CST2110 Individual Programming Assignment #2 (RESIT)

Deadline for submission **16:00, Friday 30th July, 2021**

General information

You are required to submit your work via the dedicated Unihub assignment link in the 'resits and deferrals' folder 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.

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.

When the ZIP file is extracted, there should be three folders (viewed in Windows explorer) representing three independent NetBeans projects as illustrated by Figure 1 below.

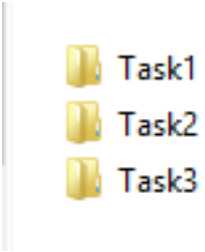

Figure 1: When the ZIP file is extracted there should be three folders named Task1, Task2 and Task3

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

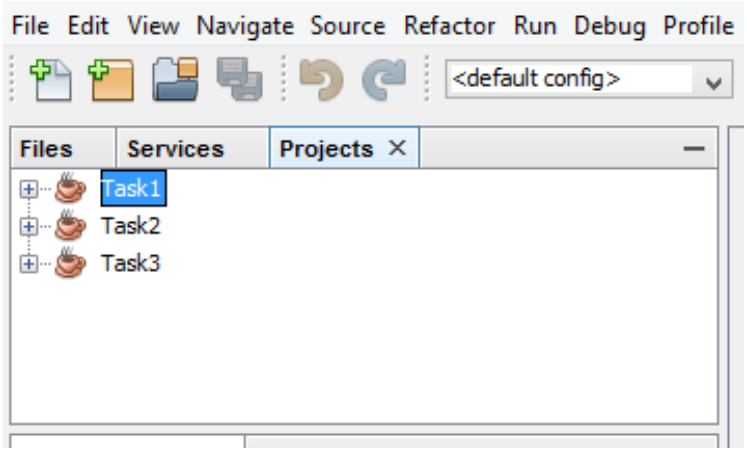

Figure 2: Each task must be a separate NetBeans Project

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 (20 marks)

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 James Bond movies (by Eon Productions). The data file is called *bond-movies.txt*. The data file must not be altered and should be considered as a *read-only* data file.

Within the data file, there are 24 entries (each on a new line) that represent a Bond movie. Each entry contains a series of data fields representing the following information (in order): the movie title and year of release, the actor who played James Bond, the actor who played the main villain and the name of the villain character, the actress who played the main heroine and the name of the heroine character, the movie director, the composer of the movie theme music, the locations around the world that are depicted in the movie, and some of the vehicles that are featured in the movie.

You are required to implement a Java class to represent a James Bond movie with respect to this data set. The program should parse the data file, create an object for each Bond movie, and store all the objects into a suitable collection. Figure 3 provides a partial UML class representation of the class that you will need to implement. It indicates required data members and accessor (i.e., *getter*) methods that map to those data members, and a *toString()* method. It is left to you to determine class data types and how the Bond movie objects should be initialised.

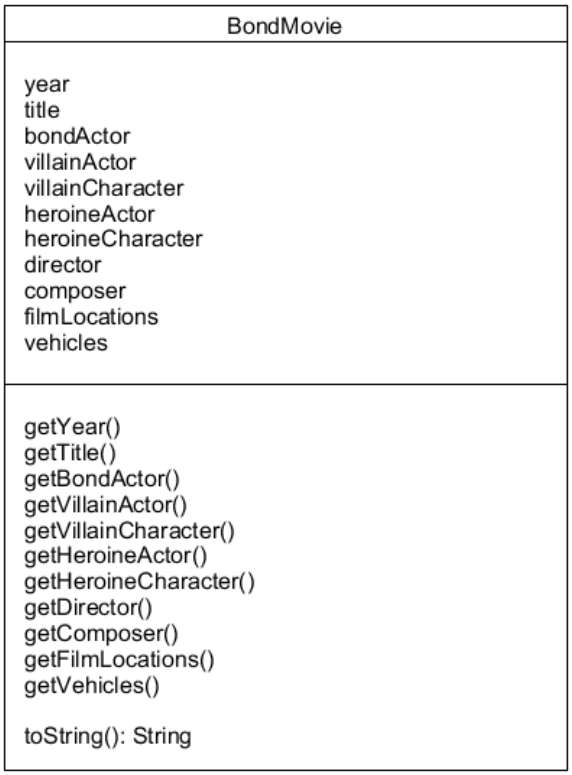

Figure 3: UML class specification for a *BondMovie* class

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 sort.

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

```
----------------------
Main 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 a single Bond movie details is selected (i.e., the 'select' menu option), it must result in the invocation of the *toString()* method for that particular *BondMovie* object. You are required to utilise a *StringBuilder* object when implementing the *toString()* method for the *BondMovie* class.

Task 2 (25 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 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 gym training session bookings. Below is an initial description of the system domain with a few very simple use cases (i.e., just simple, natural language statements).

System domain

Each gym in the chain has a unique name and employs at least five trainers. The gyms hold training sessions for members. A training session takes place in a single gym. Each trainer is identified by name, and employed by a single gym, and each member (identified by name and membership number) belongs to a single gym in the chain. Each member is assigned to a single trainer who works at the member's gym. A trainer can have at most 25 assigned members. Training sessions are of two types: class sessions and personal sessions. A class session lasts for 1 hour and focuses on a particular activity such as pilates, body balance, zumba or yoga. Each class session is taken by one trainer employed by the gym that holds the session and has a limit on the number of members who can book for it. After a gym has arranged a class session for a particular date and time and allocated the trainer taking it, members can book for the class session on a first come, first served basis so long as the number of members booked for the session does not exceed the limit for that session. Members can book for class sessions in any of the gyms in the chain. A personal session is for a single member and is always provided by the trainer assigned to that member. Personal sessions can be arranged for any time when both the trainer and the member are not involved in any other training session. There is no specified limit to the number of class or personal sessions that trainers or members can take part in. Trainers and members can each be involved in only one session of either type at any one time.

Use cases:

1. **Book personal training session for member**. The user of the software system provides the member's name and number, the date, start time and duration of the personal session. If the member or the trainer to whom the member is assigned is already involved with a session at a time that overlaps with the period of the proposed personal session, the system informs the user and takes no further action; otherwise, the system displays the trainer's name and records that the member has a personal session with the trainer, and records the date, time, and duration of the session.

2. **Book member on class session.** The user of the software provides the name and number of the member, the name of the gym, and the name of the session. The number of members already booked for the class session at the selected gym must be less than the limit for that session. If the member is already involved with a session at the given time, the system informs the system user and takes no further action; otherwise, the system records that the member has booked for the class session.

3. **List trainer's classes for given date.** The user of the software system identifies a trainer by entering a name and provides a date. For the trainer that matches the name, the system displays information about every class session that the trainer is taking on that date.

4. **List member bookings.** The user of the software system identifies a member by their membership number. For the member that matches, the system displays the member's name and information about every class and/or personal training session that has been booked for that member.

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

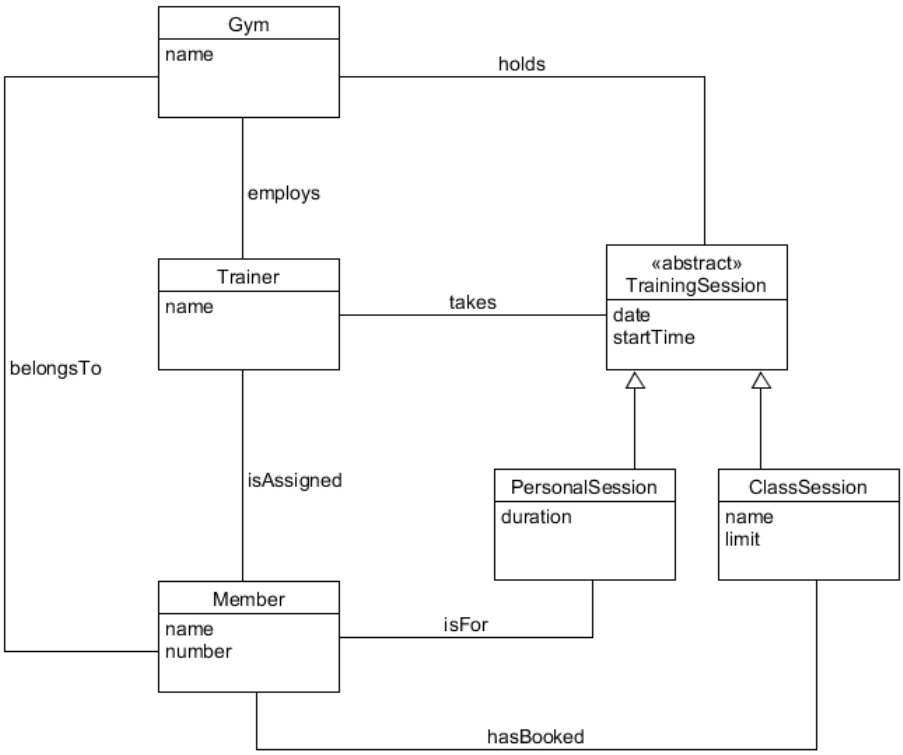

Figure 4: Class model of the gym booking domain.

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 use cases can be tested using the console-based User interface, your application should pre-load some 'dummy' data into the application, but also allow information to be input (use cases 1 and 2) during runtime. Pre-loaded data should be achieved by 'hard-coding' data within the Java program. It should **not** use an external database link (e.g., such as an SQL database). All pre-coded 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.

Your program will be assessed according to overall design and implementation, and the inclusion of an appropriate user-interface to allow the use cases to be tested fully. Some consideration will also be given to applications that demonstrate good encapsulation of system components.

Task 3 (15 marks)

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 event handling.

This program must be coded as a JavaFX program following the approach described in 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 electronic point of sale food menu touchscreen simulation. The required design is illustrated in Figure 5 below.

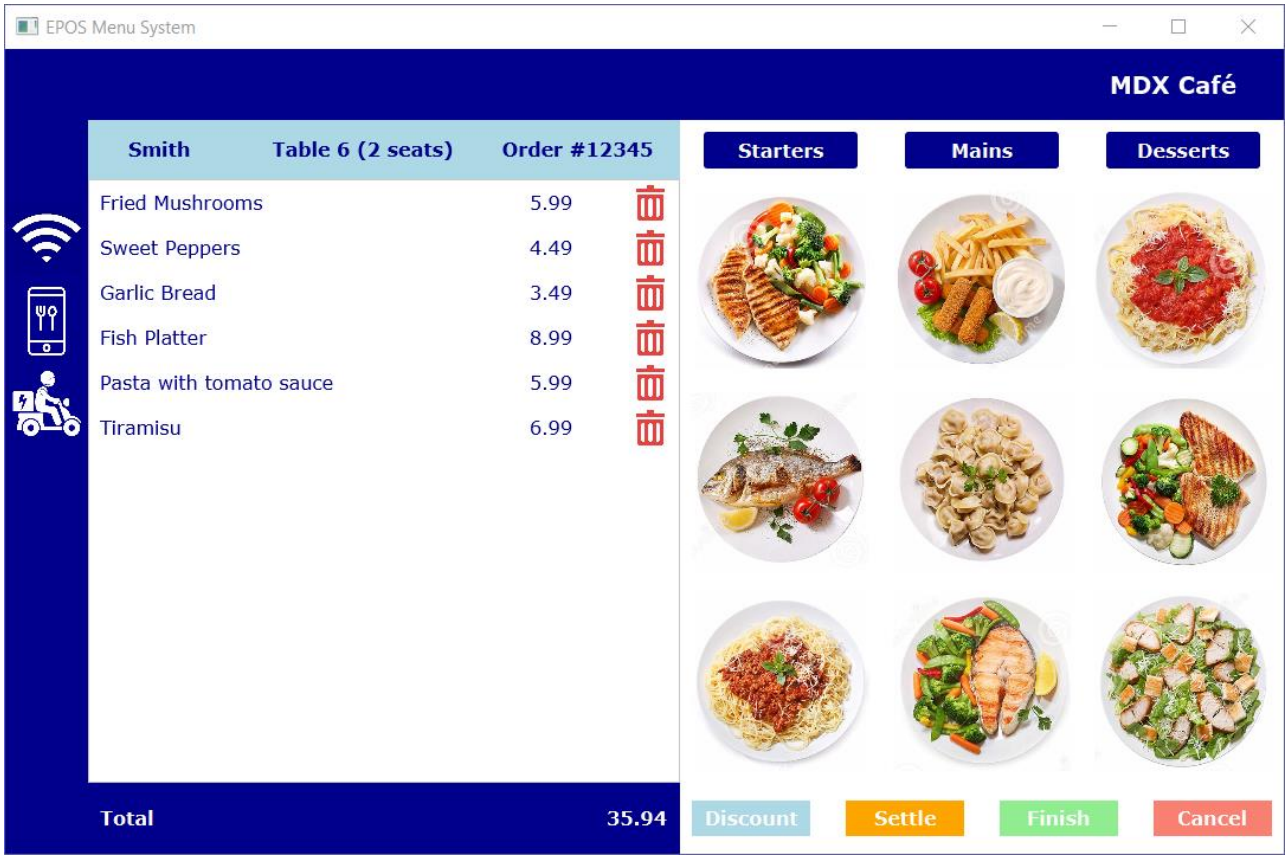

Figure 5: EPOS food menu simulation user interface

The application comprises a layout design which has two main interactive regions: a food selector region which presents images of food options (a fixed menu of nine choices for each of the three courses) and an order listing which is dynamically populated with food selections. This listing also provides a function to delete an ordered food item and keeps a running total of the order bill. 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 Unihub space.

a) Write a JavaFX program that replicates the GUI layout and design as illustrated in Figure 5. Styling of widgets must be achieved programmatically 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.

b) With reference to the regions in the display, User interaction should be as follows:

- The 'starter', 'mains' and 'desserts' buttons (or selectable regions) should toggle between three screens, each presenting nine food items as images. Images have been provided in the template project, but you are free to use your own food images provided you adhere to copyrights etc.

- To select an individual food item, the User should click on the relevant image with the mouse. The selected food item should then appear (as text) on the order list in the left of the display, along with a nominated price (which you can set as you wish) and a 'delete' icon (e.g., the red bin icon is provided). Each time a food item is added to the food order list, the overall order total should be updated.

- If the User decides to remove a food item from the food order list, then this functionality should be provided by selecting the delete icon that is positioned next to that food item. The food item should no longer be visible in the food order list and the food order total should be updated accordingly.

- If the food order list grows beyond the displayable area, then the list should scroll appropriately. Again, there are several ways this effect can be achieved, but we would recommend using a *ListView* component (see *kortext*, Chapter 16).

Appendix A: Console interaction examples for Task 1

Option 1: list movies

On selecting option 1, the User should be presented with a neatly formatted listing of the Bond movies. The data displayed should be ordered by the year of release, and include the movie title, the name of the actor that played James Bond, the movie director, and the name of the person who composed the theme music. To select a single Bond movie to view its full details, the User should be prompted to enter the year of release (i.e., option 2).

Year	Title	Bond Actor	Director	Composer
1962	Dr. No	Sean Connery	Terence Young	Monty Norman
1963	From Russia with Love	Sean Connery	Terence Young	John Barry
1964	Goldfinger	Sean Connery	Guy Hamilton	John Barry
1965	Thunderball	Sean Connery	Terence Young	John Barry
1967	You Only Live Twice	Sean Connery	Lewis Gilbert	John Barry
1969	On Her Majesty's Secret Service	George Lazenby	Peter R. Hunt	John Barry
1971	Diamonds Are Forever	Sean Connery	Guy Hamilton	John Barry
1973	Live and Let Die	Roger Moore	Guy Hamilton	George Martin
1974	The Man with the Golden Gun	Roger Moore	Guy Hamilton	John Barry
1977	The Spy Who Loved Me	Roger Moore	Lewis Gilbert	Marvin Hamlisch
1979	Moonraker	Roger Moore	Lewis Gilbert	John Barry
1981	For Your Eyes Only	Roger Moore	John Glen	Bill Conti
1983	Octopussy	Roger Moore	John Glen	John Barry
1985	A View to a Kill	Roger Moore	John Glen	John Barry
1987	The Living Daylights	Timothy Dalton	John Glen	John Barry
1989	License to Kill	Timothy Dalton	John Glen	Michael Kamen
1995	GoldenEye	Pierce Brosnan	Martin Campbell	Eric Serra
1997	Tomorrow Never Dies	Pierce Brosnan	Roger Spottiswoode	David Arnold
1999	The World Is Not Enough	Pierce Brosnan	Michael Apted	David Arnold
2002	Die Another Day	Pierce Brosnan	Lee Tamahori	David Arnold
2006	Casino Royale	Daniel Craig	Martin Campbell	David Arnold
2008	Quantum of Solace	Daniel Craig	Marc Forster	David Arnold
2012	Skyfall	Daniel Craig	Sam Mendes	Thomas Newman
2015	Spectre	Daniel Craig	Sam Mendes	Thomas Newman

Option 2: select movie

On selecting option 2, the User should be prompted to enter the year of release of a listed Bond movie (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 movie should be displayed, as illustrated below. This should be achieved by invocation of the relevant *BondMovie* object's *toString()* method. Console output should be neatly formatted.

```
----------------------
Main menu
----------------------
List ................1
Select ..............2
Sort ................3
Exit.................0
----------------------

Enter choice:> 2

Enter year of movie:> 2015

Title:    Spectre
Bond:     Daniel Craig
Villain:  "Blofeld" played by Christoph Waltz
Heroine:  "Madeleine" played by Lea Seydoux
---------------------------
Locations depicted in movie:
---------------------------
Mexico
England
Italy
Austria
Morocco
Vatican City
---------------------------
Vehicles featured in movie:
---------------------------
Aston Martin DB10
Aston Martin DB5
Rolls-Royce Silver Wraith
Mercedes S-Class
Range Rover Sport


----------------------
Main menu
----------------------
List ................1
Select ..............2
Sort ................3
Exit.................0
----------------------

Enter choice:>
```

Option 3: sort menu

On selecting option 3, the User should be presented with a sub-menu that allows to sorting options as illustrated below (as well as an option to return to the main menu):

```
------------------------
Sort options
------------------------
Sort by director.......1
Sort by composer.......2
Back to main menu......0
------------------------

Enter choice:>
```

On selecting the 'director' option, the User is presented with a list of movie information (director, title, year of release and Bond actor) ordered by movie director in alphabetic order. In addition, on those occasions that a particular director has directed several Bond movies, the films should be ordered by movie title alphabetically as illustrated below:

```
-------------------------------------------------------------------------------
| Director           | Movie title                    | Year | Bond Actor     |
-------------------------------------------------------------------------------
| Guy Hamilton       | Diamonds Are Forever           | 1971 | Sean Connery   |
|                    | Goldfinger                     | 1964 | Sean Connery   |
|                    | Live and Let Die               | 1973 | Roger Moore    |
|                    | The Man with the Golden Gun    | 1974 | Roger Moore    |
-------------------------------------------------------------------------------
| John Glen          | A View to a Kill               | 1985 | Roger Moore    |
|                    | For Your Eyes Only             | 1981 | Roger Moore    |
|                    | License to Kill                | 1989 | Timothy Dalton |
|                    | Octopussy                      | 1983 | Roger Moore    |
|                    | The Living Daylights           | 1987 | Timothy Dalton |
-------------------------------------------------------------------------------
| Lee Tamahori       | Die Another Day                | 2002 | Pierce Brosnan |
-------------------------------------------------------------------------------

Etc...
```

On selecting the 'composer' option, the User is presented with a list of movie information (composer, title, year and director) ordered by movie composer in alphabetic order. In addition, on those occasions that a particular composer has written the theme music for several Bond movies, the films should be ordered by director alphabetically as illustrated below:

```
-------------------------------------------------------------------------------
| Composer        | Movie title                    | Year | Director          |
-------------------------------------------------------------------------------
| Bill Conti      | For Your Eyes Only             | 1981 | John Glen         |
-------------------------------------------------------------------------------
| David Arnold    | Die Another Day                | 2002 | Lee Tamahori      |
|                 | Quantum of Solace              | 2008 | Marc Forster      |
|                 | Casino Royale                  | 2006 | Martin Campbell   |
|                 | The World Is Not Enough        | 1999 | Michael Apted     |
|                 | Tomorrow Never Dies            | 1997 | Roger Spottiswoode|
-------------------------------------------------------------------------------
| Eric Serra      | GoldenEye                      | 1995 | Martin Campbell   |
-------------------------------------------------------------------------------

Etc...
```