CS-102: COMPUTING AND ALGORITHMS II ASSIGNMENT 1 - 2018 SPRING

GIUSEPPE TURINI - KETTERING UNIVERSITY

INTRODUCTION

In this programming assignment, you will demonstrate: your knowledge of material covered in CS-101, as well as your knowledge of *dynamic data structures* (in particular, *linked lists*), by creating a simple database system for tracking the *win/loss record* of a set of *tennis players*.

FILE INPUT

Your program will accept a single argument on the command line, which is a non-empty string giving the name of a file in the current directory. This file will contain descriptions of two different types of objects: *tennis players* and *tennis single matches.* [1]

Tennis players will be represented by several strings, separated by slashes, in the following order: [2]

1 The string `"PLAYER"`, indicating that this line represents a *tennis player*.

2 A string representing the *unique id* of the tennis player (e.g. `"FED81"`, `"DJO87"`, `"NAD86"`).

3 A string representing the *first name* of the tennis player (e.g. `"ROGER"`, `"NOVAK"`, `"RAFAEL"`).

4 A string representing the *last name* of the tennis player (e.g. `"FEDERER"`, `"DJOKOVIC"`, `"NADAL"`).

5 A string representing the *year* the tennis player was born (e.g. `"1981"`, `"1987"`, `"1986"`).

6 A string representing the country of the player (e.g. `"SWITZERLAND"`, `"SERBIA"`, `"SPAIN"`).

Tennis single matches will be represented by several strings, separated by slashes, in this order: [2]

1 The string `"MATCH"`, indicating that this line represents a *tennis single match*.

2 A string representing the *id* of the *first tennis player* of the tennis match (e.g. `"FED81"`).

3 A string representing the *id* of the *second tennis player* of the tennis match (e.g. `"NAD86"`). [3]

4 An 8-digit integer representing the date of the tennis match, in the form "*YYYYMMDD*", where: "*YYYY*" is the year, "*MM*" is the month, and "*DD*" is the date of the tennis match (e.g. `"20070308"`).

5 A string representing the location (i.e. the tournament) of the tennis match (e.g. `"WIMBLEDON"`).

6 A string representing the score (and the winner) of the tennis match (e.g. `"6-4,2-6,7-6"`).

[1] See: Wikipedia - Types of Tennis Match.
[2] Note: all the strings should be stored and processed in uppercase.
[3] Note: the second id cannot be equal to the first id.

Your program will begin by reading in this information and storing it internally in an appropriate format (see: *"Internal Requirements"*).

INTERACTIVE INPUT

Your program will then interact with the user running the program, offering the user a menu of commands. The following commands should be offered at this time:

- *Print the list of tennis players, sorted alphabetically by their ids* (the same sorted order should be used to store the data). If selected, the program should print the descriptions of all tennis players known in the database in an appropriate format, including the win/loss record for each tennis player.
- *Print tennis matches log for one tennis player, sorted by date (most recent first)* (the same sorted order should be used to store the data). If selected, the program should ask the user for a tennis player unique id. Using that string, the program should display all tennis single matches where that id appears as the first or second tennis player. [4]
- *Print all tennis matches, sorted by date (most recent first)* (the same sorted order should be used to store the data).
- *Insert new tennis player or new tennis match*. If selected, the program will prompt the user for the appropriate information and add the specified data to the tennis database. Duplicate tennis players should be rejected by the program. [5]
- *Exit the program.* If selected, the program should terminate.

INTERNAL REQUIREMENTS

As always, your program should use good style, and your coding style will be evaluated accordingly to standard guidelines. [6]

Your program should catch (and handle appropriately) all exceptions generated by any system routines you use, as well as any exceptions you generate yourself. (That is, your `main` method should not throw any exceptions.)

Your program should be designed with modularity and modifiability in mind. You will be revising and extending this program in the next assignment; consequently, it is to your benefit to design your program in as modular a fashion as possible.

[4] Note: if no matching record can be found, an appropriate message should be displayed.
[5] Note: you are free to decide how to handle duplicate tennis matches (accepted or rejected) explaining your choice.
[6] If you need a reference you can use the Google Java Style Guide.

In particular:

- *You should define a class* `TennisPlayer` which contains members corresponding to a given tennis player, and methods which deal with the tennis player statistics (e.g. `print`, `getWinLossRecord`).
- *You should define a class* `TennisMatch` which contains members corresponding to a given tennis single match, and methods which deal with the tennis match details (e.g. `getWinner`, `getDate`).
- *You should define a class* `TennisDatabase` which contains methods which allow to manipulate a collection of tennis records (e.g. `searchTennisPlayer`, `printTennisPlayerMatches`). The `TennisDatabase` class should never access the internal data members of the `TennisPlayer` and `TennisMatch` classes directly.
- For this assignment, your `TennisDatabase` class should be designed to store 2 objects: an object of type `TennisPlayersContainer`, storing all the tennis players; and an object of type `TennisMatchesContainer`, storing all the tennis matches.
- The `TennisPlayersContainer` should be a *two-level list* to store tennis players. The *upper-level list* will be a *circular doubly linked list* (i.e. *reference-based*) without *dummy head nodes*, called `TennisPlayersContainer` and sorted by player id (alphabetically). The nodes in the *upper-level linked list* will be instances of the `TennisPlayerNode` class, and they will store tennis players, with one node per player. Each `TennisPlayerNode` object will also contain a *lower-level linked list* of type `TennisMatchesList` storing the tennis matches of that player. [7] [8]
- The `TennisMatchesContainer` should be a *dynamically allocated array-based list* to store tennis matches. Its items should be instances of the `TennisMatch` class, and the list should be sorted by date (most recent first). This list should be implemented using an array of `TennisMatch` objects, re-allocating the array (i.e. increasing its physical size) whenever you need to add a new tennis match and the array is full. [9]
- *You should define a class* `Assignment1` with a `main` method which uses a `TennisDatabase` object to store data and perform the operations requested by the user. The `Assignment1` class should never access the internal data members of the other classes (i.e. `TennisDatabase`, `TennisPlayer`, `TennisMatch` etc.) directly.
- The function to parse a tennis match score to identify the winner, should be implemented *recursively* (on a single set score).
- Each of the classes mentioned above (except `Assignment1`) will need to implement their relative *interface*, designed and developed in class during lab sessions (`TennisPlayerInterface`, `TennisMatchInterface`, `TennisDatabaseInterface`, etc.).

[7] Note: remember that each node should also store the fields to implement the circular doubly linked list.
[8] Note: all the operations on these lists should exploit their structure and the fact that they are sorted.
[9] Note: you are free to decide the strategy for re-allocating the array (i.e. the increase in the physical size of the array).

A SAMPLE SESSION

The following is an *example* of the content of the data file which could be read by this program:

```
PLAYER/FED81/ROGER/FEDERER/1981/SWITZERLAND
MATCH/FED81/NAD86/20070308/WIMBLEDON/6-4,2-6,7-6
MATCH/DJO87/FED81/20121231/US OPEN/6-2,3-6,6-7,7-5,6-1
PLAYER/DJO87/NOVAK/DJOKOVIC/1987/SERBIA
PLAYER/NAD86/RAFAEL/NADAL/1986/SPAIN
MATCH/NAD86/DJO87/20150101/ROME/6-2,6-3,6-4
```

The following is an *example* of an interactive session:

```
Welcome to the CS-102 Tennis Manager
Current available commands:
1 --> Print all tennis players
2 --> Print all tennis matches of a player
9 --> Exit
Your choice? 1

FED81: ROGER FEDERER, 1981, SWITZERLAND, 285/47 (WIN/LOSS)
DJO87: NOVAK DJOKOVIC, 1987, SERBIA, 214/34 (WIN/LOSS)
NAD86: RAFAEL NADAL, 1986, SPAIN, 195/27 (WIN/LOSS)

Current available commands:
1 --> Print all tennis players
2 --> Print all tennis matches of a player
9 --> Exit
Your choice? 2
Enter tennis player id: DJO87

2012/12/31, N.DJOKOVIC-R.FEDERER, US OPEN, 6-2,3-6,6-7,7-5,6-1
2015/01/01, R.NADAL-N.DJOKOVIC, ROME, 6-2,6-3,6-4

Current available commands:
1 --> Print all tennis players
2 --> Print all tennis matches of a player
9 --> Exit
Your choice? 9
```

Note that these are only *examples*; your program may operate differently (and appear differently) as long as it fulfils the requirements outlined above.

You should create multiple input files in order to test your program thoroughly; do not assume that simply testing the program on the input file shown above is sufficient to test your program. (What happens if the input file is empty? What if it contains too many entries?)

SUBMITTING YOUR ASSIGNMENT

Before the 11:59:59 p.m. of the 5th Friday, using your Kettering email account, you must send an email to the instructor (`gturini@kettering.edu`) with the subject: `CS-102: Assignment 1`. This email should have in attachment a zip archive (`.zip`) containing:

- all source code files of your assignment,
- a *"readme"* text file (`readme.txt`) with the instructions on how to use your program, and
- nothing else. [10]

NOTES

1 *Start early!* Please consider that this assignment has been designed for a 5-weeks period.

2 Remember to use an appropriate coding style, since it will affect 25% of your grade.

3 Input from the console and from files should be handled using the Java `Scanner` class. [11]

4 Please consider that `String` objects should be not compared directly with the `==` operator. Various methods in the Java `String` class can assist you in performing your searches. [12]

[10] Please do not include .class files or IDE configuration files in the .zip file you submit.
[11] See: Java Standard Edition (SE) 8 - Scanner.
[12] See: Java Standard Edition (SE) 8 - String.

CS-102: COMPUTING AND ALGORITHMS II
ASSIGNMENT 1 - 2018 SPRING

GIUSEPPE TURINI - KETTERING UNIVERSITY

EVALUATION FORM

STUDENT NAME:

FUNCTIONALITY		___________ / 50
Reads File	___________ / 10	
Works correctly in *"normal"* conditions (5)		
Handles *"bad"* entries and excess of data (5)		
Print Tennis Matches of a Player, and in Total	___________ / 10	
Prints all correct items in order (5)		
Handles empty list and non-present (5)		
Print Tennis Players	___________ / 10	
Prints all correct items in order (5)		
Handles empty list (5)		
Insert	___________ / 15	
Add tennis players or matches correctly (5)		
Maintain sorted order in containers (5)		
Handles duplicates (5)		
Interface	___________ / 5	

DESIGN	___________ / 25
Proper configuration of the required classes / interfaces (5)	
Proper use of OOP principles (5)	
Proper configuration / use of ADTs (10)	
Compilation requirements (exceptions, deprecated code) (5)	

STYLE	___________ / 25
Headers (files, methods) (5)	
Variables (meaningful names, capitalization, constants) (5)	
Formatting (whitespace, line length, method length) (5)	
Commenting (inline, variables) (5)	
Files (one per class, proper submission) (5)	

TOTAL __________ / 100