Exercice 01

Traitements sur une phrase

Un premier exercice consistant à réaliser des traitements élémentaires sur une phrase.

1. Enoncé

Une phrase est une chaine de caractères dont les mots alphabétiques sont séparés par des caractères blancs ou par les caractères de ponctuations « , ; . -» (virgule, point-virgule, point, trait d'union). Pour simplifier, une phrase ne contient pas de caractères accentués.

On se propose de faire un programme Java qui permet de faire les traitements suivants sur une phrase :

- Déterminer si la phrase est un palindrome. Exemple : « Tu l as trop ecrase, Cesar, ce port-salut » (on ne traite pas le cas des caractères accentués)
- Rechercher un mot dans la phrase
- Déterminer le nombre d'occurrences d'un caractère dans la phrase.
- Concaténer deux phrases.

Le programme s'exécute dans la console.

2. Enoncé plus précis

Faire la classe **Phrase** qui permet de réaliser les méthodes suivantes. Toutes ces méthodes sont testées à partir de la classe Exercice01 qui contient la méthode **main** permettant d'exécuter le programme de test.

Le programme main doit permettre de tester chacun des traitements publiques.

La classe Phrase contient :
- un attribut privé **phrase** de type String.
- un attribut privé **mots** de type ArrayList qui contient tous les mots de la phrase.

2.1. Saisir chaine

Cette méthode est une méthode statique dont le rôle est d'afficher dans la console un texte d'invitation et de saisir au clavier statique chaine de caractère.

public static String SaisirChaine(String invitation) throws IOException

Rappel : l'instruction Java qui permet de saisir une chaine peut être :

```
   String s= new BufferedReader(new InputStreamReader (System.in))
.readLine()
```

Comme on va avoir besoin dans la méthode **main** de saisir plusieurs fois une chaine de caractère, ce sera plus pratique d'utiliser cette méthode statique.

Rappel : pour afficher une chaine de caractère dans la console, il suffit d'exécuter l'instruction suivante : `System.out.println(str)`. Inutile de faire une méthode statique pour si peu.

2.2. Les mots de la phrase

Faire la méthode privée **private void isolerMots()** qui renseigne l'attribut **mots**.

Cet attribut contient les mots de la phrase (quelque soit les caractères blancs ou ponctuations se trouvant en début, à la fin ou entre les mots).

Utilisez pour cela la classe *StringTokenizer* et la classe *ArrayList*.

Cette méthode privée sera appelée dans le constructeur.

2.3. Constructeur

Faire le constructeur **Phrase(String phrase)** qui initialise tous les attributs de la classe.

Le constructeur convertit tous les caractères de la phrase en minuscule et fait de suite l'extraction des mots (appel à la méthode **isolerMots**).

2.4. Palindrome

Faire la méthode objet qui détermine si **phrase** est un palindrome.

public boolean palindrome()

Exemple : « Tu l'as trop ecrase, Cesar, ce port-salut » (on ne traite pas le cas des caractères accentués)

Rappel, si besoin : la méthode `char charAt(i)` de la classe String retourne le i-ième caractère d'une chaine. La méthode `int length()` de la classe String retourne le nombre de caractères d'une chaine. L'opérateur `+` permet de concaténer des caractères et/ou des chaines. La méthode `Character.isLetter(char ch)` détermine si ch est

une lettre alphabétique. La méthode `String toLowerCase()` convertit une chaine en minuscule.

2.5. Rechercher un mot

Faire la méthode qui recherche un mot dans la phrase. Les caractères de ponctuation « , ;. -» et le caractère blanc séparent les mots.

public boolean rechercher(Strint motAChercher)

2.6. Occurrences des lettres

Faire la méthode qui calcule les occurrences des lettres de l'alphabet ('a' à 'z').

public void calculerOccurences()

Faire la méthode qui retourne le nombre d'occurrence d'une lettre de l'alphabet dans la phrase.

public int getOccurences(char ch)

2.7. Concaténation de deux phrases

Faire la méthode qui permet de concaténer 2 phrases (de type Phrase).