LabWork JAVA 4

EEng3

You will do one project for the entire labwork. This project will contain the necessary classes, including the main class.

Objectives :
The database will be filled with real data from http://www.freebase.com/.
Retrieve on Espace.ecam.fr the files.

Retrieve on Espace.ecam.fr the file fileTP04.zip.

- Copy the three classes to the src folder of your project.
- The data folder must be copied to the root of your project.
- The json-simple-1.1.1.jar library must be added to your project :
  - Copy the json-simple-1.1.1.jar file to the root of your project.
  - Refresh the project (right click - refresh)
  - Right click on the file json-simple-1.1.1.jar and select "Build Path" / "Add to Build Path".

Be careful to name the variables, classes, methods exactly as proposed.

Part 1 : People and Movie

Test by creating two instances of People. Also test the toString, equals and compareTo methods.
What are the toString, equals and compareTo methods for?

Create the Movie class:
All methods will need to be created with Eclipse's automatic generation tools: Ask demo and explanation to your supervisor if needed. (Source / Generate).
The equals method will only take into account the title and the year

Movie
- title : String - director : People - actors : LinkedList<People> - year : integer
+ Movie(titleP : String, directorP : People, yearP : integer) + *All Get Method* + toString() : String + equals(Movie o)

Finalize the constructor.

Add a method addActor (actor: People) that adds a cast member as a parameter to the cast list. No verification will be done.

Test (Create a movie with at least 2 actors and a director).

Part 2 : Movies Integration

The populateBase() method of the ReadFileMovie class always returns an identical list of 4999 movies (and associated actors and directors).

The populateBaseRandom() method of the ReadFileMovie class returns a random list of approximately 5000 movies (and associated actors and directors) selected from 20000 available.

The displaySample (number) method displays 'number' movies in the list.

Test populateBase and populateBaseRandom by displaying 20 movies from the base.

Part 3 : Statistics

The goal of this part is to make a DVDlibrary class that will contain Movies.
Realize and test the proposed methods one by one.

DVDlibrary
- listMovies : list of Movie
+ DVDlibrary(listP : List of Movie) + researchYear(integer) : integer + nbMovieDirector(People) : integer

DVDlibrary(listP : List of Movie)
researchYear(integer) : returns the number of movies in a year passed as a parameter *(1997 : 98 / 1914 : 17 / 2005 : 246 / 2007 : 129)*
nbMovieDirector(People) : returns the number of movies made by a Person passed as a parameter. Attention: some films have no director (null) *(Tarantino : 3, Chaplin 18, Kubrick : 6)*

Part 4 : Duplicate

The Duplicate class contains a getPeopleList(ListPeople list) method that returns a list (LinkedList) of all the people in the list passed as a parameter, without duplicates (57107). The calculation time of this method is of the order of 120s.
The getAllPeople() method of the ReadFileMovie class returns all the people in the database (94776).
Test: Use System.currentTimeMillis (), which returns the time in milliseconds, to have the exact calculation time of the getPersonnesList () method.

Optimize this method by transforming the first loop by an iterator. What is the new calculation time?

Create getPersonsABR (): returns a binary search tree (TreeSet) (you have to search on google) of all the people in the list passed as a parameter, without duplicates. In Java, the TreeSet class does not handle duplicates. In practice, the add (element) method does not add the element if it is already present and crashes if it is null. What is the calculation time of this method?