COMP 2613 Assignment 2

The assignment will continue on from assignment 1, and also building on the labs and reinforce the concepts and features of the java framework we’ve learned in weeks seven through ten. You’ll be adding data persistence and a graphical user interface for the Books app.

At the core of this application is your assignment 1 and labs. The assignment is a consolidation and extension of the requirements for the labs.

## Requirements

The design of Books2 must follow good object-oriented principles and practices as it did for Books.

The jar file containing your runnable code must be named Books2.jar and the jar file must run and display a JFrame in order to be marked. Needless to say, your code must compile. Compile-time warnings are considered errors and must be eliminated from your code by using the correct implementation or appropriate annotation tags.

As in assignment 1, all activity must be written to a log file named Books.log

Exceptions must be handled such that no stack traces are displayed in the console, but as mentioned above, a message will be logged explaining the cause of the error.

In the second half of COMP 2613 the topics are database, multithreading, graphical user, database, model-view-controller and other design patterns, and network programming. This assignment will touch on many of these.

Remove, or don’t call the printing of the reports to the console and to a text report that we had in assignment 1. No program arguments are required. Additional requirements include:

* Similar to your labs, the first time Books is run, the data will be read from same data files used in assignment 1.
* A separate Database class is used to manage the connection to the database.
* The data will be stored into appropriate tables in the database – as we will all share the same SQL Server environment, you must distinguish your tabled from others by prefixing the table name with your student number, ex. "A00123456\_".
* For each of the tables a separate DAO class is required; the DAO classes will contain the table creation, and create, read, update, and delete methods.
* To re-test the data loading functionality, you will need to use a method to drop all the tables in your database and restart your application.
* If collections are used, they are only used to temporarily store datasets.

The application has a graphical user interface (UI) that will be displayed when the user runs Books. The UI will always be displayed even if there are errors loading the data. Errors will be written to the application log files and will be messaged to the user in the form of a simple dialog box.

The follow menu must be implemented:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Notes | |
| File | JMenu | Contains the following items… |
| Drop | MenuItem | Open a simple message dialog asking the user if they want to delate all Book2 input data. If yes, then drop all the tables and exit Books2. When Books2 is restarted, it will reload the data. |
| Quit | MenuItem | Quits the application |
| Books | MenuItem | Contains the following items… |
| Count | MenuItem | [optional] Opens a simple message dialog showing the total book count |
| By Author | JCheckBox-MenuItem | [optional] If checked, sorts the list by author in ascending order. This option takes no action, but is used in conjunction with ‘List’ option. |
| Descending | JCheckBox-MenuItem | [optional] Sorts the list in descending order. The order is unaffected if “By Author” isn’t checked. This option takes no action, but is used in conjunction with ‘List’ option. |
| List | MenuItem | [optional] Opens a dialog which contains a list of the books |
| Customers | JMenu | Contains the following items… |
| Count | MenuItem | Opens a simple message dialog showing the total customer count |
| By Join Date | JCheckBox-MenuItem | If checked, sorts the list by join date in ascending order. This option takes no action, but is used in conjunction with ‘List’ option. |
| List | MenuItem | Opens a dialog which contains a list of the customers. If one of the list items is selected, the Customer details dialog is displayed. |
| Purchases | JMenu | Contains the following items… |
| Total | JMenuItem | Opens a simple message dialog showing the total of the purchases, or the total of the filtered purchases. |
| By Last Name | JCheckBox-MenuItem | If checked, sorts the list by last name in ascending order. This option takes no action, but is used in conjunction with ‘List’ option. |
| By Title | JCheckBox-MenuItem | If checked, sorts the list by title in ascending order. This option takes no action, but is used in conjunction with ‘List’ option. |
| Descending | JCheckBox-MenuItem | Sorts the list in descending order. The order is unaffected if “By Last Name” or “By Title” aren’t checked. This option takes no action, but is used in conjunction with ‘List’ option. |
| Filter by Customer ID | JMenuItem | Opens a message dialog that allows the user to enter a customer ID. This ID is used to filter the list. If an invalid customer ID is entered, a warning message is displayed. If no customer ID is entered then the filter is removed. |
| Help | JMenu | Contains the following item… |
| About | MenuItem | Displays an information dialog about the application |

The ‘About’ item must be directly accessible via an accelerator key, F1. By pressing F1, the About dialog box will be displayed.

As stated above, the Customer > List menu item will open a dialog that contains a list filled with the customers. If a row in the list is clicked then a new dialog will open with the details for that customer. This dialog will look similar to the dialog that was created in Lab 9. The user can change any of the fields except the ID and update the database if ‘OK’ is pressed. If ‘Cancel’ is pressed, the changes aren’t saved. Either ‘OK’ or ‘Cancel will close the dialog. If any changes are saved to the database and the Books application is exited, and then restarted, the user will be able to navigate to the customer and see that the values have been updated.

You don’t need to implement a details dialog for either the books or purchases lists, but you can if you want to.

From the requirements listed above, you can see that only create, read, and update functionality has been used from the DAO; you’ll still need to implement delete functionality, though this feature will only be graded based on code inspection. Although create is implemented, you don’t need to create any new records over and above what already exists in the text data files.

Adding update and delete functionality for the Books and Purchases is optional.

If any of these requirements are unclear, make sure you ask for clarification.

## Frequently Asked Questions

|  |  |
| --- | --- |
|  | 1. Do I have to implement a Runnable in MainFrame in order to create a thread so the end time can be printed when the application closes?  A. No, you can add the end time and duration calculation to the WindowClosing handler.  2. What do you mean by "the filter is removed" exactly?  A. If I enter "Honda" in the inventory report, then only the Honda parts are shown. If the filter is blank then all the items are shown.  3. If an invalid make is entered, will an inventory report be displayed to the dialog?  A. That behaviour isn't defined, you can decide to not filter by anything, filter by whatever was entered (in that case show nothing), or ask the user to enter a valid make.  4. If no make is entered, will only a warning message be displayed, or will an inventory report (with all listed items) be displayed to the dialog, or will nothing be shown?  See #2. All items are shown.  5. Do I have to make all reports be shown to dialog or it's okay to show the report in MainFrame UI as you did in the Lab 10.  A separate dialog.  6. Do I have to use a derby database?  A. No, you can use derby, or SQLite, or others, but you must include all the resources and files so that your assignment jar file runs without requiring any additional setup. |

## Submission Checklist

I have:

☐ Used a file template to add my name, student number, and file creation date and time to all source files

☐ Used packages; the root package is my student number,   
ex. package a00123456.…;

☐ Met all the functional requirements

☐ Used great object-oriented design

☐ Followed the java coding guidelines, including naming conventions and source formatting

☐ Created a runnable Jar file named Books2.jar

☐ Included all source code & required resources

☐ Zipped up all my files into a single file named <your student number>.zip,   
ex. A00123456\_assignment2.zip

☐ Submitted my lab before the due date & time

For EACH requirement not followed in the checklist, you’ll lose 1 mark. OUCH!

## Grading

The assignment will be marked out of 13