ISEN Yncréa Ouest – M1 Brest
Langage Java - TP évalué
Durée 3 heures - Documents autorisés

Bien lire tout l'énoncé avant de commencer !

Ciel ! Mes aïeux…

1. Introduction

Nous nous intéressons dans cet exercice au traitement de données généalogiques contenues dans un fichier au format gedcom. GEDCOM est abréviation de **GE**nealogical **D**ata **COM**munication. La première partie concerne la lecture du fichier et l'affichage de quelques données dans la console. La seconde partie doit permettre de trier ces données via une interface graphique.

Pour ceux qui veulent en savoir plus sur le format gedcom :
https://fr.geneawiki.com/index.php/Norme_Gedcom
https://fr.wikipedia.org/wiki/GEDCOM

Pour faciliter votre tâche, la classe Personne (voir annexe) ainsi que le fichier geneafinder.ged vous sont fournis. N'hésitez-pas à consulter ce fichier avec un éditeur de texte pour mieux comprendre !

2. Travail à réaliser

2.1. Première partie : un peu d'algorithmique…

Créer une classe java « Principale.java » qui contiendra votre méthode « main ».
Il faut extraire du fichier gedcom les données suivantes :

- nom,
- prénoms,
- sexe,
- année de naissance,
- année de décès.

Attention, ces informations ne sont pas toujours présentes pour tous les individus. On ne traitera que les individus pour lesquels toutes ces données sont disponibles.

La balise **INDI** marque le début des informations concernant un individu.
Exemple : 0 @I7@ INDI
La ligne commence toujours par le chiffre 0.

Les balises **GIVN** et **SURN** concernent, respectivement, les prénoms et le nom de l'individu.
Exemple : 2 GIVN Eugénie Henriette
2 SURN BANNER
Ces lignes commencent toujours par le chiffre 2.

La balise **SEX** précise le genre de l'individu : Masculin (M) ou Féminin (F).
Exemple : 1 SEX F
Cette ligne commence toujours par le chiffre 1.

Les balises **BIRT** et **DEAT** introduisent, respectivement, les dates de naissance et de décès de l'individu. Elles sont suivies de la balise DATE si celle-ci est connue !
Exemple : 1 BIRT
2 DATE 27 FEB 1884
1 DEAT
2 DATE 10 MAR 1916
Seuls les quatre derniers caractères nous intéressent pour calculer ultérieurement l'âge au décès (voir la classe Personne).
Attention :

- l'ordre BIRT puis DEAT n'est pas toujours respecté ;
- il peut ne pas y avoir de date mais seulement la balise PLAC qui décrit le lieu de l'évènement.

Un conseil pour simplifier votre travail : copier chaque ligne du fichier dans une liste puis faire les traitements sur cette liste.

Pour lire le fichier, utiliser la classe Scanner :

```
File doc = new File("geneafinder.ged");
Scanner obj = new Scanner(doc, "UTF-8");
```

Lire chaque ligne du fichier et l'enregistrer dans une liste de chaines de caractères avec la méthode add(…) de la classe ArrayList :

```
List<String> gedcomData = new ArrayList<String>();
…
while (obj.hasNextLine()) {
        gedcomData.add(new String(obj.nextLine()));
}
```

Utiliser la méthode get(…) de la classe ArrayList pour parcourir la liste.
Pensez aussi à utiliser les méthodes de la classe String pour vos traitements : startsWith(…), substring(…), contains(…) etc.[1]

[1] N'oubliez-pas que vous avez accès à la documentation Java, par exemple : https://docs.oracle.com/javase/8/docs/api/index.html

Quand vous avez trouvé toutes les informations concernant un individu, créer un objet instance de la classe Personne et insérer cet objet dans une liste de Personne :

```
List<Personne> tribu = new ArrayList<Personne>();
…
tribu.add( new Personne(sexe, prenoms, nom, naissance, deces) );
```

Lorsque c'est terminé, afficher le contenu de la liste de Personne dans la console.

2.2. Deuxième partie - une interface

Créer une classe « MyInterface.java » qui hérite de « JFrame » et contient votre interface ainsi que les traitements associés.
Votre interface doit proposer, **à minima**, les fonctionnalités suivantes :

- choisir le type de tri de données à réaliser selon l'âge au décès, l'année de naissance ou l'année du décès,
- choisir l'ordre du tri,
- afficher le résultat correspondant aux choix précités.

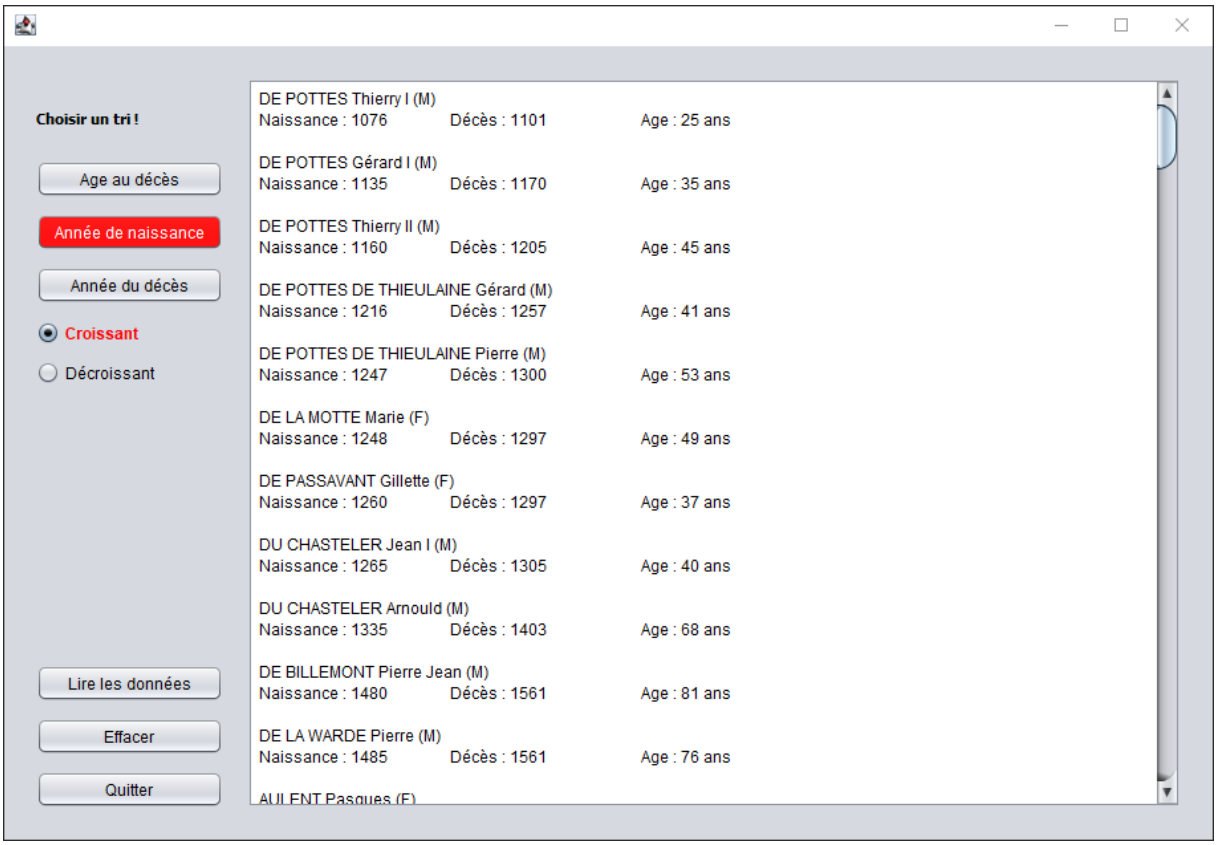

Figure 1. Exemple d'interface
Ceci n'est qu'un exemple !

Attention, version minimale n'est pas synonyme d'achèvement. Une partie de l'évaluation portera sur l'ajout de fonctionnalités non précisées dans ce cahier des charges. A vous d'imaginer et de mettre en œuvre…

Pour trier la liste de Personne c'est très simple.
On fait appel à la méthode sort(…) de la classe ArrayList. On utilise comme argument un appel à la méthode statique Comparator.comparing(…) dont l'argument ne sera autre que l'accesseur en lecture de l'attribut de la classe Personne sur lequel faire le tri.

Exemples :

```
List<Personne> tribu = new ArrayList<Personne>();
…
…
// un tri de la liste dans l'ordre croissant des années de naissance
tribu.sort(Comparator.comparing(Personne::getAnneeNaissance));
…
// un tri de la liste dans l'ordre décroissant des âges
tribu.sort(Comparator.comparing(Personne::getAge).reversed());
```

2.3. Compte-rendu

Dans votre rapport vous expliquerez votre raisonnement pour la mise en œuvre de votre application. Vous détaillerez également son architecture. N'oubliez pas de décrire vos tests !

En fin de TP, vous créerez un fichier d'archive regroupant vos codes sources ainsi que votre rapport. Vous déposerez ce fichier sur l'ENT dans la rubrique ad hoc.

Vos productions seront compilées puis testées sous l'environnement jGrasp par le correcteur.

Une attention particulière sera notamment portée sur :

- le respect du cahier des charges (i.e. le travail minimal à réaliser),
- l'ergonomie de votre application,
- l'ajout de fonctionnalités non précisées dans le cahier des charges,
- la pertinence des « concepts objets » mis en œuvre.

Conseils / Remarques

Si vous utilisez des ressources externes (i.e. sur internet) vous devez mentionner l'origine de ces ressources en référence dans votre rapport et en commentaire dans votre code. Toute omission sera sanctionnée.
On trouve sur internet de nombreux composants logiciels. Ne vous dispersez pas. Ne perdez pas de vue votre cahier des charges.

Annexe

```
public class Personne {

        // les attributs

          String sexe;
          String nom;
          String prenoms;
          int anneeNaissance, anneeDeces, age;

        // le constructeur

          Personne (String sexe, String nom, String prenoms, int anneeNaissance, int anneeDeces)
        {
             this.sexe = sexe;
             this.nom = nom;
             this.prenoms = prenoms;
             this.anneeNaissance = anneeNaissance;
             this.anneeDeces = anneeDeces;
             this.age =   this.anneeDeces - this.anneeNaissance;
          }

        // les accesseurs en lecture

          public String getSexe() {
            return(this.sexe);
          }

          public String getNom() {
            return(this.nom);
          }

          public String getPrenoms() {
            return(this.prenoms);
          }

          public int getAnneeNaissance() {
            return(this.anneeNaissance);
          }

          public int getAnneeDeces() {
            return(this.anneeDeces);
          }

          public int getAge() {
            return(this.age);
          }

        // autres méthodes

          public String toString() {
            return(this.prenoms + " " + this.nom
                      + " (" + this.sexe + ")\n"
                      + "Naissance : "+ this.anneeNaissance + "\t"
                      + "Décès : " + this.anneeDeces + "\t"
                      + "Age : " + this.age + " ans\n");
          }
}
```