CS-102: COMPUTING AND ALGORITHMS II
ASSIGNMENT 2 - 2018 SPRING

GIUSEPPE TURINI - KETTERING UNIVERSITY

INTRODUCTION

In this programming assignment, you will demonstrate your knowledge of dynamic data structures (in particular, the Java Collections Framework, and *binary search trees*) by revising the program from CS-102: Assignment 1 to incorporate different *dynamic data structures*.

FILE INPUT

The file input for this assignment will be the same as for the CS-102: Assignment 1.

Your program will store data internally in an appropriate format (see: *"New Internal Requirements"*).

NEW INTERACTIVE INPUT

Your program will be an extension of CS-102: Assignment 1, and thus should operate in the same manner as that assignment, unless otherwise specified herein. In particular, this means that any error present in your submission for CS-102: Assignment 1 should be fixed for this assignment.

The following new requirements should be implemented as well:

- *All the printing commands of tennis matches* should print the first name and last name of both players (instead of their ids), and the winner of the match (1 or 2).
- *Delete a tennis player*. If selected, the program should ask the user for a tennis player unique id. Using that string, the program should delete that tennis player (if found in the database). [1]
- *Reset tennis database*. If selected, the program will clear the tennis database: deleting all tennis players and tennis matches.
- *Import-export the tennis database from-to file*. If selected, the program will prompt the user for a file name to be used to import-export the tennis database information. Please consider that: exporting the database to file, resetting the tennis database, and then importing the previously exported file, should result in the same tennis database (with the same internal structure for all the data).
- A *graphical user interface* (GUI) developed in JavaFX. [2]

[1] Note: if no matching record can be found, an appropriate message should be displayed.

[2] See: Java Standard Edition (SE) 8 - JavaFX Overview.

NEW INTERNAL REQUIREMENTS

As always, your program should use good style, and your coding style will be evaluated accordingly to standard guidelines. [3]

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.

In particular, for this assignment, you should replace your `TennisDatabase` class with another implementation which uses:

- The `TennisMatch` class should store `TennisPlayer` objects instead of tennis player ids.
- The `TennisMatchesContainer` should be implemented using one of the following JCF classes: `ArrayDeque`, `ArrayList`, `LinkedList`, `Stack`, or `Vector`. [4] [5] [6] [7] [8]
  The student is required to explain in the *"readme"* text file, for each one of these five classes, the reasons why each JCF class listed above was chosen or discarded in order to develop the `TennisMatchesContainer` class. [9]
- The `TennisPlayersContainer` should be implemented using a *binary search tree* (*reference-based*) implemented by the student without using any JCF class. [10]
- The `TennisPlayersContainer` class should be accompanied by a `TennisPlayersContainerIterator` class representing the iterator for the collection of tennis players and implementing both: the *inorder* (i.e. *ascending order of ids*), and *reverse inorder* (i.e. *descending order of ids*) traversals of this collection. [11]
- *You should define a class* `Assignment2` with a `main` method which uses a `TennisDatabase` object to store data and perform the operations requested by the user. The `Assignment2` class should never access the internal data members of the other classes directly.
- Each of the classes mentioned above (except `Assignment2`) will need to implement their relative *interface*, designed and developed in class during lab sessions.

[3] If you need a reference you can use the Google Java Style Guide.
[4] See: Java Standard Edition (SE) 8 - ArrayDeque.
[5] See: Java Standard Edition (SE) 8 - ArrayList.
[6] See: Java Standard Edition (SE) 8 - LinkedList.
[7] See: Java Standard Edition (SE) 8 - Stack.
[8] See: Java Standard Edition (SE) 8 - Vector.
[9] Note: please write this information in the *"readme.txt"* file included in your submission.
[10] See: Wikipedia - Binary Search Tree.
[11] See: Wikipedia - Iterator.

SUBMITTING YOUR ASSIGNMENT

Before the 11:59:59 p.m. of the 10th Friday, using your Kettering email account, you must send an email to the instructor (`gturini@kettering.edu`) with the subject: `CS-102: Assignment 2`. 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, any information required by this assignment, and other comments the student wants to provide. [12]

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.

[12] Please do not include .class files or IDE configuration files in the .zip file you submit.

CS-102: COMPUTING AND ALGORITHMS II
ASSIGNMENT 2 - 2018 SPRING

GIUSEPPE TURINI - KETTERING UNIVERSITY

EVALUATION FORM

STUDENT NAME:

FUNCTIONALITY	_____ / 50
Print Tennis Matches	_____ / 10
Full name of tennis players (5)	
Correct computation of winner (5)	
Delete Tennis Players	_____ / 10
Removal preserving internal sorting (5)	
Correct update of relative tennis matches (5)	
Reset Tennis Database	_____ / 5
Import-Export Tennis Database From-To File	_____ / 10
Import tennis database allowing perfect rebuilt (5)	
Export tennis database allowing perfect rebuilt (5)	
Graphical User Interface (GUI) in JavaFX	_____ / 15

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