RMIT UNIVERSITY

School of Science

# COSC1295 Advanced Programming, S1, 2020

## Assignment Part 2: UniLink system - Java FX UI Implementation and Data Persistence

| | |
|---|---|
| | Assessment Type: Individual assignment; no group work. Submit online via Canvas → Assignments → Assignment 2. Marks are awarded for meeting requirements as closely as possible according to assignment specifications and the supplied rubric. Clarifications/updates may be made via announcements/relevant discussion forums. |
| | Due date: end of Week 13, 23:59pm Sunday 7th June 2020. Late submissions are handled as per usual RMIT regulations - 10% deduction (10 marks) per day. You are only allowed to have 5 late days maximum unless special consideration has been granted. **You must demo your assignment via Microsoft Team in Week 14 according to a demo schedule which will be announced later.** |
| | Weighting: 100 marks (45% of your final semester grade) |

### 1. Overview

You are required to use Java SE and JavaFX to develop a Graphical User Interface (GUI) for the UniLink system created in Assignment 1.

***This is an individual assignment. Group work is not allowed.***

This assignment is design

1. Enhance your ability to build a Graphical User Interface using JavaFX
2. Practise implementation of various GUI event handler
3. Read from and write to a database using Java JDBC te
4. Incorporate text file handling in your program to impo

### 2. Assessment Criteria

This assessment will determine your ability to implement Object-Oriented Java code according to a written specification and incorporating specific design patterns. In addition to functional correctness (i.e. getting your code to work) you will also be assessed on code quality. Specifically:

- You should aim to provide high cohesion and low coupling as covered in this course.
- You should aim for maximum encapsulation and information hiding.
- You should take advantage of polymorphism wherever possible when invoking methods upon objects that have been created.
- You should rigorously avoid code duplication.
- You should comment important sections of your code remembering that clear and readily comprehensible code is preferable to a comment.

### 3. Learning Outcomes

This assessment is relevant to the following Learning Outcomes:

**CL01**: use the Java programming language in the implementation of small to medium sized application programs that illustrate professionally acceptable coding and performance standards.

**CL02**: demonstrate knowledge of the basic principles of the object oriented development process and apply this understanding to the analysis and design of solutions for small to medium scale problems.

**CLO3**: describe and apply basic algorithms and data structures, in particular simple searching and sorting of data stored in data structures and manipulating data.

**CLO4:** implement basic event-driven programming and graphical user interfaces using Java.

## 4. Assessment details

**General Implementation Requirements**

- All information displayed to the user and all user interactions must be done via the GUI.
- Your GUI must be well structured and well formatted and meet the requirements shown in the following sections. Marks will be deducted for poor GUI designs.
- Any user inputs via the GUI should be validated.
- Marks will be allocated to proper documentation and coding layout and style.

**NOTE:**

To make it faster and easier to build GUI, you are allowed to use **JavaFx Scene Builder**.
https://gluonhq.com/products/scene-builder/

**Task Specifications**

NOTE: Carefully read the following requirements. In addition, regularly follow Canvas assignment discussion board for assignment related clarifications and discussion.

**Packages and Organisation of Code**

You must use the following packa rfaces:

- ***view***: contains all your G
- ***controller***: contains all y
- ***model***: contains the main business logic of your application, all ocess data (Post, Reply, Event, Sale, Job ...), all exception classes and all database and file han
- **main**: contains the startup class, i.e. UniLinkGUI.java, which is t application.

You can use sub-packages inside the packages shown above.

**Data Generation**

You are require to generate data for 6 posts, including 2 event posts, 2 open sale posts and 2 open job posts. Each event, sale and job post has two replies with responder id of your choice, and reasonable offer values for the sale and job posts.

For this assignment, each post must have a corresponding image, which should have a moderate size (from 100KB to 250KB). You must keep all images in a folder named **images** which is a direct sub-folder of your assignment project. If a post has no corresponding image, a default image with the phrase "No Image Available" should be used (please see this example).

This data must not be hard-coded into your program. You must store this data in a database in advance and when your program is running, this data must be read from database. See the section Using Database For Data Persistence below for more details.

**Exception Handlings**

**Creating Custom Exception Classes**

You are required to create at least three exception classes such as PostNotFoundException, PostClosedException, InvalidOfferPriceException... or any other custom exception class suitable for your application to represent various exceptional cases that can occur when your program executes. All those custom exception types should allow appropriate error messages to be specified when exception objects are created.

RMIT UNIVERSITY

**Generating and Handling Exception Objects**

To apply the custom exceptions in your program, you will need to look for areas in your program where you need to handle various cases such as when there is an invalid post id, or when a post is already closed or when there is an invalid user input... Then you should generate appropriate exception objects, throw and catch those exception objects correctly in your code.

All exceptions need to be caught and handled in an appropriate manner. Users should be able to see the error messages encapsulated in exception objects. In a GUI application, error messages must be shown to the users using dialog boxes. Your program must not show exception message in the console.

**Using Database For Data Persistence**

Every time your program is executed and terminated, data must be read from and save to a database. For this assignment, you must use an embedded HSQLDB (shown in the lecture) or SQLite database. You must keep all database files in a folder named **database** which is a direct sub-folder of your assignment project.

Your database must contain at least two tables, one table to store post information and another table to store replies information. It's not enough to just simply store the output of the toString methods of a post object or reply object. You must not store binary data directly in database. You need to create tables with multiple columns having appropriate data types (i.e. text and numerical data types) that are suitable for storing data in your application.

**NOTE:** No need to store images directly in the database. Only image file names should be stored as text in the database. Each image file name corresponds to an image file kept in the **images** folder which is a direct sub-folder of your assignment project.

For instructions about using HSQLDB with Eclipse, please visit this page
https://tinyurl.com/y7hty8tg

## Graphical User Interface (G

All user interaction with the UniLi ments for each component of the GUI are described below:

**NOTE:** To make it faster and easier to build GUI, you are allowed to use
https://gluonhq.com/products/scene-builder/

**Login Window**

This login window is the first window to appear when your program is executed. This window will reappear when users log out of the main window (more in Main Window section below).

The login window must contain the following components:

- A text field to allow the user to enter a username to log in. Username requirement is the same as in assignment 1 (for simplicity, no need for password).
- Log In button: after entering a valid username, the user clicks this button to log in, this login window will be closed and the main window will appear (see the Main Window section below)

This login window must be placed in the centre of the screen.

**Main Window**

See the image below for a suggestion about the components of the main window. For more information, please read the assignment specification carefully.
https://drive.google.com/file/d/1_gj9XGLzJPXkYJoqS13RK_rAUD9WLnpi/view?usp=sharing

The main window must have the following components:

Menu bar

The main window must contain a menu bar with the following two menus:

- A menu named Unilink having two menu items:
  - Developer Information: when user clicks on this menu item, a small window appears in the middle and on top of the main window, showing your student name and student number. User can click a button to close this window and return to the main window.
  - Quit UniLink: when user clicks on this menu item, your program will stop execution and quit after saving all data to database. Note: if this feature is already supported by default in a JavaFx application, then no need to implement again. You just make sure that when users quit your application, all data is saved to database.
- A menu named Data having two menu items:
  - Export: when user clicks on this menu item, all post and reply data is exported into a single text file
  - Import: when user clicks on this menu item, he or she can select a text file on your local computer to import post and reply data.
    (for more information, please see the **Using Text Files for Exporting and Importing Data** section below)

Toolbar

The toolbar must allow users to perform the following functionalities:

- Create a new post:
  - you need to include appropriate buttons that user can click on to create new posts of the three types of post (i.e. event, sale and job posts).
  - When user clicks on one of these buttons to create a new post, your program must display a window containing a form for user to fill in necessary details to create the new event or sale or job post.
  - Your progra our computer to upload to your program w s folder which is a direct sub-folder of your assi
- Filtering capabilities: s to allow the user to filter the list of posts displayed in the centre area in the main window. De
  - Filter by type: a drop-down list contains post type hen user selects an item in the list, the main post list is displayed accordingly. Fo s Event, then only Event posts are shown, when user selects All, then all posts ar more details about the list of posts are shown in the " Centre area of main window" section below.
  - Filter by status: a drop-down list contains post status values, i.e. All, Open, Closed. When user selects an item in the list, the main post list is displayed accordingly. For example, when user selects Closed, then only closed posts are displayed, when user selects Open, then only open posts are displayed.
  - Filter by creator: a drop-down list contains two items, i.e. My Post and All. When user selects My Post, then only the posts created by the current user are shown, if the user selects All, then all posts are shown.
- Log Out button: when the user clicks this button, the main window is closed and the login window reappears in the centre of the computer screen.

Centre area of the main window

The user id of the currently logged in user must be displayed at the top of the centre area so we know which user is currently logging in.

The centre area of the main window must contain a scrollable list of posts managed by your program. In this list, each post is displayed as a list item.

Each list item must contain the following information:

- A small image of the post.
- Common details of all types of posts such as post id, title, description, creator id, status.
- Details specific to each type of post:
  - event post: venue, date, capacity, attendee count, no need to show attendee list here.
  - sale post: highest offer, minimum raise

- note: asking price should only be visible in a post if the currently logged in user is the creator of the post, other users should not be able to see the asking price of a sale post.

  - job post: proposed price, lowest offer
  - different background color should be used for different type of post (for example: light cyan for event post, light pink for sale post, light yellow for job post) to make it easier to recognise the post type.

- Reply button: the user clicks this button to reply to a post (please note that a post creator should not be allowed to reply to his or her own post)
  - If the post is an event post, this button should be implemented as a Join button which the user can click on to join the event. Your program must display appropriate response messages when the user clicks this Join button, in cases when the join request is accepted and when the event is full.
  - If the post is a sale or job post, then when the user clicks Reply, your program must allow the user to enter an offer value if the post is open. Your program must display appropriate response messages when an offer value is accepted or rejected.
  - Your program must not allow post creators to reply to their own posts. This is our assumption in this assignment.
- More Details button: this button should only be visible in posts created by the currently logged in user. When the user clicks on this button, the main window will be closed and the Post Detail window will open. Your program must not show both the main window and post detail window at the same time. For more details, see the Post Detail Window section below.

**Note:** post replies must not be displayed in the main window. Post replies must be displayed in the post detail window. Only the creator of a post can view replies to that post.

To display post details in the GUI, you must use JavaFX User Interface Controls such as ImageView, Label, Buttons..., rather than just a text area showing the output of the getPostDetails() method.

**Post Detail Window**

See the image below for a sugges [illegible] re information, please read the assignment specification carefully
https://drive.google.com/file/d/1qAtYKlvrAGm-3_wFz_M7W1leuSX7sm

Only the creator of a post can open this window to see more details ab [illegible] lies to that post. If there is no reply, then the post creator is allowed to edit his or her post details. If t [illegible] then the post creator is no longer allowed to edit any post detail.

Edit post details

In this post detail window, all post details are displayed, including a larger image associated with the post. If there is no reply, your program must enable the post creator to edit post details (post id should not be editable) and upload another image which will replace the existing image. After the post creator finishes editing, he or she can click a Save button, and all details will be saved. If there is one or more replies, it means that responders have  given replies based on existing details and the post creator is no longer allowed to edit any post details.

View replies

The post creator must be able to view all replies to the post in this post detail window

- Event post: your program must show the list of users participating in the event.
- Sale post: your program must use a table to display responder ids and offer values in descending order of offer values.
- Job post: your program must use a table to display responder ids and offer values in ascending order of offer values.

Close post

There must be a button to allow the creator to close the post. If a post is closed, it is no longer possible to re-open it.

Delete post

There must be a Delete button to allow the creator to delete the post. If the creator clicks this button, a confirmation dialog box is displayed.

- If the post creator clicks No to not delete the post, he or she returns to the post detail window.
- If the post creator clicks Yes to confirm the deletion, the post is deleted, the post detail window is closed and the post creator returns to the main window. The deleted post is no longer visible in the post list in the main window.

Back to main window

The post detail window must have a button to allow the post creator to go back to the main window. When the post creator clicks on this button, the post detail window is closed and the main window is reopened.

**Other GUI Requirements and Input Validation Requirements**

- You must use layout manager classes such as BorderPane, GridPane, HBox, VBox... to control the layout of the components in your GUI.
- You are encouraged to explore and use various JavaFX User Interface (UI) controls to implement your graphical interfaces for all the required functionalities. My suggestions for such UI controls are the Dialog class and its subclasses such as TextInputDialog, ChoiceDialog and Alert classes in the javafx.scene.control package.
- All user inputs via the GUI must be validated. Your program should not crash at any point given any user input.
- All error messages and messages from exception objects should be displayed in the GUI using JavaFX Alert classes in the javafx.scene.control package. Do not output any error message to the Console.
    - Note: it is acceptable to have some database log messages or JavaFx messages in the Console. But error messages related to your program must be displayed in the GUI.

**Using Text Files for Exporting and Importing Data**

Your UniLink GUI program must contain Export Data and Import Data menu items in the main menu bar to allow users to:

- export all post data of yo
- import all post data and e format as the exported data file.

You are free to choose your own format of the data file, as long as all da le text file and imported from a text file following your own data format.

When the user selects the Export Data menu item, your program should *ext file* named **export_data.txt**, which will be saved in a folder chosen by the user. (Hint: this feature can be implemented in your GUI program by using the DirectoryChooser class in the javafx.stage package).

When the user selects the Import Data menu item, your GUI program must display a FileChooser window (hint: FileChooser class is located in the javafx.stage package) to allow the user to select a text file with the same format as your exported data file. Your program must be able to read post and reply data from that text file. If a post has no corresponding image, a default image with the phrase "No Image Available" should be used.

NOTE: You don't need to export images. All images will still be kept in the **images** folder in your assignment project. Only image file names should be exported to the text file.

**Other Requirements**

- Although you are not required to use more than one class per task, you are required to modularise classes properly. No method should be longer than 50 lines.
- Your coding style should be consistent with Java coding conventions (http://www.oracle.com/technetwork/java/codeconventions-150003.pdf)

## 5. Referencing guidelines

You are free to refer to textbooks or notes and discuss the design issues (and associated general solutions) with your fellow students or on Canvas; however, the assignment should be your OWN WORK.

The submitted assignment must be your own work. For more information, please visit http://www.rmit.edu.au/academicintegrity.

Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students, internet or other resources without proper reference. Sometimes, students study and work on assignments together and submit similar files which may be regarded as plagiarism. Please note that you should always create your own assignment work even if you have very similar ideas.

Plagiarism-detection tools will be used for all submissions. Penalties may be applied in cases of plagiarism.

## 6. Submission format

You must submit a zip file of your project via Canvas.

You can submit your assignment as many times as you would like prior to  the due date. The latest submission will be graded.

You MUST NOT submit compiled files (*.class files). Check your submission carefully, make sure all java files have been included.

## 7. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods,
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.
RMIT University treats plagiarism as a very serious offence constituting misconduct.  Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly docu
- Copyright material from t
- Collusion between students

For further information on our policies and procedures, please refer to t

## 8. Assessment declaration

When you submit work electronically, you agree to the assessment declaration.