31 }

```

1.9 Exercice (Pi)

Réalisez une fonction permettant de calculer une approximation de π :

$$\sum_{n=1}^N \frac{1}{n^4} = \frac{\pi^4}{90}$$

Cette fonction prend n en paramètre et retourne π

1.10 Exercice (Jeu du serpent)

Écrivez un programme qui affiche un serpent. A chaque itération, le serpent est affiché à l'aide d'une suite d'étoiles. Le programme demande ensuite d'augmenter ou de diminuer la taille du serpent. Le programme s'arrête lorsque la taille du serpent est nulle.

Exemple :

```

malandai@StephBook code % java serpent
(^)
Direction:+
*(^^)
Direction:+
**(^)
Direction:+
***(^)
Direction:+
****(^)
Direction:+
*****(^)
Direction:-
****(^)
Direction:-
***(^)
Direction:-
**(^)
Direction:-
*(^^)
Direction:-
(^)
Direction:-
Bye !

```

```

import java.util.Scanner;
import java.util.*;

public class serpent {

    public static void printSerpent(int n) {
        for (int i=1; i<n; i++) System.out.print("*");
        System.out.print("(^^)");
        System.out.println();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        boolean onContinue = true;
        int taille = 1;
        char direction = '+';
        String val;

        while (onContinue) {
            printSerpent(taille);
            System.out.print("Direction:");
            val = scanner.next();
            direction = val.charAt(0);

            if (direction == '+') taille ++;
            if (direction == '-') taille --;
            onContinue = (taille > 0);
        }
        scanner.close();
        System.out.println(" Bye !");
    }
}

```

1.11 Exercice (Vecteurs)

Réalisez un ensemble de fonctionnalités permettant le calcul vectoriel sur des vecteurs. Utilisez un tableau de doubles pour représenter un vecteur.

Écrivez les méthodes statiques suivantes :

- La méthode `add`. Elle prend deux vecteurs et retourne leur addition.
Ex. `add(new double[]{1.0, 2.0, 3.0}, new double[]{2.0, 1.0, -5.0})` retourne `double[3]{3.0, 3.0, -3.0}`.
- La méthode `mul` qui multiplie un vecteur par une valeur numérique.
- A l'aide des 2 premières fonctions, écrivez la méthode statique `sub` qui soustrait le deuxième vecteur du premier.
- La méthode `len` qui retourne le nombre de composante d'un vecteur
- `norm` qui retourne la norme (ou la longueur) d'un vecteur.
Par exemple, `norm(new double[]{1.0, 2.0, 2.0})` retourne `3.0`.

$$\vec{v} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \text{ est } \|\vec{v}\| = \sqrt{v_1^2 + v_2^2 + v_3^2}$$

- Le calcul de la norme d'un vecteur
- `sum` qui prend une liste de vecteurs en paramètre et les additionne
- `norms` qui prend une liste de vecteurs en paramètre et retourne la norme de leur addition
- `concat` qui concatène 2 vecteurs
- `sliceFrom` qui retourne un sous-ensemble du vecteur jusqu'à un indice (non compris)
- `slice` qui est la combinaison des deux précédentes (avec un index de début et un de fin)

Remarques :

- Respectez le nommage
- Respectez les conventions de nommage du langage :

<https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html>

- Écrivez des méthodes courtes et concises
- Retournez des nouvelles copies de tableaux. Ne modifiez pas vos arguments
- Utilisez les fonctionnalités offertes par la classe `Arrays` :

<https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Arrays.html>

- Écrivez vos méthodes statiques dans un fichier `Vector.java` (sans `main`)
- La méthode principale avec vos tests doit se trouver dans un fichier `App.java`
- Les fonctions qui retournent des vecteurs doivent retourner un vecteur vide si les vecteurs entrés en arguments ne sont pas conformes !

1.12 Exercice (Matrices)

Réalisez des fonctionnalités sur les matrices :

- L'affichage d'une matrice
- La multiplication de deux matrices
- L'addition de deux matrices

Retournez une matrice vide en cas d'erreur


```

1 public class Matrix {
2
3     public static void displayMatrix(int mat[][]) {
4         // Loop through all rows
5         for (int i = 0; i < mat.length; i++) {
6             // Loop through all elements of current row
7             for (int j = 0; j < mat[i].length; j++)
8                 System.out.format(format: "%03d ", mat[i][j]);
9             System.out.println();
10        }
11    }
12
13    public static int [][] addMatrix(int mat1[][], int mat2[][]) {
14        // Matrix dimentions to be tested
15        int dim1_1 = mat1.length ;
16        int dim2_1 = mat2.length ;
17        int dim1_2 = mat1[0].length ;
18        int dim2_2 = mat2[0].length ;
19
20        // The result Matrix
21        int [][] result = new int[dim1_1][dim2_2];
22
23        if (dim1_1 == dim2_1 && dim1_2 == dim2_2) {
24            for (int i = 0; i < dim1_1; i++) {
25                for (int j = 0; j < dim1_2; j++)
26                    result[i][j] = mat1[i][j] + mat2[i][j] ;
27            }
28            System.out.println(x: "Addition DONE");
29            return result ;
30        } else {
31            System.out.println(x: "Error : dimensions are not correct");
32            return result ;
33        }
34    }
35
36    public static int[][] multiplyMatrix(int[][] mat1, int[][] mat2) {
37        // Matrix dimentions to be tested
38        int dim1_1 = mat1.length ;
39        int dim2_1 = mat2.length ;
40        int dim1_2 = mat1[0].length ;
41        int dim2_2 = mat2[0].length ;
42        // The result Matrix
43        int [][] result = new int[dim1_1][dim2_2];
44
45        if (dim1_2 == dim2_1) {
46            for (int row = 0; row < dim1_1; row++) {
47                for (int col = 0; col < dim2_2; col++) {
48                    result[row][col] = calculateCell(mat1, mat2, row, col);
49                }
50            }
51            System.out.println(x: "Multiplication DONE");
52            return result ;
53        } else {
54            System.out.println(x: "Error : dimensions are not correct");
55            return result ;
56        }
57    }
58

```

```

58
59
60 private static int calculateCell(int[][] firstMatrix, int[][] secondMatrix, int row, int col) {
61     int cell = 0;
62     int dim = secondMatrix.length ;
63     for (int i = 0; i < dim ; i++) {
64         cell += firstMatrix[row][i] * secondMatrix[i][col];
65     }
66     return cell;
67 }
68
69 Run | Debug
70 public static void main(String args[]){
71     int A[][] = { { 1, 2, 3, 4 },
72                  { 5, 6, 7, 8 },
73                  { 9, 10, 11, 12 } };
74     int B[][] = { { 1, 2, 3, 4 },
75                  { 5, 6, 7, 8 },
76                  { 9, 10, 11, 12 } };
77     int C[][] = { { 1, 2, 3, 4 },
78                  { 5, 6, 7, 8 },
79                  { 9, 10, 11, 12 } ,
80                  { 13, 14, 15, 16 }
81                };
82
83     displayMatrix(A);
84     System.out.println(x: "-----");
85     displayMatrix(B);
86     System.out.println(x: "-----");
87     displayMatrix(B);
88     System.out.println(x: "-----");
89     displayMatrix(addMatrix(A,B));
90     System.out.println(x: "-----");
91     displayMatrix(multiplyMatrix(A,B));
92     System.out.println(x: "-----");
93     displayMatrix(multiplyMatrix(A,C));
94 }
95

```

1.13 Exercice (Tableaux)

Écrivez une méthode qui compte le nombre d'éléments positifs dans un tableau de tableaux. Un exemple d'utilisation est fourni dans la méthode principale `main` suivante :

```

1  import java.util.*;
2  public class exo1_13 {
3
4      public static int countPositive(double tab[][]) {
5          int pos = 0;
6          for (double tab1[] : tab) {
7              for (double ds : tab1) {
8                  if (ds>0.0) pos +=1;
9              }
10         }
11         return pos;
12     }
13
14     Run | Debug
15     public static void main(String[] args) {
16         double[][] example = {
17             {1.0, -2.0},
18             {3.0, -1.0, 2.1},
19             {1.5, -2.5, 3.0}
20         };
21
22         int res = countPositive( example );
23         System.out.println("Count:" + res);
24         // Affiche "Count : 4"
25     }
26 }

```

1.14 Exercice (Strings)

Écrivez les fonctionnalités ci-dessous :

```

import java.util.List;

public class App {

    /* Détermine si un nombre qui se trouve dans une chaîne de caractères contient
    * un entier
    * Note: prend en compte le signe, mais ne prend pas en compte les espaces
    * Exemples:
    * isNumeric("42") -> true
    * isNumeric(" 22 ") -> true
    * isNumeric(" -33 ") -> true
    * isNumeric(" 22.0") -> false
    * isNumeric("2f3") -> false */

```

```

public static boolean isNumeric(String term) {
    char[] chars = term.trim().toCharArray();
    if (chars[0] == '-') {
        chars[0] = '0';
    }
    for (char c: chars) {
        if (!Character.isDigit(c)) {
            return false;
        }
    }
    return true;
}

/* Retourne un tableau d'indices où chaque valeur indique la position d'un
 * caractère dans une chaîne
 * Exemples:
 * indexes("maison", 'i') -> {2}
 * indexes("tralala", 'a') -> {2,4,6}
 * indexes("coucou", 'x') -> {} */
public static int[] indexes(String term, char c) {
    List<Integer> indexes = new ArrayList<>();
    int i = 0;
    for (char ch: term.toCharArray()){
        if (ch == c) {
            indexes.add( i );
        }
        i += 1;
    }
    return transform( indexes );
}

private static int[] transform(List<Integer> is) {
    int[] res = new int[is.size()];
    for (int i = 0; i < is.size(); i += 1) {
        res[i] = is.get(i);
    }
    return res;
}
}

```