

TP1 : Les tableaux et les chaines de caractères

Radwane Khemisse

23 Octobre 2024

Introduction

Ce rapport détaille les solutions apportées aux exercices du TP1 en Java, portant sur les tableaux et les chaînes de caractères. Le TP contient quatre exercices, et j'essaierai de justifier mes choix dans la résolution des exercices.

1 Exercice 1:

1.1 Énoncé

Écrivez un programme Java qui range des notes des étudiants saisies au clavier dans un tableau nommé `notes`, et qui permet de faire les opérations suivantes :

1. Triez et affichez la liste des notes.
2. Affichez la note moyenne.
3. Affichez la note maximale et minimale.
4. Affichez le nombre d'étudiants ayant une note saisie par l'utilisateur.

NB : Pour trier le tableau, vous utilisez `Arrays.sort()`.

1.2 Résolution de l'exercice

Pour résoudre cet exercice, j'ai créé une classe appelée **NotesManagement** qui contient plusieurs méthodes.

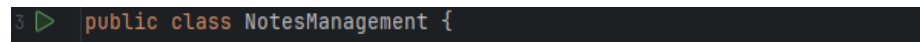
A screenshot of a code editor showing the beginning of a Java class definition. The text is: `public class NotesManagement {`. The code is highlighted in a dark background with light-colored text. There is a small green play button icon to the left of the code.

Figure 1: Creation de la classe NoteManagement

1.2.1 La Méthode DisplayListedNotes(float[] tab)

```
4  @ public void DisplayListedNotes(float[] tab) 1 usage
5  {
6      float p;
7      for (int i = 0; i < tab.length; i++)
8      {
9          float mini=tab[i];
10         int indexmini=i;
11         for (int j = i ; j < tab.length; j++)
12         {
13             if ( tab[j] < mini )
14             {
15                 mini=tab[j];
16                 indexmini=j;
17             }
18         }
19         p=tab[i];
20         tab[i]=tab[indexmini];
21         tab[indexmini]=p;
22         System.out.println(tab[i]);
23     }
24 }
25 }
```

Figure 2: La methode DisplayListedNotes

La méthode `DisplayListedNotes` prend en paramètres un tableau qui contient les notes des étudiants et les affiche dans l'ordre croissant. Pour trier la table, j'ai choisi d'utiliser un tri par sélection. C'est-à-dire de diviser la boucle en deux parties : une partie déjà triée et une partie non triée. Pour réaliser ce trie, on crée deux boucles : la première itère sur chaque élément du tableau et la deuxième recherche la note minimale `mini` dans la portion non triée l'intervalle `[i,tab.length()`]. À chaque itération, la note minimale trouvée est échangée avec l'élément en cours d'itération. À la fin de chaque itération, j'affiche `tab[i]` qui est le dernier élément de l'espace trié. Cela nous permettra d'afficher un tableau trié. A la fin de chaque itteration j'affiche `tab[i]` .

1.2.2 La Méthode GetAverage(float[] tab)

```
26 @      public float GetAverage(float[] tab) 1 usage
27      {
28          float sum=0;
29          for (float i : tab)
30          {
31              sum += i;
32          }
33          return sum/tab.length;
34      }
```

Figure 3: GLa methode GetAverage

Cette methode prend en parametres le tableau des notes et retourne un float : la moyenne des notes en additionnant toutes le notes et les stocker en `sum` et apres divisant `sum` par le nombre total des notes.

1.2.3 La Méthode GetMax(float[] tab)

```
35 @      public float GetMax(float[] tab) 1 usage
36      {
37          float maxi=tab[0];
38          for (float i : tab)
39          {
40              if (maxi < i)
41              {
42                  maxi=i;
43              }
44          }
45          return maxi;
46      }
```

Figure 4: La methode GetMax

Cette méthode prend en paramètres le tableau des notes et retourne un float: le maximum des notes dans le tableau. Pour faire cela, on initialise `maxi` au premier élément du tableau et on itère sur le tableau. À chaque fois qu'on trouve une valeur plus grande que `maxi`, on met à jour la valeur de `maxi`.

1.2.4 La Méthode GetMin(float[] tab)

```
47 @ public float GetMin(float[] tab) 1 usage
48 {
49     float mini=tab[0];
50     for (float i : tab)
51     {
52         if (i < mini)
53         {
54             mini=i;
55         }
56     }
57     return mini;
58 }
```

Figure 5: La methode GetMin

Cette méthode utilise la même logique que la méthode précédente `GetMax`, juste que pour celle-là on cherche le minimum.

1.2.5 La Méthode NoteCardinal(float[] tab, float note)

```
59 @ public int NoteCardinal(float[] tab,float note) 1 usage
60 {
61     int number=0;
62     for (float i : tab)
63     {
64         if (note == i)
65         {
66             number++;
67         }
68     }
69     return number;
70 }
71
```

Figure 6: La methode NoteCardinal

Cette méthode prend une note spécifique et compte combien de fois elle apparaît dans le tableau des notes.

1.3 La Méthode main

```
72 > public static void main(String[] args) {  
73     Scanner sc = new Scanner(System.in);  
74     NotesManagement nm = new NotesManagement();  
75  
76     System.out.print("Enter the number of notes you wish to add: ");  
77     int n = sc.nextInt();  
78     float[] tab = new float[n];  
79     for (int i = 0; i < n; i++)  
80     {  
81         System.out.print("Enter the mark number "+(i+1)+" : ");  
82         tab[i] = sc.nextFloat();  
83     }  
84     System.out.println("Listed marks: ");  
85     nm.DisplayListedNotes(tab);  
86  
87     System.out.println("The average of the marks is: "+nm.GetAverage(tab));  
88  
89     System.out.println("The maximum of the marks is: "+nm.GetMax(tab));  
90  
91     System.out.println("The minimum of the marks is: "+nm.GetMin(tab));  
92  
93     System.out.print("Enter the marks You want to check its cardinal: ");  
94     float chosen_number = sc.nextFloat();  
95     System.out.println("the Cardinal of the mark: "+ chosen_number + " is: "+nm.NoteCardinal(tab,chosen_number));  
96 }
```

Figure 7: La methode Main

Voici un exemple d'exécution:

```
"C:\Users\radwane.khemisse\.jdk\openjdk-23.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.3\lib\idea_rt.jar=58932:C:\Progr  
Enter the number of notes you wish to add: 9  
Enter the mark number 1: 9  
Enter the mark number 2: 7  
Enter the mark number 3: 5  
Enter the mark number 4: 9  
Enter the mark number 5: 3  
Enter the mark number 6: 6  
Enter the mark number 7: 9  
Enter the mark number 8: 3  
Enter the mark number 9: 9  
Listed marks:  
3.0  
3.0  
5.0  
6.0  
7.0  
9.0  
9.0  
9.0  
The average of the marks is: 6.6666665  
The maximum of the marks is: 9.0  
The minimum of the marks is: 3.0  
Enter the marks You want to check its cardinal: 9  
the Cardinal of the mark: 9.0 is: 4
```

Figure 8: Exemple d'implémentation

2 Exercice 2:

2.1 Énoncé

Écrire un programme qui lit un verbe du premier groupe et qui en affiche la conjugaison au présent sous la forme suivante :

```
Entrez un verbe du premier groupe : chanter
je chante
tu chantes
il chante
nous chantons
vous chantez
ils chantent
```

Le programme vérifiera que le verbe se termine bien par "er" et on supposera qu'il s'agit d'un verbe régulier.

2.2 Résolution de l'exercice

Pour résoudre cet exercice, j'ai créé une classe appelée Conjugaison qui contient une méthode pour conjuguer le verbe au présent.

```
public class Conjugaison {
    public void Present(String verb){ 1 usage
        String terminaison = verb.substring( verb.length()-2);
        if (terminaison.equals("er"))
        {
            String cardinal = verb.substring(0, verb.length()-2);
            System.out.println("je " + cardinal + "e");
            System.out.println("tu " + cardinal + "es");
            System.out.println("il " + cardinal + "e");
            System.out.println("nous " + cardinal + "ons");
            System.out.println("vous " + cardinal + "ez");
            System.out.println("ils " + cardinal + "ent");
        }
    }
}
```

Figure 9: creation de la classe Conjugaison

2.2.1 La Méthode Present(String verb)

La méthode **Present** prend un verbe en paramètre et commence par vérifier que ce verbe se termine par er . Si c'est le cas, elle extrait le radical du verbe et utilise ce radical pour former les différentes conjugaisons au présent :

- Prenom + radical + terminaison du present

2.2.2 La Méthode Main

```
17
18 ► public static void main(String[] args) {
19     Conjugaison c = new Conjugaison();
20     Scanner sc = new Scanner(System.in);
21
22     System.out.print("Enter a verb you want to conjugate to present: ");
23     String verb = sc.nextLine();
24
25     System.out.println("Here is the conjugation of the verb: " + verb );
26     c.Present(verb);
27 }
```

Figure 10: La methode Main

Voici l'exécution du programme:

```
"C:\Users\radwane khemisse\.jdk\openjdk-23.0.1\bin\java.exe" "-javaagent:C:\Program I
Enter a verb you want to conjugate to present: Parler
Here is the conjugation of the verb: Parler
je Parle
tu Parles
il Parle
nous Parlons
vous Parlez
ils Parlent

Process finished with exit code 0
```

Figure 11: Example d'implémentation

3 Exercice 3:

3.1 Énoncé

Écrivez un programme Java permettant d'effectuer un ensemble d'opérations sur une chaîne de caractères quelconque saisie à partir du clavier. Ce programme doit offrir un menu pour sélectionner les opérations suivantes :

1. **Saisir** : lire une chaîne de caractères à partir du clavier et la stocker dans une variable.
2. **Afficher** : afficher la chaîne saisie.
3. **Inverser** : inverser la chaîne saisie.
4. **Nombre de mots** : compter le nombre de mots de la chaîne (le caractère blanc étant le séparateur).

3.2 Resolution de l'exercice 3:

Pour résoudre cet exercice, j'ai conçu une classe nommée Menu qui permet de gérer l'interaction avec l'utilisateur. La classe contient plusieurs méthodes pour effectuer les opérations demandées et un menu de navigation.

```
public class Menu {
```

Figure 12: Creation de la Classe Menu

3.2.1 La Méthode Saisir()

```
3     private String str = ""; 5 usages
4     private Scanner scanner = new Scanner(System.in); 1 usage
5     public void Saisir() { 1 usage
6         System.out.print("Enter the phrase : ");
7         str = scanner.nextLine();
```

Figure 13: Methode Saisir

La méthode **Saisir** lit une chaîne de caractères saisie par l'utilisateur et la stocke dans une variable **str**.

3.2.2 La Méthode Afficher()

```
9      public void Afficher() 1 usage
10     {
11         System.out.println("the phrase entered is : "+str);
12     }
```

Figure 14: Methode Afficher

Cette méthode affiche simplement la chaîne de caractères stockée.

3.2.3 La Méthode Inverser()

```
13     public void Inverser() 1 usage
14     {
15         String str2 = "";
16         for (int i = str.length() - 1; i >= 0; i--) {
17             str2 += str.charAt(i);
18         }
19         System.out.println("reversed phrase is: " + str2);
20     }
```

Figure 15: Methode Inverser

La méthode `Inverser` crée une nouvelle chaîne `str2` en parcourant la chaîne initiale de la fin vers le début.

3.2.4 La Méthode NombreMots()

```
21     public void NombreMots() 1 usage
22     {
23         String[] phrase = str.trim().split(regex: "\\s+");
24         System.out.println("Number of words: " + phrase.length);
25     }
```

Figure 16: Methode NombreMots()

Cette méthode compte le nombre de mots dans la chaîne, en utilisant `split("\\s+")` pour diviser la chaîne en fonction des espaces après l'avoir trimée. l'utilisation de regex gère correctement les espaces multiples et ne compte pas les blancs comme des mots, ce qui est une solution courante et efficace pour compter les mots dans une phrase.

3.3 Implementation de la methode Main

```
27 public static void main(String[] args) {
28     Menu menu = new Menu();
29     Scanner scanner = new Scanner(System.in);
30     int choice;
31
32     do {
33         System.out.println("\n== Menu ==");
34         System.out.println("1. Enter a phrase");
35         System.out.println("2. Display the phrase");
36         System.out.println("3. Inverse the phrase");
37         System.out.println("4. Count the number of words");
38         System.out.println("5. Quit");
39         System.out.print("Choose between (1-5): ");
40
41         choice = scanner.nextInt();
42         scanner.nextLine();
43
44         switch (choice) {
45             case 1:
46                 menu.Saisir();
47                 break;
48             case 2:
49                 menu.Afficher();
50                 break;
51
52             case 3:
53                 menu.Inverser();
54                 break;
55             case 4:
56                 menu.NombreMots();
57                 break;
58             case 5:
59                 System.out.println("Good bye !");
60                 break;
61             default:
62                 System.out.println("Invalid choice: choose between (1-5)!");
63         }
64
65         if (choice != 5) {
66             System.out.println("\nTouch the keyboard to go back to the main menu!");
67             scanner.nextLine();
68         }
69     } while (choice != 5);
70
71 }
```

Figure 17: Methode Main

Le menu principal affiche les options et utilise un `switch` pour gérer l'interaction avec l'utilisateur. Après chaque opération, le programme invite l'utilisateur à

appuyer sur une touche pour revenir au menu.

Voici l'exemple d'implémentation:

```
"C:\Users\Radwane Khemisse\.jdk\openjdk-23.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.3\lib\idea_rt.jar=04715:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.3\bin" -Dfile.encoding=UTF-8
=== Menu ===
1. Enter a phrase
2. Display the phrase
3. Inverse the phrase
4. Count the number of words
5. Quit
Choose between (1-5): 1
Enter the phrase : My name Is Radwane Khemisse

Touch the keyboard to go back to the main menu!

=== Menu ===
1. Enter a phrase
2. Display the phrase
3. Inverse the phrase
4. Count the number of words
5. Quit
Choose between (1-5): 2
the phrase entered is : My name Is Radwane Khemisse

Touch the keyboard to go back to the main menu!
```

```
=== Menu ===
1. Enter a phrase
2. Display the phrase
3. Inverse the phrase
4. Count the number of words
5. Quit
Choose between (1-5): 3
reversed phrase is: essimehk enawdaR sI eman yM

Touch the keyboard to go back to the main menu!

=== Menu ===
1. Enter a phrase
2. Display the phrase
3. Inverse the phrase
4. Count the number of words
5. Quit
Choose between (1-5): 4
Number of words: 5

Touch the keyboard to go back to the main menu!
```

```
Touch the keyboard to go back to the main menu!
```

```
5
```

```
=== Menu ===
```

```
1. Enter a phrase
```

```
2. Display the phrase
```

```
3. Inverse the phrase
```

```
4. Count the number of words
```

```
5. Quit
```

```
Choose between (1-5): 5
```

```
Good bye !
```

Figure 18: Exemple d'Implementation

4 Exercice 4:

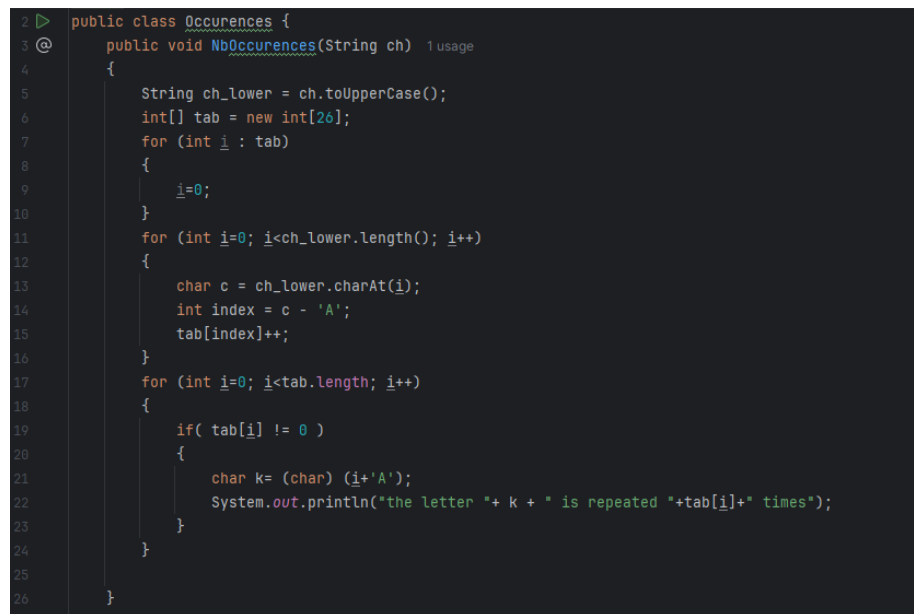
4.1 Énoncé

Écrivez un programme Java qui lit une chaîne de caractères **ch** au clavier et qui compte les occurrences des lettres de l'alphabet en ne distinguant pas les majuscules des minuscules. Utilisez un tableau **nb_occurrences** de dimension 26 pour mémoriser le résultat. Affichez seulement le nombre des lettres qui apparaissent au moins une fois dans le texte.

4.2 Resolution de l'exercice

J'ai d'abord créer la classe **Occurrences** qui comporte principalement la methode principale **NbOccurrences** et la methode **main**.

4.2.1 La methode Nboccurrences

A screenshot of a code editor showing the implementation of the `NbOccurrences` method in a Java class named `Occurrences`. The code is as follows:

```
2 public class Occurrences {
3     public void NbOccurrences(String ch) 1 usage
4     {
5         String ch_lower = ch.toUpperCase();
6         int[] tab = new int[26];
7         for (int i : tab)
8         {
9             i=0;
10        }
11        for (int i=0; i<ch_lower.length(); i++)
12        {
13            char c = ch_lower.charAt(i);
14            int index = c - 'A';
15            tab[index]++;
16        }
17        for (int i=0; i<tab.length; i++)
18        {
19            if( tab[i] != 0 )
20            {
21                char k= (char) (i+'A');
22                System.out.println("the letter "+ k + " is repeated "+tab[i]+" times");
23            }
24        }
25    }
26 }
```

Figure 19: Methode NbOccurrences

La méthode `NbOccurrences` prend en paramètre une chaîne de caractères `ch`, et applique les operations suivantes a fin de calculer le nombre d'occurrence de chaque caractere de `ch`.

- On commence par la conversion de `ch` en majuscule utilisant `toUpperCase()` cela nous permetre de traiter 'a' et 'A' de la meme maniere comme Mentionner dans l'Énoncé.

- Apres j'ai cree un tableau tab de taille 26. ce tableau a pour besoin de storer le nombre d'occurrences de chaque lettre de l'alphabet avec ('A' correspond à 0, 'B' à 1, etc.)
- alors pous compter le nombre d'occurences je vais parcourir chaque caractere c dans ch. Pour chaque caractere on calcule son index relatif a 'A' (c - 'A') pour ce cas ca serai 2. alors on incremente la valeur correspondante dans le tableau des occurences.
- Pour l'affichage on parouct notre tableau tab et on affiche seulement les lettres ayant une occurence non nulle.

4.2.2 La methode Main

Pour le Main on commence par initialise l'objet de type Occurences et apres la demande de l'utilisateur d'entrer une phrase on appelle notre methode NbOccurences

```

27 public static void main(String[] args) {
29     Scanner sc = new Scanner(System.in);
30     Occurences occ = new Occurences();
31     System.out.print("Enter the phrase: ");
32     String ch = sc.nextLine();
33     System.out.println("Here's an overview of the occurences of the caracters : ");
34     occ.NbOccurences(ch);
35 }

```

Figure 20: Methode Main

Voici un exemple de l'utilisation du programme

```

"C:\Users\radwane khemisse\.jdk\openjdk-23.0.1\bin\java.exe" "-javaagent:C:\Program I
Enter the phrase: Jeanne
Here's an overview of the occurences of the caracters :
the letter A is repeated 1 times
the letter E is repeated 2 times
the letter J is repeated 1 times
the letter N is repeated 2 times

Process finished with exit code 0

```

Figure 21: Exemple d'Implimentation

Conclusion

Ce TP a été l'occasion de manipuler des tableaux, de traiter des chaînes de caractères, d'appliquer la programmation orientée objet. et de s'habituer plus au syntax de Java.