PROG 24178
Object Oriented Programming 2
Assignment 2

This lab is based on the lessons on JavaFX GUI.

This assignment is to be done individually; you are not allowed to work on this assignment with anyone.

Description

Use Eclipse to create a new JavaFX Application project.

Project Specs

Project Name: A2_LoginName (where LoginName is your own sheridan login name).
Package Name: loginname (your own login name, in lower-case letters). Do not use any other name.

Main Application Class Name: Assign2Main

This is a JavaFX application. You will be using the Inventory class that you wrote in Assignment 1, and a new class that models a list of Inventory items.
When you create your project, make sure the main Application class is called Assign2Main.

Failure to follow the above instructions could result in penalties.

This application is used to keep track of inventory information. A company wants to allow users to input information about inventory items and then print a list of items in inventory that need to be re-ordered.

There are three classes in this assignment (specific details further down) in addition to the application main class:

- The Inventory Class
  - This is the class that you wrote in Assignment 1.
  - You must make any necessary corrections to this class before you write this assignment. See your Assignment 1 evaluation.

- The InventoryList class
  - Models an Array or List of Inventory objects that are input by the user.

- The Assign2Controller class
  - Has an object of the InventoryList class to store and manage the information of items in the inventory.

o This class must contain all event handling code needed to run the application (more details below).

All classes must be part of the same package.

1. Inventory Class
   You wrote this class in Assignment 1.
   You must make any corrections to this class before you finish Assignment 2.

2. InventoryList Class
   The InventoryList class models a list of the inventory items entered by the user. For this assignment it will be used to contain Inventory objects as they are input by the user. The specs for the InventoryList class are shown in the diagram below:

Class: InventoryList
Data Members: - invList : ArrayList<Inventory>
Method Members: + InventoryList() + add(inventory : Inventory) : void + get(index : int) : Inventory + length() : int

- The invList member is an Array or ArrayList of Inventory objects. You can instantiate the empty ArrayList in the same statement in which you declare it.
- The add() method accepts an Inventory object to add to the invList. There is nothing special going on here: simply add the object to invList.
- The get() method accepts an integer index and returns the Inventory object at that index, but only if the index is a valid index. If the index is invalid, the method should simply return a null object.
- The length() method returns the size of the invList (the # of elements that contain Inventory objects).
- You don't need a toString() at this time, but you could add one if you wanted to.

The UI Controls, Layout, and Styling

The Main UI
Your program has the following UI elements (specific details about functionality are outlined later):

- A label heading with the text "Inventory Tracker". The label is bound to the top of the screen and remains centred, regardless of screen width.
- Label/TextField pairs to be used for data entry by the user. There is one pair of controls for each field input. The labels contain the String captions.
- The ADD button. This button has the hot-key "A" and is used to clear the input fields so that the user can easily type inputs for a new Inventory object.
- The SAVE button. This button has the hot-key "S" and is used to save the current input values as an Inventory object to the InventoryList. The inputs entered by the user are validated before the Inventory object is added to the InventoryList: any errors are displayed in *Alert* dialog boxes.
- The ORDERS button. This button has the hot-key "O" and, when clicked, will display a list of Inventory objects from the InventoryList that need to be re-ordered (qoh <= rop). The list of items is displayed in the TextArea below the input area.
- The EXIT button. This button is used to exit the program, upon confirmation from the user.

- The input fields and buttons are laid out in the "Input Area", in a GridPane container (instead of AnchorPane, details later).
- The Messages Label. This label is at the bottom of the UI. It is used for:
  - It displays a message "No items to list. Add some." in red when the user clicks the ORDER button while the InventoryList is empty.
  - It displays a message "No items to re-order." when the user clicks ORDERS and there are items in the InventoryList, but none that need to be re-ordered.

The UI elements must be laid out and displayed exactly as shown in the screen shots. Your UI must contain a meaningful title in the title bar.

The UI's Functionality
Use event handlers to add control functionalities as follows:

- When ADD Button is clicked or Hot key: A is used, the following tasks are done:
  - ✓ All text fields emptied of all text.
  - ✓ Depending on how you deal with the message label, you might also need to remove the "error" from the message label at this point and empty its text.

- When SAVE Button is clicked or Hot key: S is used:
  - ✓ Construct a new Inventory object and use the mutator methods to assign the user inputs to each field.
  - ✓ Handle any error: display an Alert dialog box for the field that has the error. Note: only show one error message at a time. Showing several error dialogs at once is extremely cumbersome for the user. Alert dialogs appear with Inventory error messages when an input is invalid.
  - ✓ If no exceptions occur: Add the new Inventory item to the InventoryList.

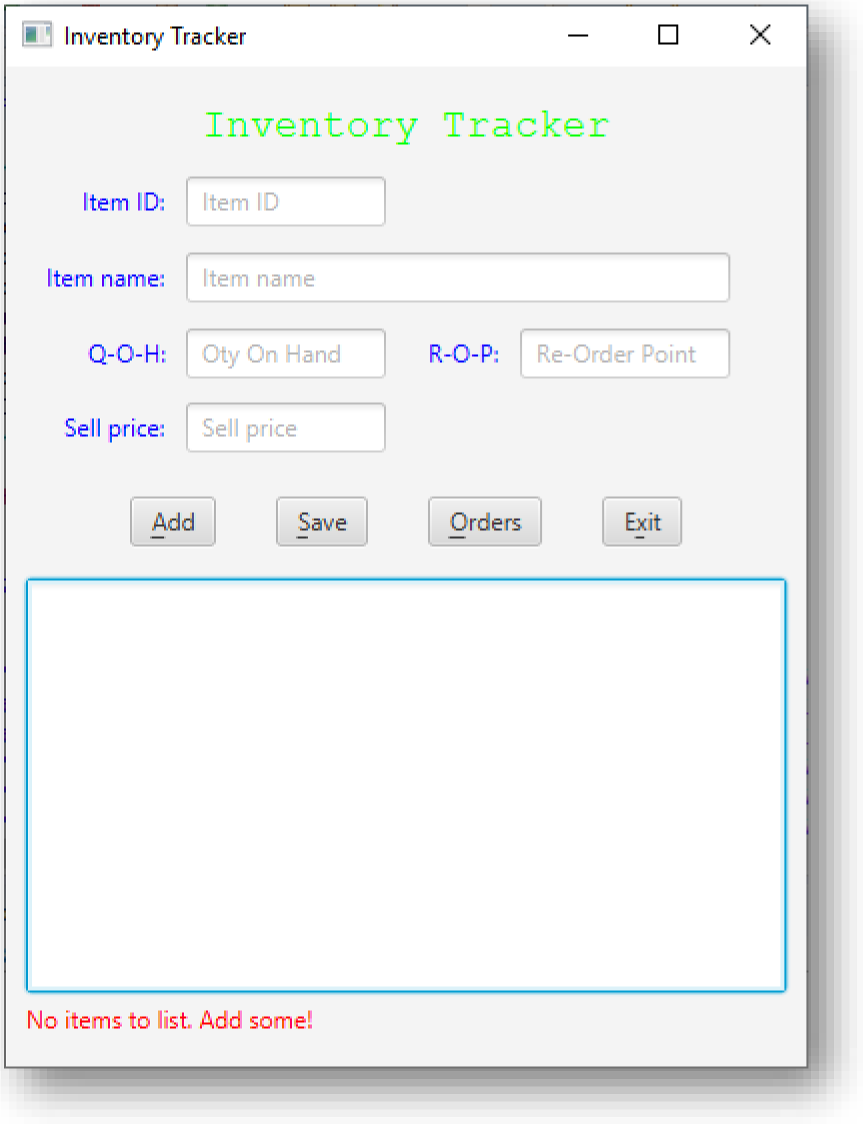
- When ORDERS Button is clicked or Hot key: O is used:

- ✓ Display only those items from the InventoryList that need re-ordering
- ✓ If the InventoryList is empty, display the message "No items to list. Add some." in the bottom message label.
- ✓ If there are items in the InventoryList, display each Inventory object that needs to be re-ordered by adding those Inventory objects to the TextArea on a separate line.
- ✓ If no items get added to the TextArea, display the message "No items to re-order." in the bottom message label.

- EXIT Button
  - ✓ Displays an Alert dialog that asks if the user is sure that they wish to exit (Yes/No). If the user says Yes, exit the program. If the user says no, do nothing.

Tips and Hints:

1. Add a GridPane instead of AnchorPane for the main GUI interface of the project.
2. Select the Gridpane and from layout tab (right side), change Padding = 10,10,10,10.
3. By right-clicking on the edge of a row or column you can add/remove one more row/column.
4. In First row, from the containers tab (left side), add an HBox container. Change the Alignment property of this HBox to "Center", then add a Label.
5. The label should now appear in the center of the first row of the GridPane. To change the font of the label, select it and change the Font property to fontname = "Courier New" and size="20.0".
6. To change the label's text color, head to properties and look for Style section, click on the field under it and search for "-fx-text-fill" (it sets the color of the text, -fx-background-color for background color etc.) then set its value on the right side to "00ff00" for green.
7. Set label text property to "Inventory Tracker".
8. In second row of the GridPane, also add an HBox container and while selected, from layout tab look for padding and spacing properties. Set padding to 10,10,10,10 and spacing to 10.
9. Add a pair of label/TextField in the HBox above (2nd row) and watch how spacing and alignment properties work automatically to position the TextField to the right side of the label.
10. Repeat 8 and 9 for each row (three more rows) of the GridPane to add a pair of label/TextField for inputting every peace of information of the Inventory item, as shown in GUI below.
11. In the sixth row, add another HBox container and then set padding to 20,20,20,20 and spacing to 30. Then add 4 buttons for Add, Save, Orders, Exit respectively where they are centered in the row.
12. To add a Hot Key for any button, for example Save button (such that the user can press Alt+S to perform a click using keyboard), precedes letter S by underscore in text property (ex., Text = "_Save") and also set MnemonicParsing = true (from property tab as usual). Do the same procedure for all other buttons.
13. For the seventh row, add a TextArea and set the property Editable = false (uncheck it from the properties tab) and set its PrefHeight to 200.
14. For the final (eighth) row, add the Message label to display feedback messages when ORDERS button is clicked as described above.
15. For Exit button, search for how to close application/window in javafx.



The GUI interface for the Project.

Submission

Follow these instructions carefully or you risk penalties up to 100% of your grade!

You are to submit two files, separately:
1. The Eclipse project that contains your assignment, in a ZIP/RAR file.
2. A text or word processed document containing all of your source code for all classes

1. ZIP/RAR of Eclipse Project:

Your project must be a valid Eclipse project. No other project types will be accepted. Your submission file must be archived into a valid ZIP or RAR file.

The name of this file should be loginName_a2.zip or loginName_a1.rar where "loginName" is your Sheridan user name.

2. Document of Source:

You must also copy and paste all of your source code for all java classes into a plain text file (e.g. .TXT) or Word document (e.g. .DOC/.DOCX).
You don't have to format this code - it's used by TurnItIn (the originality checker in SLATE, which is a piece of software that checks your submission for plagiarism against other submissions in the college, in other colleges, from the web, and various other sources).
Submit this document in addition to your source code zip/rar file. DO NOT add it inside your zip/rar file - it must be a separate file. This is used for TurnItIn (it won't accept java programs and won't examine the contents of zip/rar files).

Submit your assignment to the Assignment 2 drop box in SLATE. Upload the ZIP/RAR file, upload the TXT/DOC file separately to the same drop box.

Failure to follow any of the instructions above will result in penalties or a grade of 0.

Evaluation

Your submission will be evaluated based on the following criteria:

Efficient Code: Program doesn't use too much repetitive code (e.g. uses methods instead of copy/pasting code). Program uses variables where and only when necessary; program doesn't define variables that are never used, nor does it use too many variables for unnecessary tasks; program logic is written concisely and is not cluttered with unnecessary tasks.
Functionality: Program functions according to specifications.
Programming Style: Proper indentation and spacing; use of comments; coding conventions regarding variable/method/class names followed.
GUI design/implementation: your GUI design and implementation must follow the detailed instructions given above.
Other: All instructions regarding submissions and program specifications have been followed; submission was completed and submitted as requested in a timely fashion; techniques discussed in class have been used.

Originally adapted from Prof. Wendi Jollymore's material.