**CST2110 Assessment Part 2**

**Deadline for submission** **Tuesday 9th April, 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.

## Task 1

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 contains information about literature prize winners. The data file is called *literature-prizes.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 laureates (i.e., literature prize winners) from 1901 to 2022. Each entry in the file contains a series of data fields representing the following information: the year of the prize, the name of the award-winning laureate, the year the laureate was born and the year the laureate died if applicable, the nationality of the laureate (note that some laureates are dual nationality), the language or languages in which the laureate has produced written work, a citation about the laureate's contribution to literature that merited a prize for that year, and the writing genres that the laureate is associated with. Note that for some years multiple prizes were awarded and some years no prize was awarded.

You are required to implement Java classes to represent the literature prize winning 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 necessary domain entities 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 partial UML class representation of two classes that you must implement to represent the data. The class model indicates data members for both a *LiteraturePrize* and a *Laureate* class. The *LiteraturePrize* class is linked to the *Laureate* class via composition (aggregation). It is left to you to determine what methods the classes should contain, as well as how the respective objects should be initialised. Both classes should implement an appropriate *toString()* method to format console output appropriately when invoked as indicated below.

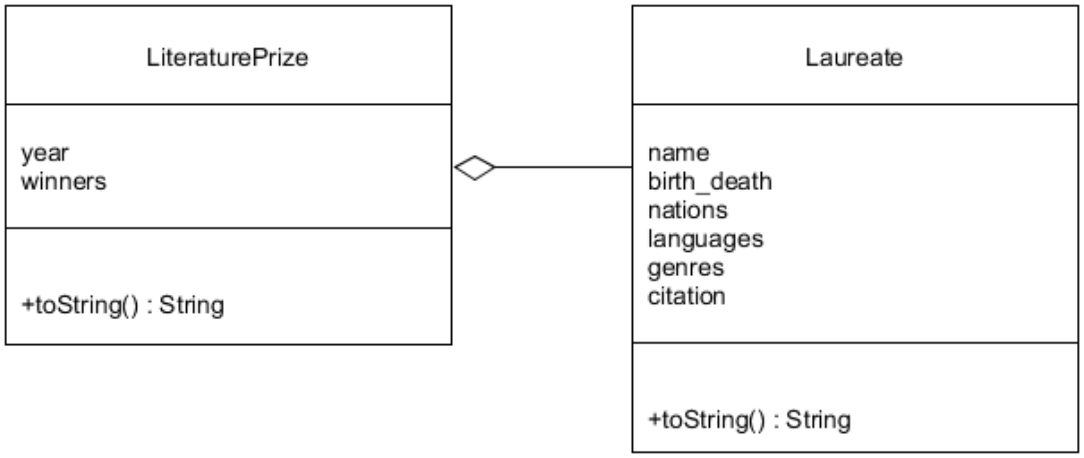


Figure 1: Partial UML class model for literature prize information.

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 three other options: list, select and search.

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

```
----------------------
Literature 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 prizes for a given year 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 classes illustrated in Figure 1.

An assessment rubric is provided in the module handbook.

Task 1 has a provisional weighting of 30 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 console user interface. The required Java application relates to a proposed software system to manage some aspects of a swimming school. Below is an initial description of the system domain and several use cases (structured as natural language statements).

**Domain description**

The swim school operates on three weekday evenings (Monday, Wednesday, and Friday) from 17:00-20:00. Each session is a group class (or lesson) of 30 minutes duration for a maximum of 4 swim students. Each group class has a designated start time (the first classes start at 17:00, with the final lesson at 19:30). The lessons are categorised into three levels: novice, improvers and advanced. At any one time, there are three classes taking place simultaneously in the swimming pool, i.e., one for each of the three levels. Each class is taken by one allocated instructor. Instructors can be allocated to several sessions per week. All instructors are qualified to teach at all levels. Swim students can only attend one session per week which is specific to their designated level. In time, the instructor will make an assessment that is specific to a potential transition from one level to the next (i.e., from novice to improver, or from improver to advanced). If the swim student effectively passes that assessment, then the instructor will record this fact and the student will be eligible to transfer to the next level. A waiting list is maintained for both students who have passed an assessment and wish to transfer to the next level, and for new students who wish to join the school (the school only accepts new students who wish to enter at the novice level). If there is currently no availability to move a student who has passed an assessment to the next level, then that student will remain in their current class and their name will be added to the waiting list. A new student who wishes to join the school at novice level can be offered a place if there are spaces available but will be put on the waiting list if not.

The swim school is authorised to award Amateur Swimming Association (ASA) qualifications. The school assesses and awards two types of qualification: distance swim certificates, and personal survival medals. Each qualification type comprises multiple levels. Novice level swim students can achieve 5-metre, 10-metre, and 20-metre awards. A 20-metre distance swim qualification is used as an assessment for a student to upgrade from novice to improver. The improver level swim students can attempt longer swim distances (100m, 200m, and 400m). To be upgraded from the improver level to the advanced level, a swim student must attain the 400m distance qualification, and the same waiting list system also applies. Advanced level swim students can attempt distance awards of 800m, 1500m and 3000m. Only advanced level swim students are assessed for personal survival qualifications, which have three categories: Bronze, Silver, and Gold.

**Use cases:**

1. View swim student information

The system prompts the User (i.e., the administrator) to select a swim student from a list, sorted in alphabetical order by student name. The listing of swim students indicates the level of the student. Following the User selection of a swim student, the system displays the day and time of the swim class that the student attends (if applicable) and the name of the instructor that takes that class. If the student is a new (novice) student on the waiting list, then the system displays that fact. In addition, the system displays a list of the distance swim qualifications awarded to that student (if applicable) by the swim school, including the name of the swim instructor who awarded the qualification. If the student is at the advanced level, the system also displays any personal survival qualifications gained by the student, along with the name of the awarding instructor.

2. View swim lesson details

The system prompts the User (i.e., the administrator) to select the day, time, and level of the swim class from appropriate lists displayed by the system. The system responds by displaying the name of the instructor taking that class, the list of students currently allocated to that class, and a message indicating if the class is currently full, or how many spaces are currently available for that class.

3. View instructor schedule

The system prompts the User (i.e., the administrator) to select a swim instructor from a list, sorted in alphabetical order by instructor name. Following the User selection, the system displays all the selected instructor's allocated classes for the week. For each allocated session, the system displays the day, time, and level of the swim lesson (i.e., novice, improver or advanced), and the names of the swim students taking the swim lesson with the selected instructor at that lesson.

4. Add new swim student

The User (i.e., administrator) provides the name of a new student to the system. The system then displays a weekly schedule of classes for the novice level. The display indicates if there are any spaces available in each class or if it is currently full. If there are classes with availability, the administrator can select one of those sessions (by selecting the day and time) and the new student is allocated to that group, which is recorded by the system, and an appropriate confirmation message is displayed. If there is no space available (on the day and time that the student wishes to attend), then the system will allow the User to add the new student to the waiting list (which is recorded by the system and an appropriate message is displayed).

5. Award swim qualification

The system prompts the User (i.e., the administrator) to select a swim instructor from a list, sorted in alphabetical order by instructor name. Following the User selection, the system prompts the User to select a swim student from a list, sorted in alphabetical order by student name. The display of swim students indicates the level of each student listed. If the selected student is an advanced level swim student, the User is prompted to indicate if the award is a distance swim qualification or personal survival qualification. In either case, the User is prompted to select a qualification from an appropriate list of awards (not already achieved by that student). The system then records the award for the student as assessed by the selected instructor. If the student is not an advanced student, the User is prompted to enter a swim distance award from a list of qualifications (not already achieved by that student). The system then records the award for the student as assessed by the selected instructor.

If the swim student is currently novice level and the distance qualification awarded is 20 metres, or the swim student is intermediate level and the distance qualification awarded is for 400 metres, then the system will automatically change the status of the level of the student (which is recorded) and add the student to the waiting list (to transfer group).

6. Move swim student from waiting list

The system allows the User (i.e., the administrator) to select a swim student from the waiting list of swim student names that is displayed to the console. The list of names that is displayed is organised according to the following criteria: first all those new students waiting to join a novice class are listed, followed by existing students currently in a novice class waiting to join an improver class, and then existing students currently in an improver class waiting to join an advanced class. Once the swim student has been selected, the system displays a weekly schedule of classes for the required level of the selected swim student. The display indicates if there are any spaces available in each class (or if it is currently full). If there are classes with availability, the User can select that session (by selecting the day and time) and the waiting swim student is allocated to that group and removed from the waiting list, which is recorded by the system and an appropriate message is displayed. If the student who has been moved is an existing student currently in another class, then they are removed from that class. If there is no availability to transfer to the required level, the User is prevented from doing so and an appropriate message is displayed.

Your task is to design a Java 8 program for the swim school 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.

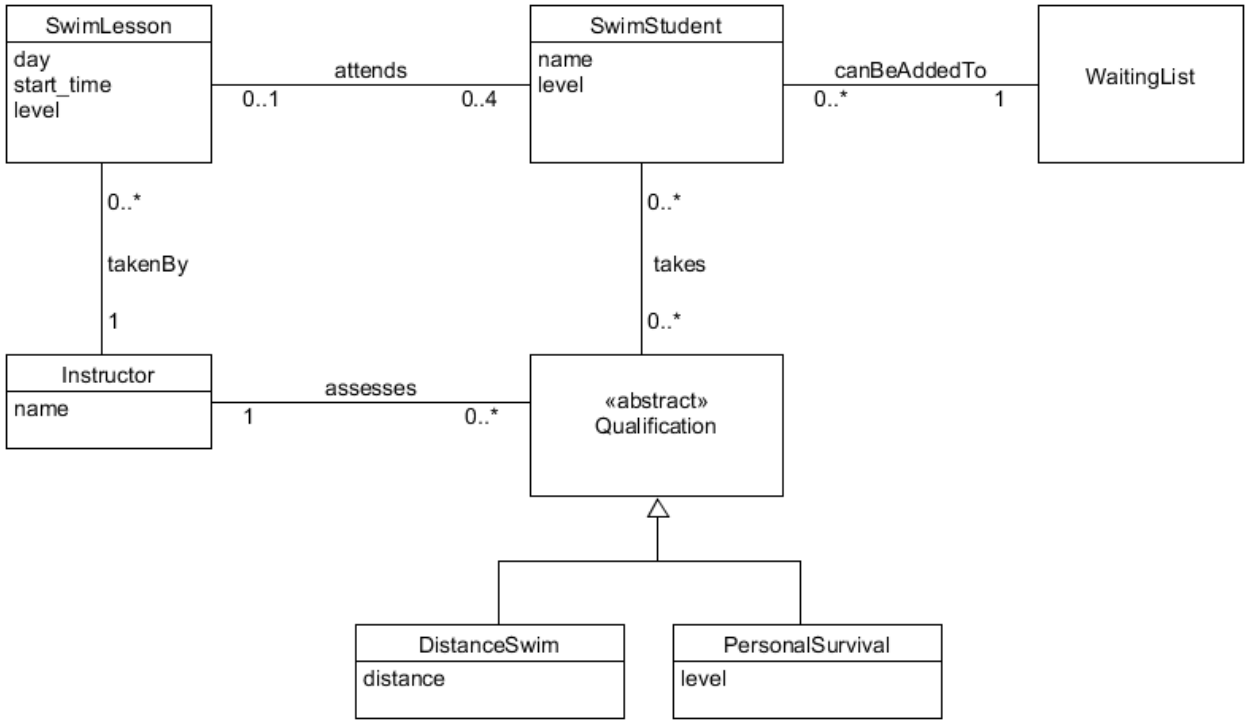


Figure 2: Class model of the swim school system.

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 4 and 5) 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.

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

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 literature prize winners

On selecting option 1, the User should be prompted to enter the range of years for which they wish to list the award winner. An example is illustrated below. The User is prompted to enter the start and end year for the listing. The program should be robust and ensure that a correctly formatted year is entered (and that is within the range of the data set). With a valid year range entered, the User should be presented with a neatly formatted listing of the names of the laureates who were awarded a prize ordered by the year in which they won the prize.

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

Enter choice> 1

Enter start year > 1910
Enter end year > 1925
```

| Year | Prize winners (and associated nations) |
|---|---|
| 1910 | Paul von Heyse [Germany] |
| 1911 | Maurice Maeterlinck [Belgium] |
| 1912 | Gerhart Hauptmann [Germany] |
| 1913 | Rabindranath Tagore [India] |
| 1914 | NOT AWARDED |
| 1915 | Romain Rolland [France] |
| 1916 | Verner von Heidenstam [Sweden] |
| 1917 | Karl Adolph Gjellerup [Denmark], Henrik Pontoppidan [Denmark] |
| 1918 | NOT AWARDED |
| 1919 | Carl Spitteler [Switzerland] |
| 1920 | Knut Hamsun [Norway] |
| 1921 | Anatole France [France] |
| 1922 | Jacinto Benavente [Spain] |
| 1923 | William Butler Yeats [Ireland] |
| 1924 | Władysław Reymont [Poland] |
| 1925 | George Bernard Shaw [United Kingdom,  Ireland] |

If the literature prize was not awarded for a given year (e.g., 1914), then the display should indicate this. Otherwise, the listing should display the name of the winning laureate and the nationalities, or nations, associated with that laureate. Note that there can be more than one laureate that received a prize for a given year (e.g., 1917), and a laureate can be associated with more than one nation.

**Option 2: select year**

To select a single individual year to view its full details, the User should be prompted to enter the year of the award (i.e., option 2).

On selecting option 2, the User should be prompted to enter the year of the prize (that was displayed when option 1 was chosen). 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 prize winner(s) for that year as illustrated below. This should include the name of the laureate(s), the year they were born, the year they died if applicable, the languages that the laureate produced written work (as a list if more than one), the writing genre(s) associated with that laureate (as a list if more than one), and the associated citation for that laureate. The console output should be achieved by invocation of the relevant *toString()* method.

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

Enter choice:> 2

Enter year of prize > 1916

-------------------------------------------------------------------------------------------
| Winner(s)                | Born | Died | Language(s)           | Genre(s)              |
-------------------------------------------------------------------------------------------
| Verner von Heidenstam    | 1859 | 1940 | Swedish               | poetry                |
|                          |      |      |                       | novel                 |
-------------------------------------------------------------------------------------------
|                                      Citation:                                          |
|                                                                                         |
|         "in recognition of his significance as the leading representative of            |
|                            a new era in our literature"                                 |
-------------------------------------------------------------------------------------------
```

Note that in some years, there were more than one laureate, such as in 1917 as shown below. In these cases, all laureates should be displayed as illustrated.

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

Enter choice:> 2

Enter year of prize > 1917
```

| Winner(s) | Born | Died | Language(s) | Genre(s) |
|---|---|---|---|---|
| Karl Adolph Gjellerup | 1857 | 1919 | Danish<br>German | poetry |
| Citation: | | | | |
| "for his varied and rich poetry, which is inspired by lofty ideals" | | | | |
| Henrik Pontoppidan | 1857 | 1943 | Danish | novel |
| Citation: | | | | |
| "for his authentic descriptions of present-day life in Denmark" | | | | |

Note that some of the later laureates are still alive, and so the 'died' field should reflect this, as illustrated below.

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

Enter choice:> 2

Enter year of prize > 2010
```

| Winner(s) | Born | Died | Language(s) | Genre(s) |
|---|---|---|---|---|
| Mario Vargas Llosa | 1936 | ---- | Spanish | novel<br>short story<br>essay<br>drama<br>memoir |
| Citation: | | | | |
| "for his cartography of structures of power and his trenchant images of the individual's resistance, revolt, and defeat" | | | | |

**Option 3: search**

On selecting option 3, the User should be prompted to enter a 'search string' to match against the writing genres for all years of the literature prizes. All searches should be case insensitive. The program should return a listing of any winners for whom an associated genre matches the search string, with the section(s) that match displayed in upper-case. The returned matches should first display the name of the laureate, followed by the list of genres associated with that laureate (with the matching text in uppercase) and the year that the award was won by the laureate. The listing should be presented to the console in alphabetical order i.e., sorted by the laureate name.

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

Enter choice:> 3

Enter search term for writing genre > biography
```

| Name | Genres | Year |
|---|---|---|
| Annie Ernaux | autoBIOGRAPHY, novel | 2022 |
| Doris Lessing | novel, short story, memoir, autoBIOGRAPHY, drama, poetry, essay | 2007 |
| Orhan Pamuk | novel, screenplay, autoBIOGRAPHY, essay | 2006 |
| Pearl Buck | novel, BIOGRAPHY | 1938 |