Compte rendu
TP 1

Source code : https://github.com/YassirJr/jeraidi-yassir-java-oop.git

Elaboré par : **Jeraidi Yassir**

Encadrant pédagogique : **Aminou Loubna**

- Exercice 1

Notes

Ce projet contient une classe Java `Notes` qui fournit diverses fonctionnalités pour gérer et manipuler un tableau de notes flottantes (float).

Définition de Classe et Attributs

```
package org.example.ex1;

import java.util.Arrays;

public class Notes {
    private final float[] notes;
    public Notes(float[] notes) {
        this.notes = notes;
    }
    ...
```

- `package org.example.ex1;`: Définit que la classe `Notes` fait partie du package `org.example.ex1`.
- `import java.util.Arrays;`: Permet à la classe d'utiliser les méthodes utilitaires pour les tableaux définies dans la classe `java.util.Arrays`.

Attribut

- `private final float[] notes;`: Un champ privé qui contient un tableau de flottants. Le mot-clé `final` indique que la référence à ce tableau ne peut pas être modifiée après son initialisation.

Constructeur

```
public Notes(float[] notes) {
    this.notes = notes;
}
```

- `public Notes(float[] notes)`: Constructeur de la classe `Notes` qui prend un tableau de flottants comme argument.
- `this.notes = notes;`: Initialise le champ `notes` avec le tableau passé au constructeur.

Méthodes

`sortAndShow`

```
public void sortAndShow() {
    Arrays.sort(notes);
```

```
    System.out.println("Le tri des notes : " + Arrays.toString(notes));
}
```

- `public void sortAndShow()`: Trie le tableau `notes` et l'affiche.
- `Arrays.sort(notes);`: Trie le tableau dans l'ordre croissant.
- `System.out.println("Le tri des notes : " + Arrays.toString(notes));`: Affiche le tableau trié en utilisant `Arrays.toString(notes)` pour convertir le tableau en représentation string.

getAverage

```
public float getAverage() {
    float sum = 0;
    for (float num : notes) {
        sum += num;
    }
    return sum / notes.length;
}
```

- `public float getAverage()`: Calcule et retourne la moyenne des notes.
- `float sum = 0;`: Initialise une variable pour contenir la somme des notes.
- `for (float num : notes) { sum += num; }`: Une boucle for-each qui itère sur chaque élément du tableau `notes` et les ajoute tous à `sum`.
- `return sum / notes.length;`: Retourne la moyenne en divisant la somme par le nombre d'éléments dans le tableau.

getMin

```
public float getMin() {
    float min = notes[0];
    for (float num : notes) {
        if (num < min) {
            min = num;
        }
    }
    return min;
}
```

- `public float getMin()`: Trouve et retourne la valeur minimale dans le tableau `notes`.
- `float min = notes[0];`: Initialise `min` avec le premier élément du tableau.
- `for (float num : notes) { if (num < min) { min = num; } }`: Une boucle qui itère sur le tableau `notes`. Si une valeur inférieure à `min` est trouvée, elle met à jour `min`.
- `return min;`: Retourne la valeur minimale trouvée.

getMax

```
public float getMax() {
    float max = notes[0];
    for (float num : notes) {
        if (num > max) {
            max = num;
        }
    }
    return max;
}
```

- `public float getMax()`: Trouve et retourne la valeur maximale dans le tableau `notes`.
- `float max = notes[0];`: Initialise `max` avec le premier élément du tableau.
- `for (float num : notes) { if (num > max) { max = num; } }`: Une boucle qui itère sur le tableau `notes`. Si une valeur supérieure à `max` est trouvée, elle met à jour `max`.
- `return max;`: Retourne la valeur maximale trouvée.

getRepeatedNote

```
public int getRepeatedNote(int note) {
    int count = 0;
    for (float num : notes) {
        if (num == (float) note) {
            count++;
        }
    }
    return count;
}
```

- `public int getRepeatedNote(int note)`: Compte et retourne le nombre de fois qu'une note spécifiée apparaît dans le tableau `notes`.
- `int count = 0;`: Initialise un compteur à zéro.
- `for (float num : notes) { if (num == (float) note) { count++; } }`: Une boucle qui itère sur le tableau `notes`. Si un élément correspond à la note spécifiée (après l'avoir convertie en float), elle incrémente le compteur.
- `return count;`: Retourne le nombre d'occurrences de la note spécifiée.

Classe Principale : `Main`

Ce fichier contient le point d'entrée principal du programme avec la logique suivante :

1. **Package et Importation** : `java package org.example.ex1; import java.util.Scanner;`
   - Le code fait partie du package `org.example.ex1`.
   - Il importe `java.util.Scanner` pour lire les entrées de l'utilisateur.
2. **Méthode Principale** : `java public static void main(String[] args) {`
   - La méthode `main` est le point d'entrée du programme.
3. **Création d'une Instance de Scanner** : `java Scanner scanner = new Scanner(System.in);`
   - Un objet `Scanner` est créé pour lire l'entrée depuis l'entrée standard (clavier).
4. **Demande de Nombre d'Étudiants** : `java System.out.println("Combien d'etudiants ? : "); float[] notesFromInput = new float[scanner.nextInt()];`
   - L'utilisateur est invité à entrer le nombre d'étudiants.
   - Un tableau `notesFromInput` est créé pour stocker les notes, avec une longueur déterminée par l'entrée de l'utilisateur.
5. **Lecture des Notes** : `java for (int i = 0; i < notesFromInput.length; i++) { System.out.printf("Note %d : ", (i + 1)); notesFromInput[i] = scanner.nextFloat(); }`
   - Une boucle est utilisée pour lire la note de chaque étudiant.
   - Chaque note est stockée dans `notesFromInput`.
6. **Création d'un Objet `Notes`** : `java Notes notes = new Notes(notesFromInput);`
   - Une instance de la classe `Notes` est créée, initialisée avec le tableau `notesFromInput`.
7. **Tri et Affichage des Notes** : `java notes.sortAndShow();`
   - La méthode `sortAndShow` de la classe `Notes` est appelée pour trier les notes et les afficher.
8. **Calcul et Affichage des Statistiques** : `java System.out.println("Moyenne : " + notes.getAverage()); System.out.println("Min : " + notes.getMin()); System.out.println("Max : " + notes.getMax());`
   - Les méthodes de la classe `Notes` sont appelées pour afficher la moyenne (`getAverage`), la note minimale (`getMin`) et la note maximale (`getMax`).
9. **Recherche des Notes Correspondantes à une Valeur Spécifique** : `java System.out.println("Entrez une note : "); int note = scanner.nextInt(); System.out.println("Nombre d'etudiants ayant la note " + note + " : " + notes.getRepeatedNote(note));`
   - L'utilisateur est invité à entrer une note spécifique.
   - La méthode `getRepeatedNote` est appelée pour trouver et afficher combien d'étudiants ont cette note spécifique.
10. **Fermeture du Scanner** : `java scanner.close();`
    - L'objet `Scanner` est fermé pour libérer les ressources.
    - En résumé, la classe `Main` et la classe `Notes` travaillent ensemble pour lire les notes des étudiants, les trier et les afficher, calculer des

statistiques telles que la moyenne, la note minimale et maximale, et compter combien de fois une note spécifique apparaît.

- Exercice 2

Déclaration de Package

```
package org.example.ex2;
```

Cette ligne définit le package `org.example.ex2` dans lequel se trouve la classe `Conjugaison`. Les packages sont utilisés pour regrouper les classes liées, fournissant un moyen d'organiser le code et d'éviter les conflits de noms.

Déclaration de la Classe

```
public class Conjugaison {
```

Cette ligne déclare une classe publique nommée `Conjugaison`. Elle est accessible depuis d'autres classes.

Variable Privée

```
private final String verb;
```

Cette ligne déclare une variable d'instance privée et finale nommée `verb` de type `String`. Le mot-clé `final` indique que cette variable ne peut être assignée qu'une seule fois, ce qui signifie qu'elle est effectivement une constante au sein de l'instance de la classe.

Constructeur

```
public Conjugaison(String verb){
    this.verb = verb.trim().toLowerCase();
}
```

Ceci est le constructeur de la classe `Conjugaison`. Il prend un paramètre `String` nommé `verb`. À l'intérieur du constructeur, il assigne la version ajustée (en supprimant les espaces en début et en fin) et mise en minuscules de `verb` à la variable d'instance `verb`.

Méthode `conjuge`

```
public void conjuge() {
    if (verb.endsWith("er") && verb.length() > 2) {
        String verbWithoutER = verb.substring(0, verb.length() - 2);
        System.out.println("je " + verbWithoutER + "e");
        System.out.println("tu " + verbWithoutER + "es");
        System.out.println("il " + verbWithoutER + "e");
```

```
            System.out.println("nous " + verbWithoutER + "ons");
            System.out.println("vous " + verbWithoutER + "ez");
            System.out.println("ils " + verbWithoutER + "ent");
        } else {
            System.out.println("Le verbe ne pas valide.");
        }
    }
```

Explication de la Méthode `conjuge`

Cette méthode est utilisée pour conjuguer les verbes français qui se terminent par "er".

1. **Vérification de Condition :**

   ```
   if (verb.endsWith("er") && verb.length() > 2) {
   ```

   Cela vérifie si le `verb` se termine par "er" et a plus de 2 caractères. Si les deux conditions sont vraies, il procède à la conjugaison du verbe ; sinon, il imprime un message d'erreur ("Le verbe ne pas valide.").

2. **Conjugaison du Verbe :**

   ```
   String verbWithoutER = verb.substring(0, verb.length() - 2);
   ```

   Cette ligne enlève le "er" à la fin du `verb` pour former la racine du verbe (`verbWithoutER`).

3. **Impression des Conjugaisons :**

   ```
   System.out.println("je " + verbWithoutER + "e");
   System.out.println("tu " + verbWithoutER + "es");
   System.out.println("il " + verbWithoutER + "e");
   System.out.println("nous " + verbWithoutER + "ons");
   System.out.println("vous " + verbWithoutER + "ez");
   System.out.println("ils " + verbWithoutER + "ent");
   ```

   Ces lignes impriment les formes conjuguées du verbe pour différents pronoms : "je," "tu," "il," "nous," "vous," et "ils."

Résumé

Cette classe (`Conjugaison`) est conçue pour conjuguer les verbes français se terminant par "er". La méthode `conjuge` vérifie si le verbe est valide pour la conjugaison (se termine par "er" et a plus de 2 caractères) et imprime ensuite les formes conjuguées pour chaque pronom si c'est le cas. Si le verbe n'est pas valide, un message d'erreur est imprimé.

Classe Principale (`Main.java`)

La classe `Main` est le point d'entrée de l'application :

```java
package org.example.ex2;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Entrer un verbe : ");
        String verb = scanner.nextLine();
        Conjugaison conjugaison = new Conjugaison(verb);
        conjugaison.conjuge();
        scanner.close();
    }
}
```

1. **Déclaration du Package** :

   ```java
   package org.example.ex2;
   ```

   Ceci spécifie que la classe fait partie du package `org.example.ex2`.

2. **Imports** :

   ```java
   import java.util.Scanner;
   ```

   La classe `Scanner` du package `java.util` est importée pour faciliter la saisie utilisateur.

3. **Méthode Principale** :

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

   Ceci est le point d'entrée de l'application.

4. **Initialisation du Scanner** :

   ```java
   Scanner scanner = new Scanner(System.in);
   ```

   Un objet `Scanner` est créé pour lire la saisie utilisateur depuis la console.

5. **Demande de Saisie** :

   ```java
   System.out.println("Entrer un verbe : ");
   ```

   Un message est imprimé pour inviter l'utilisateur à entrer un verbe.

6. **Lecture de la Saisie** :

   ```java
   String verb = scanner.nextLine();
   ```

   La saisie utilisateur (verbe) est lue et stockée dans une variable.

7. **Création de l'Objet Conjugaison** :

   ```java
   Conjugaison conjugaison = new Conjugaison(verb);
   ```

Un objet `Conjugaison` est créé en utilisant le verbe fourni par l'utilisateur.

8. **Conjugaison du Verbe** :

   ```
   conjugaison.conjuge();
   ```

   La méthode `conjuge` de la classe `Conjugaison` est appelée pour conjuguer le verbe.

9. **Fermeture du Scanner** :

   ```
   scanner.close();
   ```

Ce code démontre bien l'utilisation des classes et des méthodes pour accomplir une tâche simple de conjugaison de verbe. Si vous avez d'autres questions ou besoin d'explications plus détaillées, n'hésitez pas à demander!

Exercice 4

Déclaration du package

```
package org.example.ex3;
```

Cette ligne déclare que la classe `MyString` appartient au package `org.example.ex3`.

Déclaration de la classe

```
public class MyString {
```

Cette ligne déclare une classe publique nommée `MyString`.

Champ statique privé

```
private static String string;
```

Un champ statique privé nommé `string` de type `String` est déclaré. Étant `static`, cela signifie qu'il n'y a qu'une seule instance de ce champ partagée entre toutes les instances de la classe `MyString`.

Méthode : `saisir`

```
public static void saisir(String s){
    string = s;
}
```

Cette méthode statique `saisir` prend un paramètre `String s` et l'assigne au champ statique `string`.

Méthode : `afficher`

```
public static void afficher(){
    System.out.println(string);
}
```

La méthode statique `afficher` imprime la valeur du champ statique `string` sur la console.

Méthode : `inverser`

```
public static String inverser(){
    StringBuilder sb = new StringBuilder(string);
    sb.reverse();
    string = sb.toString();
    return string;
}
```

La méthode statique `inverser` : 1. Crée un objet `StringBuilder` initialisé avec la valeur de `string`. 2. Inverse le contenu du `StringBuilder`. 3. Convertit le `StringBuilder` en `String` et le réassigne au champ statique `string`. 4. Retourne la chaîne inversée `string`.

Méthode : `nombreDeMots`

```
public static int nombreDeMots(){
    return string.split(" ").length;
}
```

La méthode statique `nombreDeMots` : 1. Divise la chaîne `string` en un tableau de sous-chaînes basé sur les espaces (" "). 2. Retourne la longueur du tableau résultant, qui représente le nombre de mots dans la chaîne `string`.

Résumé

La classe `MyString` fournit les fonctionnalités suivantes : 1. Stocker une chaîne (`méthode saisir`). 2. Afficher la chaîne (`méthode afficher`). 3. Inverser la chaîne (`méthode inverser`). 4. Compter le nombre de mots dans la chaîne (`méthode nombreDeMots`).

Voici comment vous pourriez utiliser ces méthodes :

```
public class Main {
    public static void main(String[] args) {
        MyString.saisir("Hello World");
        MyString.afficher();        // Imprime: Hello World
        String reversed = MyString.inverser();
        System.out.println(reversed); // Imprime: dlroW olleH
        int wordCount = MyString.nombreDeMots();
```

```
        System.out.println(wordCount); // Imprime: 2
    }
}
```

Cet exemple montre comment stocker une chaîne, l'afficher, l'inverser et compter le nombre de mots qu'elle contient.

Déclaration du package

```
package org.example.ex3;
```

Cette ligne déclare que la classe `Menu` appartient au package `org.example.ex3`.

Déclaration de la classe

```
public class Menu {
```

Cette ligne déclare une classe publique nommée `Menu`.

Méthode : `afficherMenu`

```
public static void afficherMenu() {
    System.out.println("************** Menu *************");
    String[] operations = {"saisir", "afficher", "inverser", "nombre de mots" , "Fin"};
    for (int i = 0; i < operations.length; i++) {
        System.out.printf("%d : %s%n", i + 1, operations[i]);
    }
}
```

La méthode statique `afficherMenu` : 1. Affiche une ligne "************** Menu *************" sur la console. 2. Crée un tableau de chaînes de caractères `operations` contenant les différentes options du menu. 3. Utilise une boucle `for` pour parcourir les éléments du tableau `operations`. 4. À chaque itération de la boucle, il affiche l'index de l'opération (commençant à 1) suivi de l'opération elle-même. Le format d'affichage est géré par `System.out.printf` avec le spécificateur de format `%d : %s%n`, où `%d` est l'index de l'élément, `%s` est l'élément lui-même et `%n` est un saut de ligne.

Résumé

La classe `Menu` fournit une seule fonctionnalité : - Afficher un menu avec différentes options (`méthode afficherMenu`).

Voici comment vous pourriez utiliser cette méthode :

```
public class Main {
    public static void main(String[] args) {
        Menu.afficherMenu();
```

```
    }
}
```

Cet exemple montre comment appeler la méthode `afficherMenu` pour afficher un menu sur la console avec plusieurs options comme “saisir”, “afficher”, “inverser”, “nombre de mots” et “Fin”.

Déclaration du package

```
package org.example.ex3;
```

Cette ligne déclare que la classe `Main` appartient au package `org.example.ex3`.

Importation de la classe Scanner

```
import java.util.Scanner;
```

Cette ligne importe la classe `Scanner` du package `java.util`, utilisée pour lire les entrées de l'utilisateur à partir de la console.

Déclaration de la classe

```
public class Main {
```

Cette ligne déclare une classe publique nommée `Main`.

Méthode principale : `main`

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

Cette méthode `main` est le point d'entrée de l'application.

Variables et initialisation

```
Scanner scanner = new Scanner(System.in);
int choix;
```

- `scanner`: Un objet `Scanner` est créé pour lire les entrées de l'utilisateur à partir de la console.
- `choix`: Une variable entière est déclarée pour stocker le choix de l'utilisateur.

Boucle `do-while`

```
do {
```

La boucle `do-while` continue d'exécuter le bloc de code tant que `choix` n'est pas égal à `5`.

Affichage du menu

```
Menu.afficherMenu();
```

La méthode `afficherMenu` de la classe `Menu` est appelée pour afficher les options du menu.

Lecture du choix de l'utilisateur

```
System.out.println("Entrez votre choix :");
choix = scanner.nextInt();
scanner.nextLine();
```

Un message demande à l'utilisateur de saisir son choix. Le choix est lu en tant qu'entier avec `nextInt` et une ligne vide est lue après pour gérer la fin de ligne avec `nextLine`.

Instruction `switch-case`

```
switch (choix) {
    case 1:
        System.out.print("Saisir : ");
        String s = scanner.nextLine();
        System.out.println(s);
        MyString.saisir(s);
        break;
    case 2:
        MyString.afficher();
        break;
    case 3:
        System.out.println("Inverse : " + MyString.inverser());
        break;
    case 4:
        System.out.println("Nombre de mots : " + MyString.nombreDeMots());
        break;
    case 5:
        System.out.println("Fin.");
        break;
    default:
        System.out.println("Choix incorrect");
        break;
}
```

En fonction de la valeur de `choix` : - `case 1`: Demande à l'utilisateur de saisir une chaîne, l'affiche, puis appelle la méthode `saisir` de `MyString`. - `case 2`: Appelle la méthode `afficher` de `MyString`. - `case 3`: Appelle la méthode `inverser` de `MyString` et affiche le résultat inversé. - `case 4`: Appelle la méthode `nombreDeMots` de `MyString` et affiche le nombre de mots. - `case 5`:

Affiche “Fin.” et termine la boucle. - `default`: Affiche “Choix incorrect” pour tout choix non valide.

Pause avant de revenir au menu

```
if (choix != 5) {
    System.out.println("\nFrappez une touche pour revenir au menu...");
    scanner.nextLine();
}
```

Si le choix n’est pas `5`, demande à l’utilisateur de frapper une touche pour revenir au menu.

Fermeture de la boucle `do-while`

```
} while (choix != 5);
```

La boucle continue tant que le choix n’est pas `5`.

Fermeture du scanner

```
scanner.close();
```

Ferme l’objet `Scanner` pour libérer les ressources.

Résumé

La classe `Main` implémente un simple menu interactif pour interagir avec la classe `MyString`: - Affiche un menu. - Lit le choix de l’utilisateur. - Exécute l’action correspondante (saisir, afficher, inverser, compter les mots ou terminer).

Voici comment le menu pourrait être utilisé :

```
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choix;

        do {
            Menu.afficherMenu();
            System.out.println("Entrez votre choix :");
            choix = scanner.nextInt();
            scanner.nextLine();
            switch (choix) {
                case 1:
                    System.out.print("Saisir : ");
                    String s = scanner.nextLine();
                    System.out.println(s);
                    MyString.saisir(s);
```

```
                    break;
                case 2:
                    MyString.afficher();
                    break;
                case 3:
                    System.out.println("Inverse : " + MyString.inverser());
                    break;
                case 4:
                    System.out.println("Nombre de mots : " + MyString.nombreDeMots());
                    break;
                case 5:
                    System.out.println("Fin.");
                    break;
                default:
                    System.out.println("Choix incorrect");
                    break;
            }

            if (choix != 5) {
                System.out.println("\nFrappez une touche pour revenir au menu...");
                scanner.nextLine();
            }

        } while (choix != 5);

        scanner.close();
    }
}
```

Cet exemple montre comment l'utilisateur peut interagir avec le menu pour effectuer diverses actions sur une chaîne de caractères.

Exercice 4

Déclaration du package

```
package org.example.ex4;
```

Cette ligne déclare que la classe `OccurrencesLettres` appartient au package `org.example.ex4`.

Déclaration de la classe

```
public class OccurrencesLettres {
```

Cette ligne déclare une classe publique nommée `OccurrencesLettres`.

Champ privé final

```
private final String text;
```

Un champ privé et final nommé `text` de type `String` est déclaré. `final` signifie que cette variable ne peut être assignée qu'une seule fois.

Constructeur

```
public OccurrencesLettres(String text) {
    this.text = text.toLowerCase();
}
```

Le constructeur de la classe prend un paramètre `String text` et l'assigne au champ `text` en convertissant tous les caractères en minuscules (`toLowerCase`).

Méthode : `calc`

```
public void calc(){
    String alphabets = "abcdefghijklmnopqrstuvwxyz";
    int[] nb_occurrences = new int[26];
    for (int i = 0; i < text.length(); i++) {
        char c = text.charAt(i);
        if (alphabets.contains(String.valueOf(c))) {
            nb_occurrences[alphabets.indexOf(String.valueOf(c))]++;
        }
    }

    for (int i = 0; i < nb_occurrences.length; i++) {
        if (nb_occurrences[i] > 0) {
            System.out.printf("%c : %d%n", alphabets.charAt(i) , nb_occurrences[i]);
        }
    }
}
```

Explication

1. **Initialisation des alphabets :** `java String alphabets = "abcdefghijklmnopqrstuvwxyz";` Cette ligne initialise une chaîne contenant toutes les lettres de l'alphabet en minuscules.
2. **Initialisation du tableau de comptage :** `java int[] nb_occurrences = new int[26];` Un tableau d'entiers de taille 26 est créé pour compter les occurrences de chaque lettre de l'alphabet.
3. **Comptage des occurrences :** `java for (int i = 0; i < text.length(); i++) { char c = text.charAt(i); if (alphabets.contains(String.valueOf(c))) { nb_occurrences[alphabets.index } } }`

- Une boucle `for` parcourt chaque caractère de `text`.
- `char c = text.charAt(i);` extrait le caractère à la position `i`.
- `if (alphabets.contains(String.valueOf(c)))` vérifie si le caractère `c` est une lettre de l'alphabet.
- Si oui, `nb_occurrences[alphabets.indexOf(String.valueOf(c))]++` incrémente le compteur correspondant à cette lettre.

4. **Affichage des résultats :** `java for (int i = 0; i < nb_occurrences.length; i++) { if (nb_occurrences[i] > 0) { System.out.printf("%c : %d%n", alphabets.charAt(i) , nb_occurrences[i]); } }`

   - Une seconde boucle `for` parcourt les éléments du tableau `nb_occurrences`.
   - `if (nb_occurrences[i] > 0)` vérifie si la lettre a été trouvée au moins une fois.
   - Si oui, `System.out.printf("%c : %d%n", alphabets.charAt(i) , nb_occurrences[i]);` affiche la lettre et son nombre d'occurrences.

Résumé

La classe `OccurrencesLettres` permet de : - Initialiser un texte en minuscules. - Calculer et afficher le nombre d'occurrences de chaque lettre de l'alphabet présent dans le texte.

Voici comment vous pourriez utiliser cette classe :

```
public class Main {
    public static void main(String[] args) {
        OccurrencesLettres occurrences = new OccurrencesLettres("Hello World");
        occurrences.calc();
    }
}
```

Cet exemple montre comment instancier la classe `OccurrencesLettres` et appeler la méthode `calc` pour afficher le nombre d'occurrences de chaque lettre dans le texte "Hello World".

Déclaration du package

```
package org.example.ex4;
```

Cette ligne déclare que la classe `Main` appartient au package `org.example.ex4`.

Importation de la classe Scanner

```
import java.util.Scanner;
```

Cette ligne importe la classe `Scanner` du package `java.util`, utilisée pour lire les entrées de l'utilisateur à partir de la console.

Déclaration de la classe

```
public class Main {
```

Cette ligne déclare une classe publique nommée `Main`.

Méthode principale : `main`

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

Cette méthode `main` est le point d'entrée de l'application.

Initialisation du Scanner

```
Scanner scanner = new Scanner(System.in);
```

Un objet `Scanner` est créé pour lire les entrées de l'utilisateur à partir de la console.

Demande de saisie de texte

```
System.out.println("Entrez le texte :");
```

Affiche un message demandant à l'utilisateur de saisir un texte.

Création d'une instance d'OccurrencesLettres

```
OccurrencesLettres occurrencesLettres = new OccurrencesLettres(scanner.nextLine());
```

- Lis le texte saisi par l'utilisateur avec `scanner.nextLine()`.
- Crée une instance de `OccurrencesLettres` en passant le texte saisi au constructeur.

Calcul et affichage des occurrences des lettres

```
occurrencesLettres.calc();
```

Appelle la méthode `calc` de l'objet `OccurrencesLettres` pour calculer et afficher le nombre d'occurrences de chaque lettre dans le texte.

Fermeture du Scanner

```
scanner.close();
```

Ferme l'objet `Scanner` pour libérer les ressources.

Résumé

La classe `Main` permet à l'utilisateur de : - Saisir un texte par l'intermédiaire de la console. - Créer une instance de `OccurrencesLettres` avec le texte saisi. - Calculer et afficher le nombre d'occurrences de chaque lettre présente dans le texte.

Voici comment l'application pourrait être utilisée :

```java
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Entrez le texte :");
        OccurrencesLettres occurrencesLettres = new OccurrencesLettres(scanner.nextLine());
        occurrencesLettres.calc();
        scanner.close();
    }
}
```

Cet exemple montre comment l'utilisateur peut saisir un texte, puis calculer et afficher les occurrences de chaque lettre dans ce texte, en utilisant la classe `OccurrencesLettres`.

Explication du code Java

Déclaration du package

```java
package org.example;
```

Cette ligne déclare que la classe `Main` appartient au package `org.example`.

Déclaration de la classe

```java
public class Main {
```

Cette ligne déclare une classe publique nommée `Main`.

Méthode principale : `main`

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

La méthode `main` est le point d'entrée de l'application Java. Elle est public, statique, ne retourne rien (`void`), et prend un tableau de chaînes de caractères (`String[] args`) comme argument.

Affichage du message et appel des méthodes `main` externes

```java
System.out.println("Exercice 1");
org.example.ex1.Main.main(args);
System.out.println("Exercice 2");
org.example.ex2.Main.main(args);
System.out.println("Exercice 3");
org.example.ex3.Main.main(args);
System.out.println("Exercice 4");
org.example.ex4.Main.main(args);
```

1. **Affichage et appel pour `Exercice 1`** : `java System.out.println("Exercice 1"); org.example.ex1.Main.main(args);`
   - `System.out.println("Exercice 1");` : Affiche “Exercice 1” sur la console.
   - `org.example.ex1.Main.main(args);` : Appelle la méthode `main` de la classe `Main` du package `org.example.ex1`, en passant les arguments `args`.
2. **Affichage et appel pour `Exercice 2`** : `java System.out.println("Exercice 2"); org.example.ex2.Main.main(args);`
   - `System.out.println("Exercice 2");` : Affiche “Exercice 2” sur la console.
   - `org.example.ex2.Main.main(args);` : Appelle la méthode `main` de la classe `Main` du package `org.example.ex2`, en passant les arguments `args`.
3. **Affichage et appel pour `Exercice 3`** : `java System.out.println("Exercice 3"); org.example.ex3.Main.main(args);`
   - `System.out.println("Exercice 3");` : Affiche “Exercice 3” sur la console.
   - `org.example.ex3.Main.main(args);` : Appelle la méthode `main` de la classe `Main` du package `org.example.ex3`, en passant les arguments `args`.
4. **Affichage et appel pour `Exercice 4`** : `java System.out.println("Exercice 4"); org.example.ex4.Main.main(args);`
   - `System.out.println("Exercice 4");` : Affiche “Exercice 4” sur la console.
   - `org.example.ex4.Main.main(args);` : Appelle la méthode `main` de la classe `Main` du package `org.example.ex4`, en passant les arguments `args`.

Résumé

Cette classe `Main` dans le package `org.example` agit comme un point d’entrée central pour exécuter les méthodes `main` de différentes classes `Main` situées dans d’autres packages : - `org.example.ex1` - `org.example.ex2` - `org.example.ex3` - `org.example.ex4`

Pour chacune de ces classes, le message “Exercice X” est affiché, suivi de l’exécution de leur méthode `main` respective. Cela permet d’organiser et d’exécuter des exercices ou des modules séparés dans différentes packages, tout en ayant un seul point d’entrée pour tous.

Voici le code commenté pour mieux comprendre :

```
package org.example;

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

```
        System.out.println("Exercice 1");
        // Exécute le main de org.example.ex1.Main avec les mêmes arguments
        org.example.ex1.Main.main(args);

        System.out.println("Exercice 2");
        // Exécute le main de org.example.ex2.Main avec les mêmes arguments
        org.example.ex2.Main.main(args);

        System.out.println("Exercice 3");
        // Exécute le main de org.example.ex3.Main avec les mêmes arguments
        org.example.ex3.Main.main(args);

        System.out.println("Exercice 4");
        // Exécute le main de org.example.ex4.Main avec les mêmes arguments
        org.example.ex4.Main.main(args);
    }
}
```

J'espère que cela clarifie le fonctionnement de ce code !