CST2110 Summative Assessment Part 2

Deadline for submission **16:00, Friday 22nd April, 2022**

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 'week 24' folder of 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 if you miss this deadline. Therefore, you are strongly advised to allow plenty of time to upload your work prior to the deadline.

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.

Submission should comprise a single 'ZIP' file. This file should contain a separate, cleaned[1], NetBeans project for each of the three tasks described below. The work will be compiled and run in a Windows environment, so it is strongly advised that you test your work in a Windows environment prior to cleaning and submission.

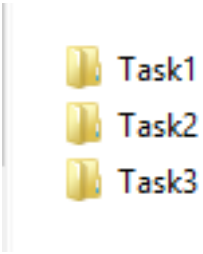

Figure 1: When the ZIP file is extracted there should be three folders named Task1, Task2 and Task3

Accordingly, when loaded into NetBeans, each task must be a separate project as illustrated by Figure 2 below.

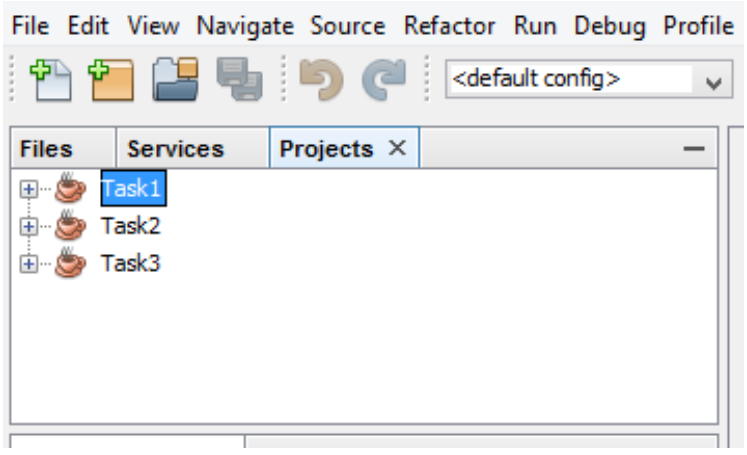

Figure 2: Each task must be a separate NetBeans Project

To make this easier, a template NetBeans project structure is provided for you to download.

[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.

Task 1

Create a NetBeans 8 project for this task, named *Task1*.

You are required to write a Java 8 program that opens and reads a delimited data file that is located relative to the NetBeans project root folder. The delimited data file contains information about Booker prize winning and shortlisted novels. The data file is called *booker-data.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 structure and contains entries relating to 20 years of Booker prize winners and shortlisted nominees. Each entry contains a series of data fields representing the following information: the year of the prize, the winning author, the winning book title, the publisher of the winning book, the authors and book titles of shortlisted nominees, the panel members who were responsible for deciding the winner, and an indication of which panel member acted as chairperson.

You are required to implement Java classes to represent the Booker prize winning information with respect to this data set. The program should parse the data file and create and store a collection of objects for each entity. Figure 3 provides a partial UML class representation of the classes that you will need to implement. It illustrates a core class to represent the Booker prize information alongside a utility class to represent the basic details of a book. The class model indicates the data members and accessor (i.e., *getter*) methods that map to those data members, and a *toString()* method for the *BookerPrize* class. It is left to you to determine how the objects should be initialised.

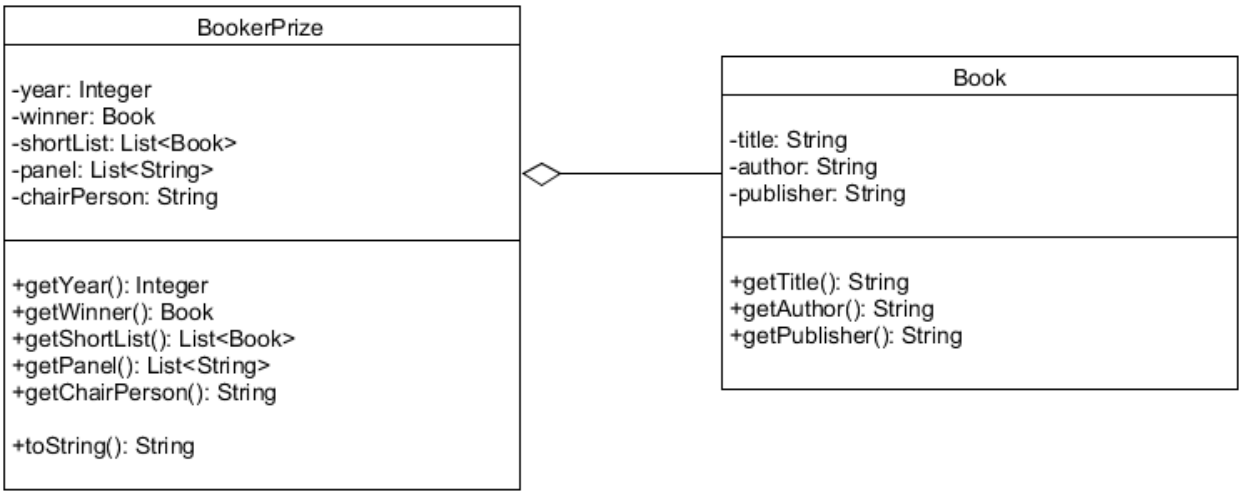

Figure 3: UML class model for Booker prize information

Once all the objects are loaded into the collection, the program should present the User 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 three other options: list, select and search.

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

```
----------------------
Booker prize menu
----------------------
List ................1
Select ..............2
Search ..............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 of the Booker prize winner and shortlist for a given year is selected (i.e., the 'select' menu option), it must result in the invocation of the *toString()* method for that particular *BookerPrize* object. You are required to utilise a *StringBuilder* object when implementing the *toString()* method for the *BookerPrize* class.

An assessment rubric is provided in the module handbook.

Task 1 has a provisional weighting of 20 per cent of the overall module grade.

Task 2

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 motorboat club, including hiring motorboats and training session bookings (lessons) for any given week. Below is an initial description of the system domain. Five use cases (structured as step-by-step natural language statements) are listed in Appendix B.

System domain

There are two types of members: novices and motorboat licence holders (MBLH). The club employs several instructors (who are not members), and each novice 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 novices or to MBLHs (who are not assigned to instructors). The club owns 10 motorboats, 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 all seven days of the week. Lessons are each taken by a single member and provided by one instructor in one of the club motorboats. MBLHs may book motorboats for hire during the day (i.e., from 09:00 to 18:00) 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 MBLH may have bookings for no more than two motorboat hires per week. Motorboats, instructors, and members can each be involved in only one lesson or hire at any one time.

Your task is to design a Java 8 program for the motorboat management system described above, which provides a console-based (text I/O) user interface that facilitates the five use cases listed in Appendix B. An initial analysis of the domain has already been completed, and a class model has been designed as shown in Figure 4.

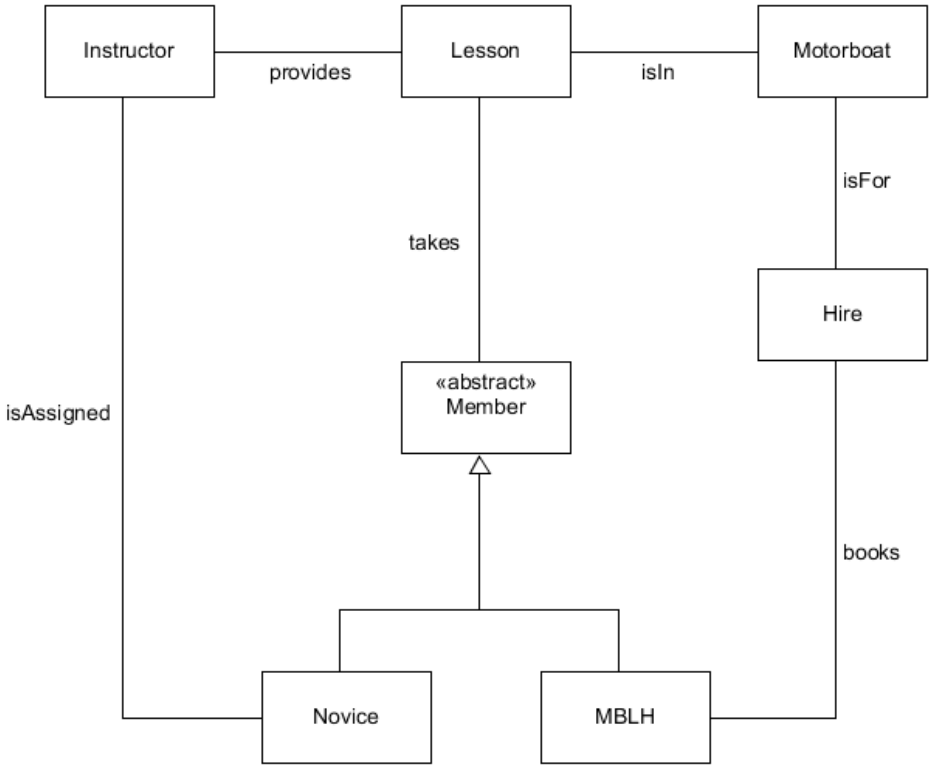

Figure 4: Class model of the motorboat domain.

The class diagram above is provided as a 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 five use cases specified in Appendix B can be tested, your program must provide a console-based User interface menu, with a selectable menu item for each of the five use cases. Your application should pre-load some 'dummy' data into the application, but also allow information to be input (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 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.

An assessment rubric is provided in the module handbook.

Task 2 has a provisional weighting of 40 per cent of the overall module grade.

Task 3

Create a new NetBeans project, called Task3.

For this task you are required to program a JavaFX Graphical User Interface (GUI), with a focus on layout and some event handling.

This program must be coded as a JavaFX program following the approach described in the lecture series and Chapters 14 to 16 of the *kortext*. It must not be created using a visual drag-and-drop tool, and must not be an FXML program, neither of which have been taught on the module. If the submission is either a drag-and-drop application, or a FXML application, then it will simply be awarded zero marks.

The GUI application that you need to create is a basic cross trainer touchscreen simulation. The required design is illustrated in Figure 5a below.

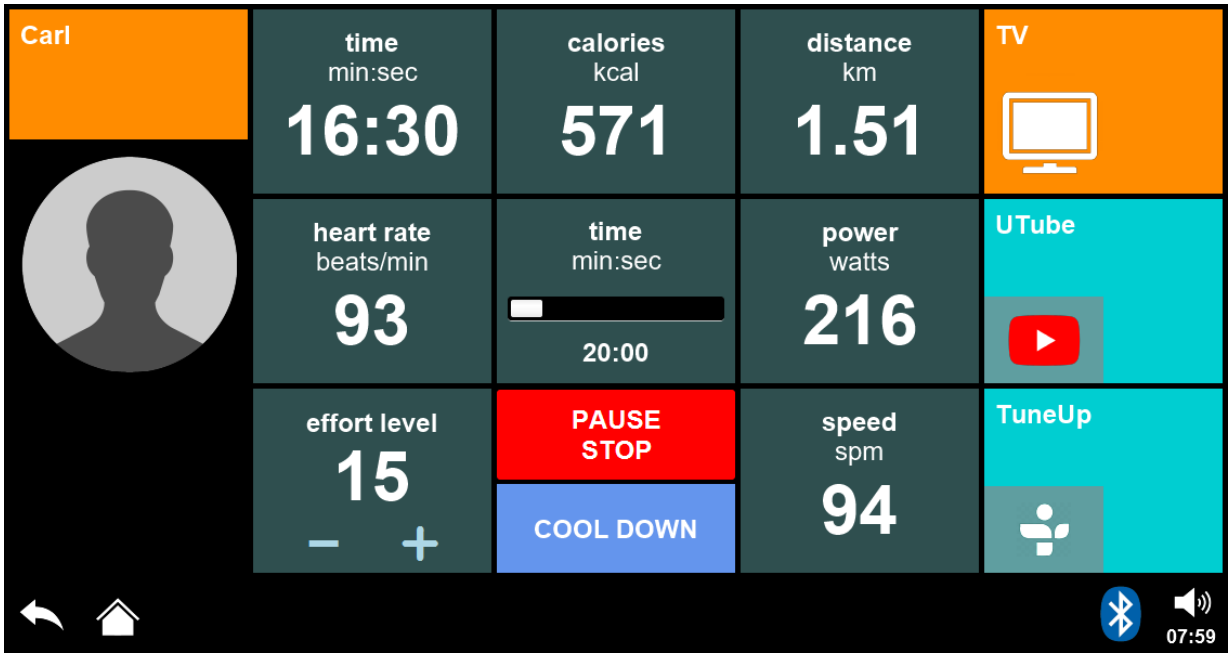

Figure 5a: Cross trainer simulation user interface

The application comprises a layout design with various display sections. A number of these are fixed for the display and are not dynamic or interactive (e.g., calories, distance, heart rate, power, speed, cool down, tv, UTube, and TuneUp sections). There are three sections that are dynamic without the requirement of any user interaction: the countdown timer (upper left) which defaults to 20 minutes when the program starts (displayed as minutes and seconds) and counts down in seconds to zero (at which point it should simply stop), the time section (in the centre) that displays a progress bar that synchronises with the countdown (the 20 minutes text in this section is merely a fixed display), and the real-time clock (bottom right beneath the speaker icon), which displays the current time when the application runs (in hours and minutes).

There are two interactive aspects to the GUI. Firstly, the 'effort level' section should allow the User to increment the effort value up or down by clicking the mouse (to simulate a touchscreen) on the minus and plus symbols. The maximum effort level is thirty and the minimum is zero. The second aspect that is interactive is the 'pause/stop' section. On selection with the mouse, a dialog is overlayed on the main screen which prompts the User to stop the application (i.e., closedown) or restart the simulation (see Figure 5b).

Whilst this modal dialog is displayed, the countdown timer and progress bar should be paused. On selecting 'restart' the countdown timer and progress bar continue from that point.

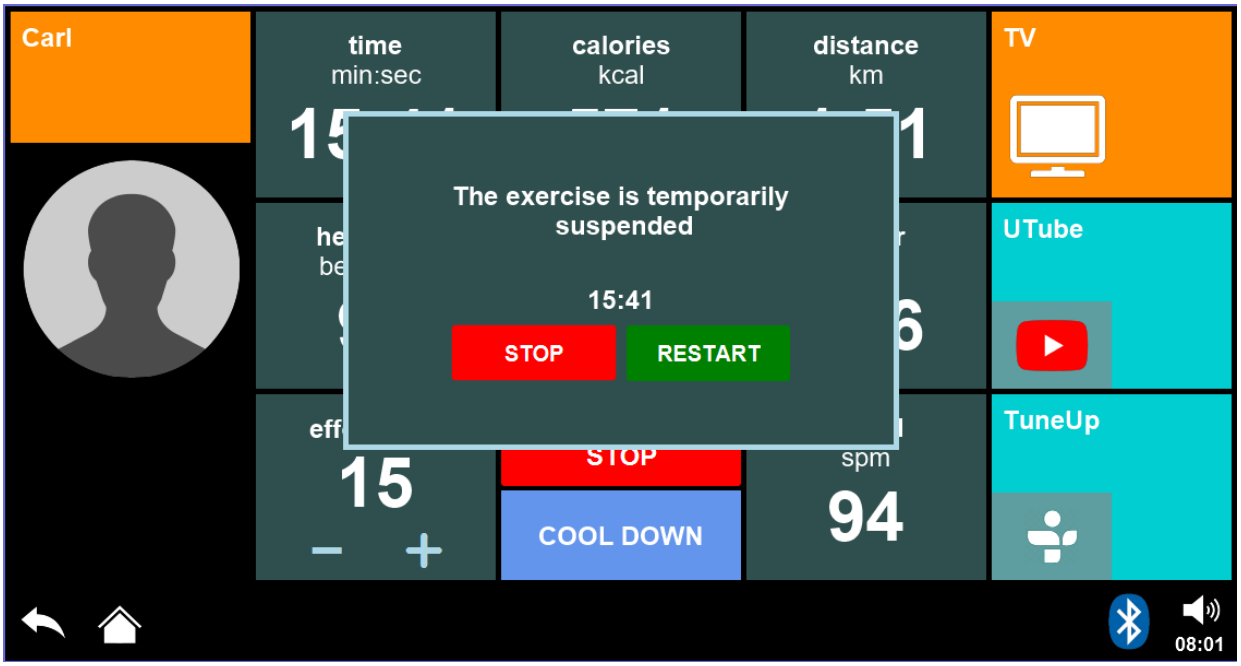

Figure 5b: Model dialog to close or restart the simulation

On selecting 'stop' from the pause dialog, the application should close (this should be the only way that the application can be closed as the main screen has no title bar).

Each area on screen comprises JavaFX nodes and controls. There is no constraint on which nodes and controls and layout panes that you decide to use, and the design can be achieved in different ways.

To assist in explaining how this JavaFX simulation GUI should operate, a video demonstration of a prototype will be made available to view in the module myUnihub space (week 23, main lecture recordings).

Your task is to write a JavaFX program that replicates (as closely as possible) the GUI layout and design as illustrated in Figures 5a and 5b. Styling of widgets must be achieved programmatically using node and control property setting directly (e.g., by using the relevant 'setStyle' method for the respective Node objects). In other words, all layout and styling should be contained within the Java code, use standard JavaFX components, and not by use of a separate CSS file or external custom JavaFX GUI libraries. It is advised that you focus on layout aspects first with regards to the main screen, and when you have a working layout then focus on the interactive and dynamic aspects of the application.

An assessment rubric is provided in the module handbook.

Task 3 has a provisional weighting of 15 per cent of the overall module grade.

Appendix A: Console interaction examples for Task 1

Option 1: list Booker prize winners

On selecting option 1, the User should be presented with a neatly formatted listing of the winning nominees for the Booker prize for the 20 years pertaining to the data set. The data displayed should be ordered by the year of prize, and include the book title, the name of the author, and publisher of the book. To select a single Booker prize to view its full details, the User should be prompted to enter the year of publication (i.e., option 2).

Year	Title	Author	Publisher
2000	The Blind Assassin	Margaret Atwood	Bloomsbury
2001	True History of the Kelly Gang	Peter Carey	Faber & Faber
2002	Life of Pi	Yann Martel	Canongate Books
2003	Vernon God Little	DBC Pierre	Faber & Faber
2004	The Line of Beauty	Alan Hollinghurst	Picador
2005	The Sea	John Banville	Picador
2006	The Inheritance of Loss	Kiran Desai	Hamish Hamilton
2007	The Gathering	Anne Enright	Jonathan Cape
2008	The White Tiger	Aravind Adiga	Atlantic
2009	Wolf Hall	Hilary Mantel	Fourth Estate
2010	The Finkler Question	Howard Jacobson	Bloomsbury
2011	The Sense of an Ending	Julian Barnes	Jonathan Cape
2012	Bring Up the Bodies	Hilary Mantel	Fourth Estate
2013	The Luminaries	Eleanor Catton	Granta
2014	The Narrow Road to the Deep North	Richard Flanagan	Chatto & Windus
2015	A Brief History of Seven Killings	Marlon James	Oneworld Publications
2016	The Sellout	Paul Beatty	Oneworld Publications
2017	Lincoln in the Bardo	George Saunders	Bloomsbury
2018	Milkman	Anna Burns	Faber & Faber
2019	The Testaments	Margaret Atwood	Chatto & Windus
2020	Shuggie Bain	Douglas Stuart	Picador

Option 2: select year

On selecting option 2, the User should be prompted to enter the year of Booker prize (that was displayed when option 1 was chosen). If an incorrect year is entered, an appropriate message should be displayed so the User tries again. On entering a valid year, all the details of that Booker prize should be displayed, as illustrated below. Note that the winning book information must appear at the top with the shortlisted books below and the winning book information must be in upper case. This should be achieved by invocation of the relevant *BookerPrize* object's *toString()* method. Console output should be neatly formatted.

```
----------------------
Booker prize menu
----------------------
List .................1
Select ...............2
Search ...............3
Exit..................0
----------------------

Enter choice:> 2

Enter year of prize winner:> 2019

-------------------------------------------------------------------------------------------------------------------------------------------
| Author               | Book Title                                 | Publisher              | Chair                | Panel                    |
-------------------------------------------------------------------------------------------------------------------------------------------
| MARGARET ATWOOD      | THE TESTAMENTS                             | CHATTO & WINDUS        |                      |                          |
-------------------------------------------------------------------------------------------|                      |                          |
| Bernardine Evaristo  | Girl Woman Other                           | Hamish Hamilton        |                      | Afua Hirsch              |
| Lucy Ellmann         | Ducks Newburyport                          | Galley Beggar Press    | Peter Florence       | Joanna MacGregor         |
| Chigozie Obioma      | An Orchestra of Minorities                 | Little Brown           |                      | Liz Calder               |
| Salman Rushdie       | Quichotte                                  | Jonathan Cape          |                      | Xiaolu Guo               |
| Elif Shafak          | 10 Minutes 38 Seconds in This Strange World | Viking                 |                      |                          |
-------------------------------------------------------------------------------------------------------------------------------------------

----------------------
Booker prize menu
----------------------
List .................1
Select ...............2
Search ...............3
Exit..................0
----------------------

Enter choice:>
```

Option 3: search book titles

On selecting option 3, the User should be prompted to enter a 'search string' to match against the book titles of all the entries (both winning and shortlisted books). All searches should be case insensitive. The program should return a listing of any books that match the search string, with the section that matches in upper-case. The returned matches should also display if the book was a prize winner or shortlisted, and the year of publication.

```
----------------------
Booker prize menu
----------------------
List ..............1
Select ............2
Search ............3
Exit...............0
----------------------

Enter choice:> 3

Enter book title or partial book title > pi


-------------------------------------------------------------------------------------
| Title                                    | Author              | Status      | Year |
-------------------------------------------------------------------------------------
| Life of PI                               | Yann Martel         | Winner      | 2002 |
-------------------------------------------------------------------------------------
| Mister PIp                               | Lloyd Jones         | Shortlisted | 2007 |
-------------------------------------------------------------------------------------
| Sea of PopPIes                           | Amitav Ghosh        | Shortlisted | 2008 |
-------------------------------------------------------------------------------------
| PIgeon English                           | Stephen Kelman      | Shortlisted | 2011 |
-------------------------------------------------------------------------------------


List ..........1
Select .........2
Search..........3
Exit............0

Enter choice:>
```

Appendix B: Use cases for Task 2

Use case #1: Book lesson for member

1 - 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).

2a - If the selected member is a novice, then the system displays the instructor's name that the novice 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).

2b - If the selected member is a MBHL, 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).

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

4a - 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 a MBLH and has a motorboat 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).

4b - Otherwise, the system checks whether any of the club motorboats 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).

4b(i) - If there are no available club motorboats 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)

4b(ii) - If any of the club motorboats are available, then the names of the available motorboats are displayed to the console and the User is prompted to select which motorboat 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.

Use case #2: List member lessons

1 - The system allows the User (i.e., the administrator) to select a member from a list of members of the motorboat club.

2a - 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)

2b - If the member does have lesson bookings, then the system displays the details (instructor, day, time, and motorboat name) of the bookings ordered by day and then time.

Use case #3: List instructor lessons

1 - The system allows the User (i.e., the administrator) to select an instructor from a list of instructors of the motorboat club.

2a - 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)

2b - If the instructor does have lesson bookings, then the system displays the details (member, day, time, and motorboat name) of the bookings ordered by day and then time.

Use case #4: Hire motorboat for MBLH

1 - The software system prompts the User (i.e., the administrator) to select a MBLH member from a list of MBLH members. If the selected MBLH member has already reached their limit on motorboat 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).

2 - The User is then prompted to enter a day and time to propose a motorboat hire.

3a - If the MBLH member is already involved with a lesson or hire at the day/time of the proposed motorboat 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).

3b - Otherwise, the system checks whether any of the club motorboats 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).

3b(i) - If there are no available club motorboats 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)

3b(ii) - If any of the club motorboats are available, then the names of the available motorboats are displayed to the console and the User is prompted to select which motorboat 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.

Use case #5: Display motorboat bookings

1 - The system allows the User (i.e., the administrator) to select a motorboat from a list of all motorboat names owned by the club.

2 – On selection of a motorboat name, the system displays a weekly schedule (i.e., all days and times) of bookings for the selected motorboat, indicating if the motorboat is booked for a lesson, booked for hire, or still available for each day/time slot across the week.