CST2110 Resit Assessment Part 2

Deadline for submission Thursday July 4th, 2024. 12:00.

Please read all the assignment specification carefully first, before asking your tutor any questions.

General information

You are required to submit your work via the dedicated assignment link in the module myUnihub space by the specified deadline. This link will 'timeout' at the submission deadline.

Your work will not be accepted as an email attachment.

Therefore, you are strongly advised to allow plenty of time to upload your work prior to the deadline.

Submission should comprise a single 'ZIP' file. This file should contain a separate, cleaned[1], NetBeans projects for the two programming tasks described below. The work will be compiled and run in a Windows environment, i.e., the same configuration as the University networked labs and it is strongly advised that you test your work using the same configuration prior to cleaning and submission.

Important notes (please read carefully)

- Work submitted that is not programmed with the correct Java version (i.e., Java 8) will not be assessed, and will attain zero marks.
- Work submitted that Is not configured correctly as a NetBeans 'Ant' build (as described in the lectures) will not be assessed and attain zero marks.
- Your submission (ZIP file) must also include a completed declaration of authenticity using the form provided in the module myUnihub space. Your work will not be marked if you do not complete and submit the declaration.

Additional note (please read very carefully)

Please refer to Section 6 of the module handbook with regards to integrity and academic misconduct. All assessment for CST2110 is individual. The CST2110 module teaching team regards plagiarism and collusion as very serious matters and apply a zero-tolerance policy. Accordingly, effort is made to check the authenticity of student work which may include the use of (program code) plagiarism detection tools.

[1] In the NetBeans project navigator window, right-click on the project and select 'clean' from the drop-down menu. This will remove .class files and reduce the project file size for submission.

<u>Task 1 (30 marks)</u>

Create a NetBeans project for this task, named *Task1*.

You are required to write a Java 8 program that opens and reads a data file that is located relative to the NetBeans project root folder. The data file is called *footballers.txt*. The data file must not be altered and should be considered as a *read-only* data file.

The data file is delimited in a consistent format and contains entries relating to football player of the year awards from the 1994/95 season to the 2022/23 season of the English premier league. Each entry in the file contains a series of data fields representing the following information: the season that the football player won the award, the name of the football player, the team position of the football player (i.e., forward, defender, goalkeeper etc.), the nationality of the football player, and the club side that the football player played for.

You are required to implement a Java class to represent the football player of the year information with respect to this data set. The program should parse the data file once and create and store a collection of objects that represent the domain entity on program start-up. Once the collection is created, the program should only access the data in that collection (and not make further access to the external data file). Figure 1 provides a <u>partial</u> UML class representation of a *FootballPlayer* class that you must implement to represent the data. It is left to you to determine what methods the class should contain, as well as how the respective objects should be initialised. The *FootballPlayer* class should implement an appropriate *toString()* method to format console output appropriately when invoked as indicated below.

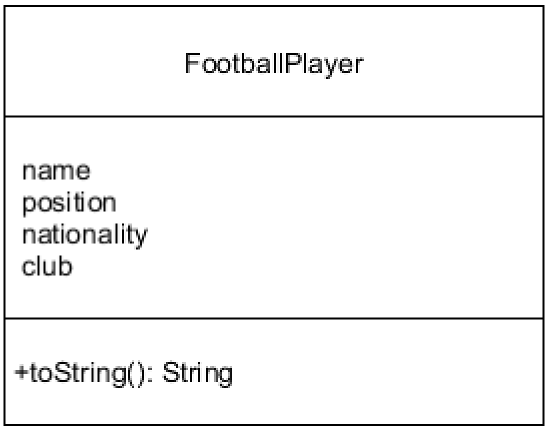

Figure 1: Partial UML class model for football player information.

Once the data has been loaded into the program, the User should be presented with a console-based menu to interact with the data set. This menu should loop until the User enters a character to exit the menu (e.g., zero as illustrated below). In addition to an exit option, the menu should offer <u>three</u> other options: list, select and sort.

On starting the program, the following menu should be displayed to the console:

```
-----------------------
Player of the year menu
-----------------------
List .................1
Select ...............2
Sort..................3
Exit..................0
-----------------------
Enter choice:>
```

The User can simply exit the program by entering zero. The three other menu options allow the User to inspect the information in the data set (note again that this program is entirely *read-only* and there is no requirement to add, update or delete any part of the data set). The necessary interaction of the program with respect to these options is illustrated in Appendix A.

Note that console output should be neatly formatted, and some consideration will be given to formatting when the program is assessed. In particular, when the option to view the details for a given season is selected (i.e., the 'select' menu option), it must result in the invocation of the relevant *toString()* method. You are required to utilise a *StringBuilder* object when implementing the *toString()* methods for the domain class illustrated in Figure 1.

The assessment rubric provided in the module handbook for the first sit will be applied.

Task 2 (40 marks)

Create a new NetBeans project, called *Task2*.

For this task you are required to write a Java 8 program that incorporates a series of Java classes that represent the core logic of an application and provides a NetBeans 8 console user interface. The required Java application relates to a proposed software system to manage some aspects of a sailing club, including hiring sailing dinghies and lesson bookings for any given week. Below is an initial description of the system domain including five use cases (structured as step-by-step natural language statements).

System domain

There are two types of sailing club members: learners and qualified. The club employs several sailing instructors (who are not members), and each learner is assigned to a single instructor. All members and instructors are identified by their name. An instructor can provide individual lessons to either their assigned learners or to qualified members (who are not assigned to instructors). The club owns 10 sailing dinghies, and each has a unique name to identify it. Lessons, which are one hour long and start on the hour (between the hours of 09:00 and 18:00), can be booked for Wednesdays, Saturdays and Sundays. Lessons are each taken by a single member and provided by one instructor in one of the club dinghies. Qualified members may book dinghies for hire during the day (i.e., from 09:00 to 18:00, Wednesday, Saturday, or Sunday) for one hour (on the hour) without involving an instructor. A member may be scheduled to take no more than three lessons for any given week. A qualified member may have bookings for no more than two dinghy hires per week. Sailing dinghies, instructors, and members can each be involved in only one lesson or hire at any one time.

Use cases

1: Book lesson for member

The software system prompts the User (i.e., the administrator) to select a member from a list of members. If the selected member has already reached their limit on lessons for the week, then the system informs the User (by displaying an appropriate message) and takes no further action (i.e., the system returns to the main menu). If the selected member is a learner, then the system displays the instructor's name that the learner is allocated to and displays a weekly schedule of lessons for that instructor (i.e., for the current week), indicating which hourly slots are currently free for that instructor (and conversely, which slots are already booked). If the selected member is a qualified member, then the system prompts the User to select an instructor from a list of instructor names displayed to the console. On selection of an instructor, the system displays a weekly schedule of lessons for that instructor (i.e., for the current week), indicating which hourly slots are currently free for that instructor (and conversely, which slots are already booked).

The User is then prompted to enter a day and time (from the displayed schedule) to propose a booking.

If the member or the instructor is already involved with a lesson at the day/time of the proposed booking (i.e., User selection), or the member is qualified and has a dinghy hire at the day/time of the proposed booking, then the system informs the User (by displaying an appropriate message) and takes no further action (i.e., the system returns to the main menu).

Otherwise, the system checks whether any of the club dinghies are available at the selected time/day (i.e., they are not already booked for either a lesson or a hire at the selected time/day). If there are no available club dinghies available at that time/day, the system informs the user (by displaying an appropriate message) and takes no further action (i.e., the system returns to the main menu). If any of the club dinghies are available, then the names of the available dinghies are displayed to the console and the User is prompted to select which dinghy should be used and, on selection, the lesson is recorded (for the runtime of the program). A confirmation message is displayed with the details of the lesson.

2: List member lessons

The system allows the User (i.e., the administrator) to select a member from a list of members of the sailing club. If the member currently has no lesson bookings for that week, then the system informs the user (by displaying an appropriate message) and takes no further action (i.e., the system returns to the main menu). If the member does have lesson bookings, then the system displays the details (instructor, day, time, and dinghy name) of the bookings ordered by day and then time.

3: List instructor lessons

The system allows the User (i.e., the administrator) to select an instructor from a list of instructors of the sailing club. If the instructor currently has no lesson bookings for that week, then the system informs the user (by displaying an appropriate message) and takes no further action (i.e., the system returns to the main menu). If the instructor does have lesson bookings, then the system displays the details (member, day, time, and dinghy name) of the bookings ordered by day and then time.

4: Hire dinghy for qualified member

The software system prompts the User (i.e., the administrator) to select a qualified member from a list of qualified members. If the selected member has already reached their limit on dinghy hires for the week, then the system informs the User (by displaying an appropriate message) and takes no further action (i.e., the system returns to the main menu).

The User is then prompted to enter a day and time to propose a dinghy hire.

If the qualified member is already involved with a lesson or hire at the day/time of the proposed dinghy hire, then the system informs the User (by displaying an appropriate message) and takes no further action (i.e., the system returns to the main menu).

Otherwise, the system checks whether any of the club dinghies are available at the selected time/day (i.e., they are not already booked for either a lesson or a hire at the selected time/day). If there are no available club dinghies available at that time/day, the system informs the user (by displaying an appropriate message) and takes no further action (i.e., the system returns to the main menu). If any of the club dinghies are available, then the names of the available dinghies are displayed to the console and the User is prompted to select which dinghy should be used and, on selection, the hire is recorded (for the runtime of the program). A confirmation message is displayed with the details of the hire.

5: Display dinghy bookings

The system allows the User (i.e., the administrator) to select a dinghy from a list of all dinghy names owned by the sailing club. On selection of a dinghy name, the system displays a weekly schedule (i.e., all days and times) of bookings for the selected dinghy, indicating if the dinghy is booked for a lesson, booked for hire, or still available for each day/time slot across the week.

Your task is to design a Java 8 program for the sailing club administration system described above, which provides a console-based (text I/O) user interface that facilitates the listed use cases. An initial analysis of the domain has already been completed, and a partial class model has been designed as shown in Figure 2.

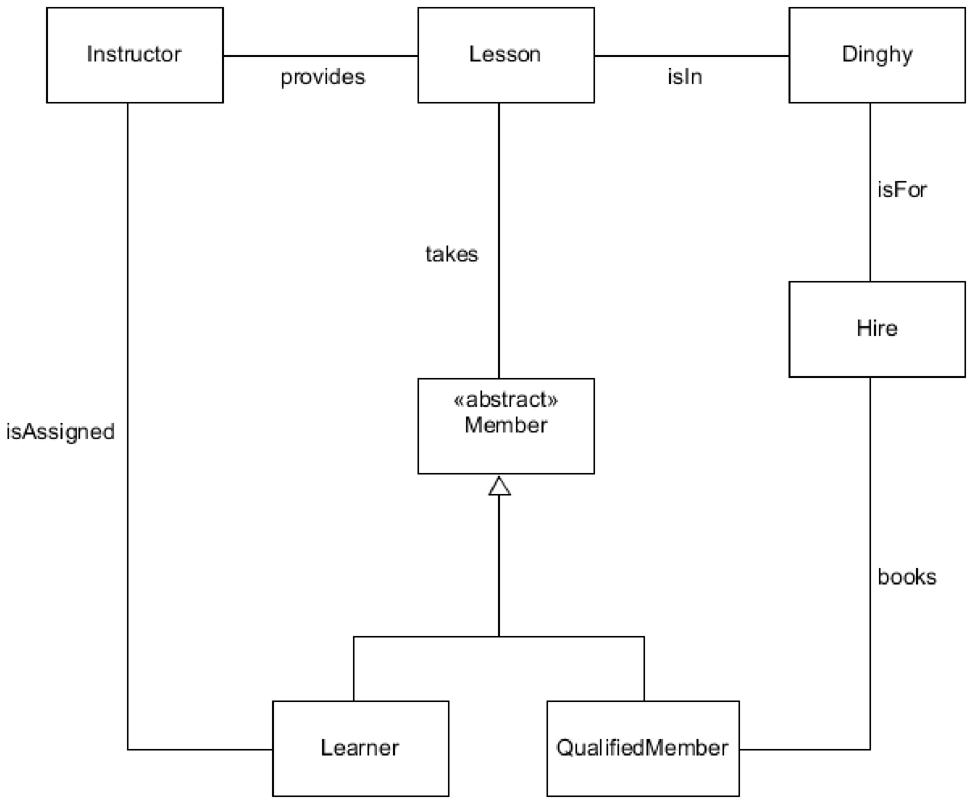

Figure 2: Class model of the swim school system.

The class diagram above is provided as an initial design for your Java implementation, i.e., your application must include Java classes for the above class model as a minimum. You can extend/enhance the model if you wish to, but the above model structure must be implemented as specified. You are not required to submit any UML for this task.

So that the specified use cases can be tested, your program must provide a console-based User interface menu, with a selectable menu item for each of the use cases. Your application should pre-load some 'dummy' data into the application, but also allow information to be input (e.g., use cases 1 and 4) during runtime. Pre-loaded data should be achieved by coding data directly within the Java program. It should **not** use a file, or an external database link (e.g., such as an SQL database). All pre-loaded data should be *read-only*. Any data that is input during the runtime of the Java program should be volatile i.e., it should exist only for the time the program is executing and should not be stored to non-volatile location such as an external file or a database.

The assessment rubric provided in the module handbook for the first sit will be applied.

Your program will be assessed according to overall object-oriented design and implementation, and the inclusion of an appropriate text-based (console-driven) user-interface to allow the use cases to be tested fully. Credit will be given for appropriate encapsulation of system components.

Appendix A: Console interaction examples for Task 1

Option 1: list players of the year

On selecting option 1, the User should be presented with a neatly formatted table of the following information: the season, the name of the player of the year for that season, and the English league club that they played for. The required output is illustrated below. Note that all data displayed to the console must be dynamically derived from the collection of *FootballPlayer* objects that have been loaded on start-up. Simply 'hard coding' the output illustrated below with fixed strings will be awarded no marks.

```
-----------------------
Player of the year menu
-----------------------
List .................1
Select ...............2
Sort..................3
Exit..................0
-----------------------
Enter choice:> 1

---------------------------------------------------------------------
|  Season   | Player                      | Club                     |
---------------------------------------------------------------------
|  1994-95  | Alan Shearer                | Blackburn Rovers         |
|  1995-96  | Peter Schmeichel            | Manchester United        |
|  1996-97  | Juninho                     | Middlesbrough            |
|  1997-98  | Michael Owen                | Liverpool                |
|  1998-99  | Dwight Yorke                | Manchester United        |
|  1999-00  | Kevin Phillips              | Sunderland               |
|  2000-01  | Patrick Vieira              | Arsenal                  |
|  2001-02  | Freddie Ljungberg           | Arsenal                  |
|  2002-03  | Ruud van Nistelrooy         | Manchester United        |
|  2003-04  | Thierry Henry               | Arsenal                  |
|  2004-05  | Frank Lampard               | Chelsea                  |
|  2005-06  | Thierry Henry               | Arsenal                  |
|  2006-07  | Cristiano Ronaldo           | Manchester United        |
|  2007-08  | Cristiano Ronaldo           | Manchester United        |
|  2008-09  | Nemanja Vidić               | Manchester United        |
|  2009-10  | Wayne Rooney                | Manchester United        |
|  2010-11  | Nemanja Vidić               | Manchester United        |
|  2011-12  | Vincent Kompany             | Manchester City          |
|  2012-13  | Gareth Bale                 | Tottenham Hotspur        |
|  2013-14  | Luis Suárez                 | Liverpool                |
|  2014-15  | Eden Hazard                 | Chelsea                  |
|  2015-16  | Jamie Vardy                 | Leicester City           |
|  2016-17  | N'Golo Kanté                | Chelsea                  |
|  2017-18  | Mohamed Salah               | Liverpool                |
|  2018-19  | Virgil van Dijk             | Liverpool                |
|  2019-20  | Kevin De Bruyne             | Manchester City          |
|  2020-21  | Rúben Dias                  | Manchester City          |
|  2021-22  | Kevin De Bruyne             | Manchester City          |
|  2022-23  | Erling Haaland              | Manchester City          |
---------------------------------------------------------------------
```

Option 2: select year

On selecting option 2, the User should be prompted to enter the year of the award. For example, if the User wishes to view the details of the player of the year for the 1997/98 season, then the User must enter 1998. If an incorrect year is entered, an appropriate message should be displayed so the User can try again. On entering a valid year, the console should display a neatly formatted representation of information pertaining to the winning player for that season as illustrated below. This should include the name of the football player, their position, their nationality, and the name of the English club side that they played for. The console output should be achieved by invocation of the relevant *toString()* method.

```
-----------------------
Player of the year menu
-----------------------
List .................1
Select ...............2
Sort..................3
Exit..................0
-----------------------
Enter choice:> 2

Enter year of award > 1998

Player of the year for the 1997-98 season:

-----------------------------------------------
| Player          | Michael Owen              |
-----------------------------------------------
| Position        | Forward                   |
-----------------------------------------------
| Nationality     | England                   |
-----------------------------------------------
| Club            | Liverpool                 |
-----------------------------------------------


-----------------------
Player of the year menu
-----------------------
List .................1
Select ...............2
Sort..................3
Exit..................0
-----------------------
Enter choice:>
```

Option 3: sort

On selecting option 3 from the main menu, the User should be presented with a sub-menu that allows the User to inspect the data set in respect of three sorting options:

```
-----------------------
Player of the year menu
-----------------------
List .................1
Select ...............2
Sort..................3
Exit..................0
-----------------------
Enter choice:> 3
```

```
-------------------------
Sort number of awards by:
-------------------------
Position ...............1
Nationality ............2
Club....................3
Exit....................0
-------------------------
Enter choice:>
```

On selecting the first option, the data is sorted (and displayed to the console) by the player position, with the highest number of winners who played in that position listed first to the lowest. This should be ordered by the total number of awards for players in that position (which can be different to the number of players because the same player may have won more than once). The data sorting must be performed dynamically from the preloaded collection of objects. Simply 'hard coding' the output illustrated below with fixed strings will be awarded no marks.

```
-------------------------
Sort number of awards by:
-------------------------
Position ...............1
Nationality ............2
Club....................3
Exit....................0
-------------------------
Enter choice:> 1

----------------------------------------
| Position   |  Players   |   Total    |
----------------------------------------
| Forward    |     12     |     14     |
----------------------------------------
| Midfielder |     8      |     9      |
----------------------------------------
| Defender   |     4      |     5      |
----------------------------------------
| Goalkeeper |     1      |     1      |
----------------------------------------
```

Similarly, selecting the second option should sort and display the data by nationality, as illustrated below. Again, this should be awarded by highest (total) first. And again, simply 'hard coding' the output illustrated below with fixed strings will be awarded no marks.

```
-------------------------
Sort number of awards by:
-------------------------
Position ..............1
Nationality ...........2
Club...................3
Exit...................0
-------------------------
Enter choice:> 2

--------------------------------------------------
|       Country        |  Players   |   Total    |
--------------------------------------------------
| England              |     6      |     6      |
--------------------------------------------------
| Belgium              |     3      |     4      |
--------------------------------------------------
| France               |     3      |     4      |
--------------------------------------------------
| Portugal             |     2      |     3      |
--------------------------------------------------
| Netherlands          |     2      |     2      |
--------------------------------------------------
| Serbia               |     1      |     2      |
--------------------------------------------------
| Uruguay              |     1      |     1      |
--------------------------------------------------
| Wales                |     1      |     1      |
--------------------------------------------------
| Egypt                |     1      |     1      |
--------------------------------------------------
| Sweden               |     1      |     1      |
--------------------------------------------------
| Norway               |     1      |     1      |
--------------------------------------------------
| Brazil               |     1      |     1      |
--------------------------------------------------
| Denmark              |     1      |     1      |
--------------------------------------------------
| Trinidad and Tobago  |     1      |     1      |
--------------------------------------------------
```

And similarly, the third sort option should sort and display the data by club, as illustrated below.

```
-------------------------
Sort number of awards by:
-------------------------

Position ...............1
Nationality ............2
Club....................3
Exit....................0
-------------------------
Enter choice:> 3

--------------------------------------------------
|         Club          |  Players    |   Total    |
--------------------------------------------------
| Manchester United     |      6      |     8      |
--------------------------------------------------
| Manchester City       |      4      |     5      |
--------------------------------------------------
| Liverpool             |      4      |     4      |
--------------------------------------------------
| Arsenal               |      3      |     4      |
--------------------------------------------------
| Chelsea               |      3      |     3      |
--------------------------------------------------
| Tottenham Hotspur     |      1      |     1      |
--------------------------------------------------
| Leicester City        |      1      |     1      |
--------------------------------------------------
| Middlesbrough         |      1      |     1      |
--------------------------------------------------
| Sunderland            |      1      |     1      |
--------------------------------------------------
| Blackburn Rovers      |      1      |     1      |
--------------------------------------------------
```