Assignment 6:

(total 30 points)

Exercise 1: Implementing a GUI for the Gourmet Coffee System

(30 points)

Prerequisites, Goals, and Outcomes

Prerequisites: Before you begin this exercise, you need mastery of the following:

- *Graphical user interface*
  - Knowledge of Swing components and containers
  - Knowledge of Swing event handling
  - Knowledge of `JFileChooser` dialog

Goals: Reinforce your ability to create a GUI with event handling

Outcomes: You will master the following skills:

- Produce interactive applications that use a Swing GUI

Background

In this assignment, you will create a comprehensive GUI for the *Gourmet Coffee System*. Part of the work has been done for you and is provided in the student archive. You will implement the code that handles the button events.

Description

Class `GourmetCoffeeGUI` implements a GUI for the *Gourmet Coffee System*. This GUI lets the user display the product catalog, process orders, and track the store's sales:

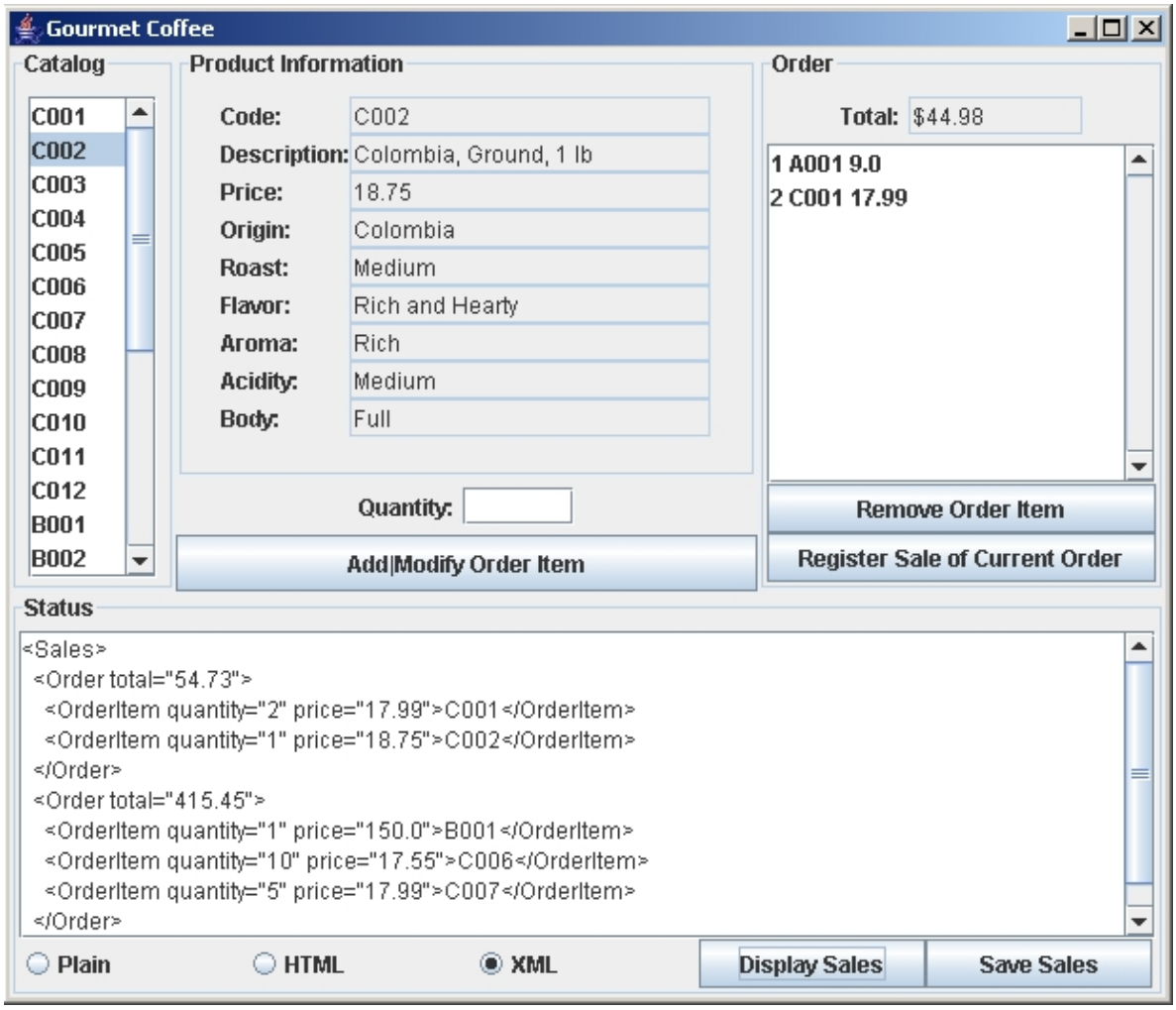

Figure 1 *Execution of GourmetCoffeeGUI*

The GUI is divided into four panels: Catalog, Product Information, Order, and Status:

- The Catalog panel contains a list of product codes.
- The Product Information panel contains product details. Positioned immediately beneath this panel are a text field to specify the quantity of the selected product and a button to add the selected product to the current order.
- The Order panel contains the current order, its cost, and buttons to edit the order and register its sale.
- The Status panel contains a text area to display messages and sales information; command buttons to display and save sales information; and radio buttons to specify the format of the sales information when it is displayed or saved.

When a user clicks one of the following command buttons, `GourmetCoffeeGUI` handles the associated event as described below:

- *`addModifyButton`*. Adds the selected product to the current order. If the selected product is already part of the order, modifies the quantity of that product in the order.
- *`removeButton`*. Removes the selected item from the current order.
- *`registerSaleButton`*. Adds the current order to the store's sales and empties the current order.
- *`displaySalesButton`*. Lists all orders that have been sold in the specified format (plain text, HTML, or XML).
- *`saveSalesButton`*. Saves all orders that have been sold in a file with the specified format (plain text, HTML, or XML).

When a user clicks one of the following radio buttons, `GourmetCoffeeGUI` handles the associated event as described below:

- `plainRadioButton`. Changes the format of the sales information to plain text.
- `HTMLRadioButton`. Changes the format of the sales information to HTML.
- `XMLRadioButton`. Changes the format of the sales information to XML.

When a user selects an element in the following list, `GourmetCoffeeGUI` handles the associated event as described below:

- `catalogList`. Displays the details of the selected product in the "Product Information" panel.

`GourmetCoffeeGUI` defines the following Swing components:

- `catalogList`. Displays the product code of every product in the product catalog.
- `orderList`. Displays the items in the current order.
- `quantityTextField`. Allows user to specify the quantity of the selected product.
- `totalTextField`. Displays the total cost of the current order.
- `statusTextArea`. Displays status messages and sales information.
- `fileChooser`. A `JFileChooser` object. Allows user to specify the name and location of the file in which the sales information will be saved. This dialog box appears when the "Save Sales" button is clicked.
  `GourmetCoffeeGUI` defines the following instance variables:
- `catalog`. Contains the product catalog, a `Catalog` object.
- `currentOrder`. An Order object that contains the items to be purchased.
- `sales`. A Sales object that contains the orders that have been sold.
- `salesFormatter`. A `SalesFormatter` object that specifies the format to be used when the sales information is displayed or saved.
- `dollarFormatter`. A `NumberFormat` object used to find the dollar representation of a number. (The dollar representation of 1.0 is "$1.00".)
  New methods have been added to classes Catalog and Order:
- The method `getCodes` has been added to the class Catalog. It returns an array of product codes (all the product codes in the product catalog) which is used by `GourmetCoffeeGUI` to populate the catalog `JList`.
- The method `getItems` has been added to the class Order. It returns an array of the `OrderItem` objects (all the items in the current order) which is used by `GourmetCoffeeGUI` to populate the order `JList`.
  The button events are handled using named inner classes in class `GourmetCoffeeGUI`. The following inner classes are complete and should not be modified:
- `DisplayProductListener`. Listener for the catalog list
- `RegisterSaleListener`. Listener for the button "Register Sale of Current Order"
- `PlainListener`. Listener for the radio button "Plain"
- `HTMLListener`. Listener for the radio button "HTML"
- `XMLListener`. Listener for the radio button "XML"
- `DisplaySalesListener`. Listener for button "Display Sales"
  In this assignment, you should complete the implementation of the following inner classes:
- `AddModifyListener`. Listener for button "Add|Modify Order Item"
- `RemoveListener`. Listener for button "Remove Order Item"
- `SaveSalesListener`. Listener for button "Save Sales"

Files

The following files are needed to complete this assignment:

- *exe-gourmet-coffee-gui.jar*. Download this file. It is the sample executable.

- *student-files.zip*. Download this file. This archive contains the following:
  - Class files
    - *Coffee.class*
    - *CoffeeBrewer.class*
    - *Product.class*
    - *Catalog.class*. A modified version of class Catalog
    - *OrderItem.class*
    - *Order.class*. A modified version of class Order
    - *Sales.class*
    - *CatalogLoader.class*
    - *File__CatalogLoader.class*
    - *DataFormatException.class*
    - *SalesFormatter.class*
    - *PlainTextSalesFormatter.class*
    - *HTMLSalesFormatter.class*
    - *XMLSalesFormatter.class*
    - *DataField.class*
  - Documentation
    - *Coffee.html*
    - *CoffeeBrewer.html*
    - *Product.html*
    - *Catalog.html*. A modified version of class Catalog
    - *OrderItem.html*
    - *Order.html*. A modified version of class Order
    - *Sales.html*
    - *CatalogLoader.html*
    - *File__CatalogLoader.html*
    - *DataFormatException.html*
    - *SalesFormatter.html*
    - *PlainTextSalesFormatter.html*
    - *HTMLSalesFormatter.html*
    - *XMLSalesFormatter.html*
    - *DataField.html*
  - Java files
    - *GourmetCoffeeGUI.java*. Use this template to complete your implementation.
  - Data files for the test driver
    - *catalog.dat*. A file with product information for every product in the product catalog

Tasks

Complete the implementation of `AddModifyListener`, `RemoveListener`, and `SaveSalesListener`. The messages displayed in the status area by these inner classes should match the messages displayed by the sample executable. Follow Sun's code conventions. The following steps will guide you through this assignment. Work incrementally and test each increment. Save often.

1. **Extract** the files from *student-files.zip*
2. **Run** the sample executable by issuing the following command at the command prompt:

```
C:\>java -jar exe-gourmet-coffee-gui.jar
```

Note : This application requires the file *catalog.dat* which you will find in *student-files.zip*.

3. **Then** , copy the code of method `getDataFieldsPanel` that you created in the previous exercise.
4. **(10 points)Next**, implement method `actionPerformed` in the inner class `AddModifyListener`: If the current order does not contain an item with the selected product, then create a new order item and add it to the current order. Otherwise, locate the item in the order with the selected product and changes its quantity to the value specified by the user. Finally, update the current order list (use method `Order.getItems` to obtain an array containing all the items in this order), display a status message in the status area, and update the display of the order's total cost. Use the following code to display the order's total cost in dollars:

```
totalTextField.setText(dollarFormatter.format(currentOrder.getTotalCost()));
```

This method should display an error message in the status area when it detects one of the following errors:

- The quantity text field does not contain an integer
- The quantity text field contains a negative integer or zero.
- The user has not selected a product.

5. **(10 points)Then**, implement method `actionPerformed` in the inner class `RemoveListener`: Remove the selected item from the current order, display a status message in the status area, update the current order list (use method `Order.getItems` to obtain an array containing all the items in this order), and update the display of the order's total cost. Use the following code to display the order's total cost in dollars:

   ```
   totalTextField.setText(dollarFormatter.format(currentOrder.getTotalCost()));
   ```

   This method should display an error message in the status area when it detects one of the following errors:

- The current order is empty.
- The user has not selected an item in the order.

6. **(10 points)Next**, implement method `actionPerformed` in the inner class `SaveSalesListener`: Save the sales information in a file. Begin by opening a file chooser so the user can specify the name and location of the file in which the sales information will be saved. Next, save the sales information in the specified file. The file should be saved in the format indicated by the radio button selection. Finally, display a status message in the status area.

   This method should display an error message in the status area when it detects one of the following errors:

- No orders have been sold.
- The user closes the file chooser without selecting a file.
- The file specified by the user cannot be created or opened.

8. **Finally** , compile and execute the class `GourmetCoffeeGUI`.

Submission

Upon completion, submit **only** the following.

1. GourmetCoffeeGUI.java, GourmetCoffeeGUI.class
2. a word file including the program running result.