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 BCMC 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 BCMC2 must follow good object-oriented principles and practices as it did for BCMC.

Your code must compile and the jar file must run. Compile-time warnings are considered errors and must be eliminated from your code by using appropriate annotation tags.

The main class must be named Bcmc, and the jar file containing your runnable code must be named <student#>Bcmc.jar, i.e. A00123456Bcmc.jar.

As in assignment 1, all activity must be logged to a text file named bcmc.log, i.e. bcmc.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 the printing of the reports to the console and to a text report that we had in assignment 1.

Similar to your labs, the first time Bcmc is run, the data will be read from same data files used in assignment 1. The data will be stored into three separate tables in the ‘beangrinder’ database – you can create additional tables if you need them. If collections are used, they are only used to temporarily store datasets. A separate Database class is used to manage the connection to the database. 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.

The application contains a graphical user interface that will be displayed when the user runs BCMC. 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.

We’ll be focusing on the Inventory.

The follow menu must be implemented:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Notes | |
| File | JMenu | Contains the following item |
| Quit | MenuItem | Quits the application |
| Data | JMenu | Contains the following three items… |
| Customers | MenuItem | Opens a dialog which contains a simple dialog box stating this feature is not available yet |
| Service | MenuItem | Opens a dialog which contains a simple dialog box stating this feature is not available yet |
| Inventory | MenuItem | Opens a dialog which contains a list filled with the inventory data |
| Reports | JMenu | Contains the following five items…  NOTE! Only the Inventory report is displayed in this assignment |
| Total | JMenuItem | Prints the total value of the inventory in a simple dialog box |
| Descending | JCheckBox-MenuItem | If selected, will indicate the inventory will be sorted in a descending order; ascending otherwise |
| By Description | JMenuItem | Sorts the inventory report by description and displays the results in a JTextArea or JList within a dialog box |
| By Count | JMenuItem | Sorts the report by part count and displays the results in a JTextArea or JList within a dialog box (If you’re feeling brave, you can try a JTable) |
| Make | JMenuItem | Opens a JOptionPane that allows the user to enter a motorcycle make. This make is used to filter the report output. If an invalid make is entered then a warning message is displayed. If no make 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.

**Note on JtextArea or JList, they do not come with Scroll-ability.**

**Do not do : JTextArea ta = new JTextArea(): after this do pass a scroll pane to the text area, like so:**

**JScollPane = js = new JScrollPane(ta)**

**Note: on the inventory data selection initially list the ID or description. When one element is selected the dialog would show all.**

**Note: Make is only used as a filter, not actually displayed in the inventory report.**

**Note: put an accelerator on Quit, maybe one on Inventory if we want, but they should all have mnemonics.**

As stated above, the Inventory menu item will open a dialog that contains a list filled with the inventory items. If a row in the list is clicked then a new dialog will open with the details for that item. 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 GIS application is exited, and then restarted, the user will be able to navigate to the item and see that the values have been updated.

You don’t need to implement a details dialog for either the customers or service lists.

From the requirements listed above, you can see that only create, read, and update functionality has been used from the DAOs; 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.

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

## Submission Checklist

I have:

☐ Named my main class Gis

☐ Met all the functional requirements

☐ Followed the java coding guidelines

☐ Run “Source > Format” on my project

☐ Used a file template to add my name & student number to all source files

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

☐ Used great object-oriented design

☐ Created a runnable Jar file

☐ 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 10