Date : Décembre 2020
Enseignants : Marie-Laure Nivet
Diplôme : L3 SPI Informatique
Nom de l'UE : Programmation Orientée Objet, Langage Java
Type de document : TP
Date de rendu : Vendredi 8 Janvier 23h59 heures sur un dépôt Git que vous renseignerez sur le fichier partagé dans l'espace dédié au cours sur Teams (Fichiers/BinomesTPJava.xlsx).

Interface, Collections et types génériques

Objectifs :

Manipuler les interfaces, les collections d'objets en Java et découvrir les principes sous-jacents en particulier la généricité. Vous allez devoir plus que jamais aller piocher dans la documentation du JDK proposée par Oracle.

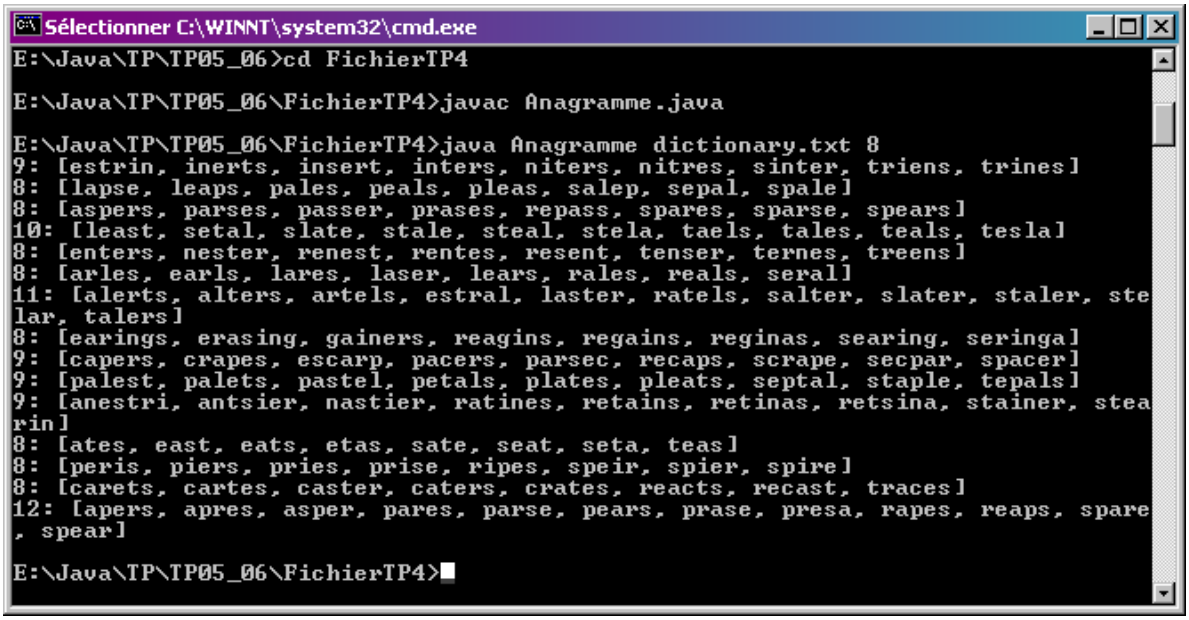
1. Création d'une classe utilisant les types génériques

Pour ceux qui n'auraient pas bien réussi l'exercice du TD sur les triplets… Celui-ci c'est la même chose mais en plus facile !
On vous demande de créer une classe générique permettant de manipuler des paires d'objets de types hétérogènes. C'est-à-dire que vos paires doivent pouvoir par exemple accueillir en premier élément une chaîne de caractère et en deuxième élément une `Personne`, ou bien, en premier élément un objet `Integer` et en deuxième élément une instance de `GregorianCalendar`.

1. Ecrivez un code minimal pour cette classe, à savoir, une classe possédant un constructeur à deux argument (les deux éléments de la paire), les accesseurs et modificateurs nécessaires.
2. Ecrivez un code de Test pour votre classe, créez une liste de paires, remplissez là et testez sur les instances contenues toutes les méthodes définies.

2. Anagramme et MultiMap…

Ecrivez un programme prenant en paramètre le nom d'un fichier de mot (un mot par ligne) ainsi qu'un nombre entier qui recherche et affiche tous les anagrammes présents dans le fichier à partir du moment ou il y en a plus d'un certain nombre (l'entier passé en paramètre).
Rappel : les anagrammes sont des mots composés des mêmes lettres mais apparaissant dans un ordre différents, ainsi arbre et barre sont des anagrammes.
Voici la sortie écran obtenue suite à l'analyse du fichier dictionary.txt, mis à votre disposition sur l'ENT et/ou téléchargeable à l'adresse https://docs.oracle.com/javase/tutorial/collections/interfaces/examples/dictionary.txt. On obtient la liste des anagrammes issues de ce fichier en nombre supérieur à 8.



Pour lire le fichier et en récupérer le contenu je vous conseille d'utiliser la classe java.util.Scanner[1]. Pour vous aider dans la lecture des mots contenus dans le fichier regardez le code ci-dessous, il utilise un Scanner pour lire le contenu d'un fichier nommé text.txt.

```
import java.util.* ;
import java.io.*;
public class Test{
    public static void main(String[] args){
      try{
          Scanner scan = new Scanner(new File("text.txt"));
          while(scan.hasNext())System.out.print(scan.next()+" ");
      }catch(FileNotFoundException e){
          System.err.println("Le fichier n'a pas ete trouve...");
      }
    }
}
```

```
>java Test
Ceci est un exemple
```

Sachant que text.txt est placé dans le répertoire courant et contient :

```
Ceci
est
un
exemple
```

Pour ceux qui n'ont pas idée de la marche à suivre, vous trouverez des pistes sur les transparents du cours illustrant les principes des Map et des multi-Maps.

3. Listes triées, d'après Henri Garreta

Écrivez un programme qui construit une collection triée contenant n nombres entiers (représentés par des objets `Integer`, profitez en pour allez voir le concept d'AutoBoxing…) tirés au hasard (Allez voir du côté des Random) dont la valeur est comprise entre 0 et 1000. A votre choix, la valeur de n est lue au début de l'exécution ou est un argument du programme. Ensuite, ce dernier affiche la collection construite afin qu'on puisse constater qu'elle est bien triée.

[1] Documentation à l'adresse http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html

Dans la première version du programme la collection est une sorte de `List` que vous triez après la construction, en utilisant une méthode ad hoc de la classe `Collection`**`s`**.

Dans une deuxième version, la collection est une sorte de `Set` qui vous choisirez pour qu'elle soit constamment triée.

4. Faire des comparaisons (cf. transparents de fin du cours)

Ecrire un programme qui prend des arguments (des mots) sur la ligne de commande et les affiches :

1 Dans l'ordre lexicographique (ordre du dictionnaire).

2 Dans l'ordre militaire (on compare d'abord la taille des deux chaînes avant d'utiliser l'ordre lexicographique ; par exemple, tb, tau et tata sont dans l'ordre militaire).

3 Dans l'ordre inverse de l'ordre lexicographique.

Une fois les trois ordres implantés, on souhaite modifier le programme pour qu'il prenne en premier argument une chaîne de caractères identifiant l'affichage à choisir.

Exemple :

```
java Comparaison cette chaine avec quelques autres toutes differentes

args avant tri:[cette, chaine, avec, quelques, autres, toutes, differentes]
tri lexico:[autres, avec, cette, chaine, differentes, quelques, toutes]
tri militaire:[avec, cette, autres, chaine, toutes, quelques, differentes]
tri inverse lexico:[toutes, quelques, differentes, chaine, cette, avec, autres]
```

Indices : Transparents de la fin du cours, interface Comparable et interface Comparator…

5. Faire des comparaisons sur les Personnes

Faîtes les modifications nécessaires sur votre classe Personne pour permettre le tri d'une Collection de Personne en fonction d'abord de leur âge puis de l'ordre alphabétique de leur nom et prénom…
Une fois cette nouvelle possibilité ajoutée n'oubliez pas de la tester !!!!

Indices : Transparents de la fin du cours, interface Comparable et interface Comparator…