Taking Orders Fast - ELSA (Sprint 4)

Due Tuesday, April 18 at 8 a.m.

CSE 1325 - Spring 2023 - Homework #10 / Sprint 4 / Rev 1

Assignment Background

The Exceptional Laptops and Supercomputers Always (ELSA) store offers the coolest (ahem) deals in computing technology for the savvy computer geek and their lucky friends. Each computer can be hand-crafted to match the technologist's exact needs, with a growing selection of convenient predefined configurations already purchased by your discerning peers (and competitors). They now belatedly seek to automate their physical and online storefronts, replacing paper forms and ink pens with the miracle of modern computing technology. Your goal is to prove that you can implement their store management system and thus win the contract to build it, with all of the associated fame and cash.

This is Sprint 4 of 6.

IMPORTANT: Do NOT use full_credit, bonus, or extreme_bonus subdirectories for this project. Instead, organize your code into two packages, `store` and `gui`. ALWAYS build and run from the P10 directory.

The Scrum Spreadsheet

The Feature backlog HAS changed for this sprint. Please incorporate the change into your own planning.

The intent of using a *very* simplified version of Scrum on this project is to introduce you to the concept of planning your work rather than just hacking code at the last minute and hoping for the best. **Professionals make a plan and then execute it.** Amateurs sling code and suffer the consequences.

Submitting an *accurate* spreadsheet is part of your grade for these assignments. This is not what you *wish* had happened - it's what actually happened. **Be certain you update the spreadsheet and add, commit, and push it at cse1325/P10/Scrum.xlsx before the end of the sprint!** If you prefer to use an online spreadsheet, ensure that it is publicly visible and include a cse1325/P10/Scrum.url text file instead containing its *public* link.

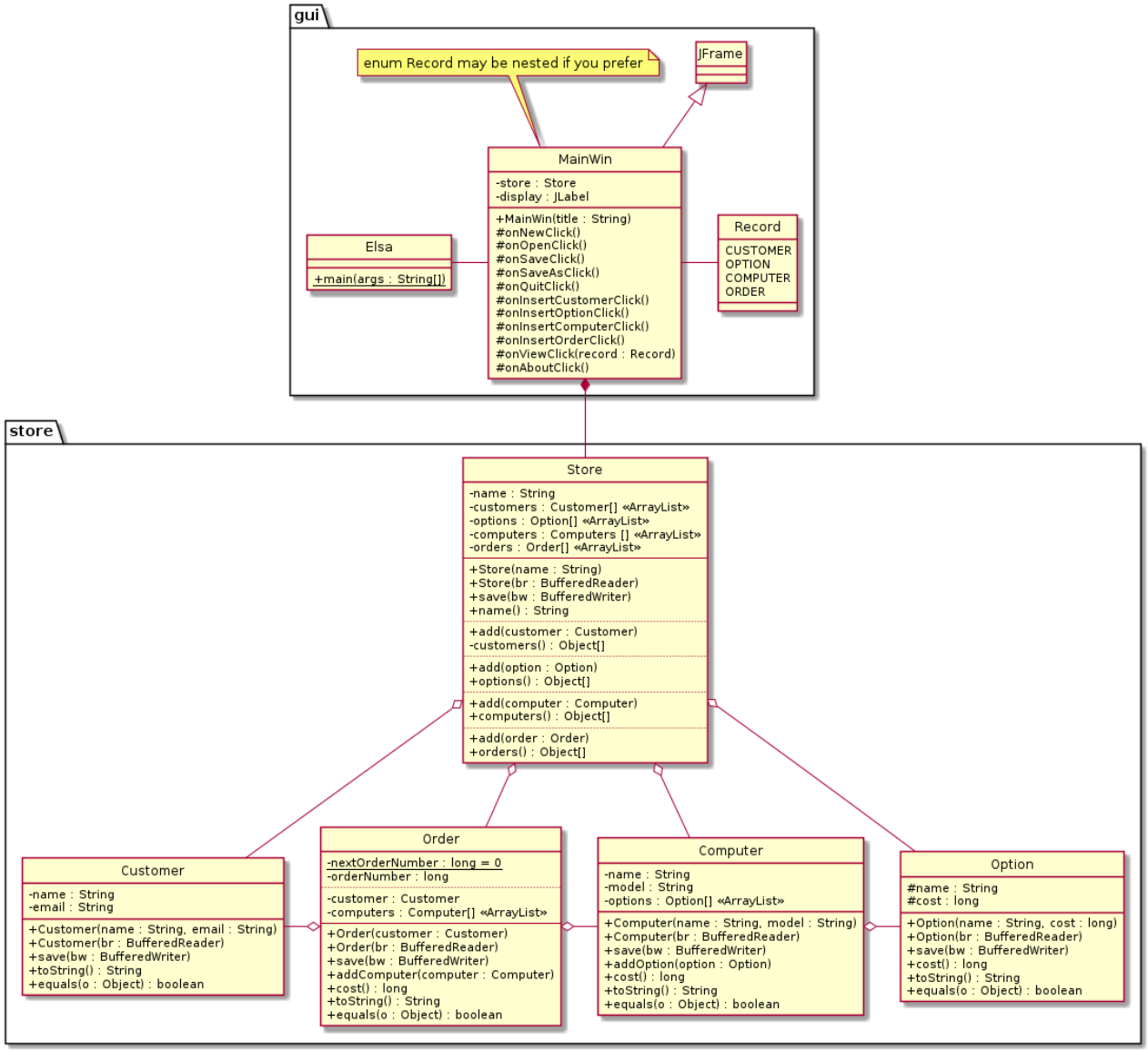
For this sprint, at a minimum, you will update the Product Backlog AND Sprint 04 Backlog tabs. Then add, commit, and push your Scrum spreadsheet.

Now that you *have a plan* - on to coding!

Class Diagram

The diagram on the next page covers just Sprint 3. A more complete diagram will be provided once you have a basic understanding of Graphical User Interfaces (GUI) and Java Swing (our GUI toolkit). And updates to fix bugs and design issues are all but inevitable. (Dotted lines are purely visual and have no semantic meaning.)

For this sprint, we will move our Store-related code into package `store`, then focus on a basic implementation of package `gui` based on the Nim code from Lecture 12. Details instructions follow the class diagram - read and follow them *closely* to avoid wasting time and writing incorrect code!



Write the Code

Updating store.Order

(OCOST) Add the `cost()` method to class Order, which returns the sum of the cost of the elements of the `computers` field. Also include the `cost()` in `Order.toString()`.

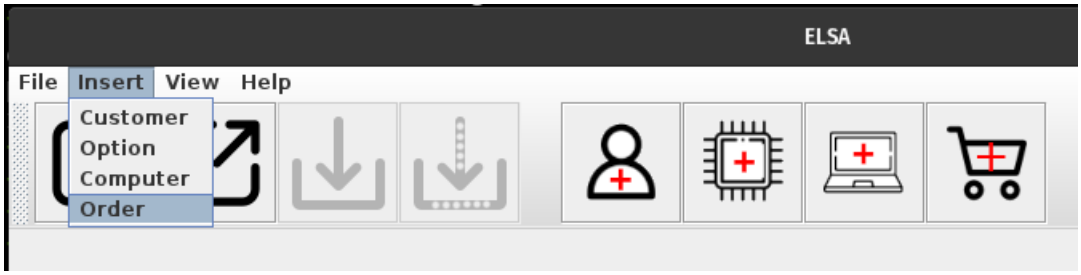
Updating gui.MainWin

View > Orders

(DISP2) Add Orders to the View menu along with a button on the toolbar. Both should add an ActionListener to display all Order objects in the Store in the main window.

Insert > Order

(INSOR) Add Order to the Insert menu along with a button on the toolbar. Both should add an ActionListener to call the `onInsertOrderClick()` method when selected.



`onInsertOrderClick()` should first ask the user for a Customer for whom the Order will be taken (this is a required parameter for the `Order` constructor).

- If no Customer objects exist, call `onInsertCustomerClick()` to give the user a chance to create one. If no Customer is created, return.
- If one Customer object exists, use it `without asking the user`.
- If two or more Customer objects exist, display a dialog with the JLabel ""Order for which Customer?" and a JComboBox listing the available customers, including an OK and a Cancel button. If OK is clicked, use the Customer selected in the JComboBox, otherwise return.

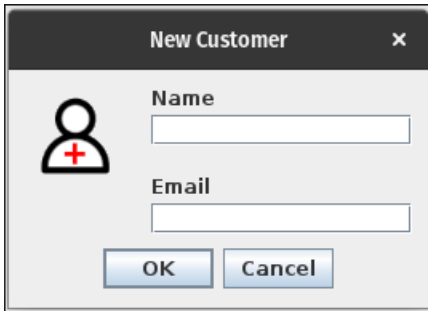
Most people expect applications to be smart enough not to ask questions for which only one answer is possible.

If you successfully obtain a Customer:

- Instance an Order. It will not yet include any computers.
- Loop, providing a dialog *showing the current order* with a JComboBox listing available Computer products. If the user selects one, add it to the Order. If they ask to place the Order, and at least one Computer has been added, then add the Order to the store.
- If a new Order was placed, set the dirty bit (if your program has one) and change the main window to display orders.

Unified Dialogs

(DCUST) Replace the sequence of two dialogs for creating a new Customer - one each for the name and the email address - with a single dialog in which both can be entered. *Show the icon you used on the toolbar button* for Insert > Dialog as the icon in this dialog. Include an OK (or similar name) and Cancel buttons, and create a new Customer and add to the store if OK is selected, otherwise the dialog should simply close.



Do the same for obtaining the name and cost of a new Option.

Do the same for obtaining the name and model of a new Computer. You will need additional dialogs for selecting each option.

Hint

Copy and pasting code in 3 different methods of your program should set off a flashing red warning light in your brain. If you need to do something 3 times, you should probably wrap it in a single method, right?

So you *may* find it easier to write a UnifiedDialog method. (Yes, minor deviations from the class diagram are fine.) For example, you might write

```
String[] UnifiedDialog(String[] fields, String title, String iconFilename)
```

where `fields` are the text to be displayed in the JLabels that should accompany the JTextBox objects, `title` is the dialog title (third JOptionPane.showConfirmDialog parameter), and `iconFilename` is the path for loading the (required for full credit) icon that matches the button on the toolbar.

Here's an outline for such a method, if you'd like to write one (it's optional, but better).

- Load the icon from `iconFilename` if possible. If not, use `null`.
- Create an `Object[]` that contains alternating JLabel objects (from `fields`) and new JTextBox objects. The size will thus be twice that of `fields`.
- Run JOptionPane.showConfirmDialog.
- If OK was clicked, load the text from the alternating objects (that is, the JTextField objects) into a `String[]` and return it.

With this, you can obtain any number of String inputs in an array from the user with a single method call.

That's how the suggestion solution handles it, but you are free to follow a better approach if you have one.

About Dialog

Update the About dialog as needed to credit the two additional toolbar button icons needed for this sprint. Updating the program version would be a good idea as well.

File I/O

Ensure that your program can still save and open its own files. Backward compatibility is not required, but would be nice to maintain.

Screenshot

IMPORTANT: Your application should NOT look exactly like this. **Make ELSA your own!** Be creative with your icons, your About dialog, and other visual effects.