CM2100 Advanced Software Design and Development

Coursework – Part 1

Sandwich Shop Application – GUI Application

Handout Date:	03 December 2018
Submission Deadline Date:	14 January 2019 by 23:55pm
Feedback will be returned by:	28 January 2019
Submission Instructions:	You must submit your complete Netbeans project solution as a single zipped file the **CM2100 GUI Coursework** submission Dropbox in the Assessments section on the CM2100 Moodle page. Within the Netbeans project should include a **Statement of Compliance** see notes on page 2.

Coursework Instructions

This part of the Coursework component of this module contributes 20% towards your final CM2100 grade.

In this coursework you will create a Graphical User Interface that adds interactive functionality to an existing Java project. You are asked to adapt the project by adding and coding a JFrame class and JDialog classes that work with the existing hierarchy of interacting classes.

The project starting point is available to download from the Assessment section at the top of the CM2100 Moodle page. See the notes below on **Setting up your Netbeans Project**

You have considerable freedom in how to implement your solution. A set of expected functionality (in the form of 16 Functional requirements on pages 5-7) is given in this coursework specification, but you can implement this functionality using any appropriate GUI components as you see fit.

The context of the project, and relationship between the starting point classes is described on page 4.

Setting up your Netbeans project

- Download the **`cm2100guicw.zip`** file from the Assessment section of the CM2100 Moodle page.

- Unzip the cm2100guicw.zip file. You may rename the Netbeans project if you wish.

- Open the project in Netbeans.

- There should be seven Java files in the project: Bread.java, Drink.java, Sandwich.java, Order.java, MenuItem.java, Menu.java and Side.java

- The starting point folder also contains a sample StatementOfCompliance.txt file and a folder with some sample jpg image files for use in your solution.

- You should add further Java classes that extend the JFrame and JDialog classes as appropriate to fulfil the coursework requirements.

Testing and Statement of Compliance

There are no specific marks for testing your code, but obviously you should test all functionality that you implement since correct functionality is the primary assessment criterion.

For convenience the Menu.java class has a method that can be used to populate a Menu object with some data – to avoid having to enter lots of data every time the system is run.

Within your Netbeans project include a StatementOfCompliance.txt file. (An empty one is provided in your starting point netbeans project). In this, list each of the 16 requirements and clearly indicate whether there are complete or not, and whether you have tested them.

e.g. Fully implemented, working and tested.
Partially implemented, working but with some errors, and tested.
Partially implemented, not tested.
Not attempted.

Code Comments and Code Layout

You should include an appropriately edited comment in the following format at the top of each Java class that you edit or create:

```
/* Classname.java, edited/created by Your Name
 *Coursework Part 1 – due 14 December 2019 */
```

If you make any changes to the existing starting point classes then you must provide code comments explaining the nature and purpose of the changes, and also include a comment in the Statement of Compliance file indicating in which files any changes have been made.

Additional code comments are not required, and there are no additional marks for code comments in this exercise.

You may comment-out any code that has syntax errors. Partial marks may be given for code that is partially correct and commented-out.

Neat coding style will be one of the assessed aspects.

- Use code indentation to ensure code legibility.
- Use appropriate spacing between methods and sections of code to ensure good readability.

Deliverables

You are required to submit a zip file copy of your *completed* Netbeans project, submitted via the **GUI Coursework Code Submission Dropbox** on CampusMoodle.

Once you have completed the coursework:

- **SAVE YOUR NETBEANS PROJECT**
- Close Netbeans
- Locate the Netbeans project folder containing your work
- Create a *.zip version of your entire Netbeans Project Folder
- Go to the **GUI Coursework Code Submission Dropbox** on Moodle
- Submit your zip file to the assignment dropbox

Marking

You will be assessed on your attempt at the coding exercise, in terms of correct functionality, but also coding efficiency, and also from a design viewpoint in terms of appropriate choice of GUI components and the layout and utility of the GUI frames and dialogs.

An indicative grading grid is given below:

GRADE	A	B	C	D	E	F
DEFINITION	EXCELLENT Outstanding Performance	COMMENDABLE Meritorious Performance	GOOD Highly Competent Performance	SATISFACTORY Competent Performance	BORDERLINE FAIL Open To Compensation	FAIL Unsatisfactory
CODING AND FUNCTIONALITY	Coding fully meets specification except for perhaps minor errors. No syntax errors Reasonable attempt made at all the more complicated functions. Code is presented in a professional manner in terms of indentation and legibility Statement of compliance is complete and accurate	Coding meets most of specification except for perhaps minor errors, or some missing features. An attempt made at the more complicated functions. Any syntax errors are commented out. Code is presented in a near professional manner in terms of indentation and legibility Statement of compliance is complete and accurate	Coding meets majority of specification except for perhaps minor errors, may have some missing or incomplete features. Any syntax errors are commented out. Code is well presented in terms of indentation and legibility Statement of compliance is mostly complete and accurate	Coding has meaningful attempt at substantial proportion of the specification. May have some missing or incomplete features or several syntax errors. Few syntax errors. Code is adequately presented in terms of indentation and legibility Statement of compliance is mostly complete and accurate apart from minor oversights	Design and implementation attempts to meet specification requirements but has omissions or deficiencies that make the software unusable. Few syntax errors. Code is poorly presented in terms of indentation and legibility Statement of compliance is either incomplete and/or inaccurate	Coding fails to meet specification requirements. Minimal code and/or many or significant syntax errors Code is very poorly presented in terms of indentation and legibility Statement of compliance is missing or highly inaccurate.
DESIGN AND CHOICE OF GUI COMPONENTS	Design of the interface suitable, user friendly and shows efficient/rational use of components.	There are some slight deficiencies in the design of the interface	The interface design meets the specification but the choice of components is poor	Design of the interface meets the specification but there are several deficiencies.	Design of the interface only meets some of the specification.	Design of the interface is poor.

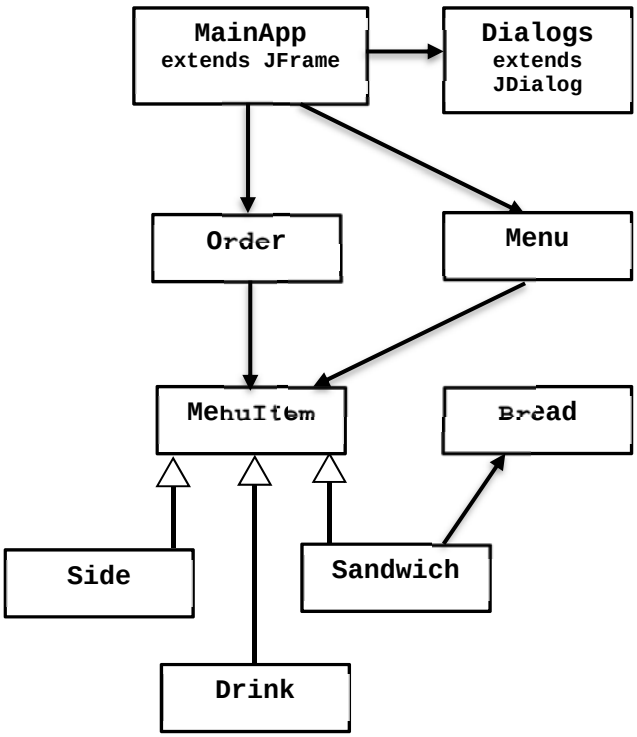
Hints

The following hints might be useful.

- Read and understand the description of each Requirement before you start coding it.
- Test your code as you implement it – don't allow errors to accumulate.
- Remember to use the appropriate conventions for naming variables and classes.
- Any renaming of variables or classes should be done using Refactor.

Coding Exercise Context – A Sandwich Shop GUI Application

The aim of this coding exercise is to create a graphical user interface that uses the set of interacting classes constructed as part of the practical exam for storing the application data.



The system should allow users to manage and organise Orders of items from a Sandwich Shop. The basic design of the class relationships in the project is shown in the diagram:

Seven of these classes are available and complete in the Netbeans project **`cm2100guicw.zip`** download from the Assessment area on the CM2100 **Moodle** page.

As part of the assessment you will use these classes to store and manipulate the data for your GUI application

In this version of the Netbeans project:

- ***MainApp*** should extend JFrame, and run as the main starting point of the application.
- ***Dialogs*** should extend JDialog. This is not necessarily a single file, but represents any separate dialogs that you create as part of achieving the specified functionality and which are started from the MainApp of the application.

MainApp and any separate Dialogs are the files you should add and work on. The other classes described below are provided as a starting point and should not need to be edited.

- ***Menu*** is a class containing a list of ***MenuItem*** objects representing the food and drink items for sale at a Sandwich Shop. This was not part of the Part 1 of the Practical Exam, so you should study and investigate the functionality provided in this class.
- ***Order*** is a class containing a collection of ***MenuItem*** objects associated with a customer
- ***MenuItem*** is an abstract class encapsulating common features of different food and drink that can be ordered from the Sandwich Shop
- ***Sandwich*** is a concrete class extending ***MenuItem*** and which encapsulates properties and methods associated with sandwiches sold by the shop
- ***Bread*** is a an enum type storing the fixed constant values associated with the different varieties of bread from which sandwiches can be made
- ***Side*** is a concrete class extending ***MenuItem*** and which encapsulates properties and methods associated with sides (such as salad or chips) sold by the shop
- ***Drink*** is a concrete class extending ***MenuItem*** and which encapsulates properties and methods associated with drinks sold by the shop

Within these classes, all values representing costs and prices are stored as **int** variables, and represent costs and prices in pence.

Functional Requirements for the GUI Sandwich Shop Application

Application Behaviour

Requirement 1 Main Window Contents

- The program should run as a graphical user interface (GUI).
- The main window should have a Menu Bar and a number of menus that will depend on how you organise the other requirements.
- On the main window there should be:
  - A button with which a new customer order can be started
  - A means of selecting items from a menu of food and drink items
  - A display of the customer order details
- Other features can be placed on menus, or as other components on the main window.

Requirement 2 Closing the System

- The user should be able to close the system via an **Exit** option on one of the **Menus**
- The user should also be able to exit the system by clicking on the **close** button of the main window.
- In either case the windows should close and the program should shut down without crashing.

Application Data In the `MainApp` class

Requirement 3 Food and Drink Menu

- The program should be able to store and manage a Menu of food and drink items that are for sale at the shop.
- This can be done as a single Menu.java object or as separate Menu objects for Sandwiches, Sides and Drinks if you prefer. (The Menu.java class is available in the starting point files).
- In this and all the following requirements **Food and Drink Menu** refers to these items that customers can select from at the shop and not a GUI menu component. So on the GUI these don't need to be displayed on JMenu components. Instead, ideally the user should be able to choose items from from GUI lists such as a JComboBox or JList.

Requirement 4 Customer Order Details

- The program should be able to store and manage one current Order object for food and drink terms being purchased by the current customer.

Menu Admin Management

Requirement 5 **Adding new Sandwich items to the Food and Drink Menu**

- The program should allow the user to specify and add new sandwiches to the main menu of items offered by the shop.
- Sandwiches require a name, a cost, a description of the filling and a default bread type.
- A dialog (or some other means) should allow the user to specify the properties and add the new Sandwich type to the appropriate food and drink Menu list.

Requirement 6 **Adding new Side items to the Food and Drink Menu**

- The program should allow the user to specify and add new side dishes to the main menu.
- Sides require a name, a cost and a default sauce.
- A dialog (or some other means) should allow the user to specify the properties and add the new Side type to the main menu.

Requirement 7 **Adding new Drink items to the Food and Drink Menu**

- The program should allow the user to specify and add new drinks to the main menu.
- Drinks require a name, a cost and must be classified as whether containing alcohol or not.
- A dialog (or some other means) should allow the user to specify the properties and add the new Drink type to the main menu.

Requirement 8 **Saving the Food and Drink Menu to a text File**

- The program should allow the user to save the details of the food and drink items available at the shop to a text file.
- The **Menu** class has a method that can be used to write to a file.
- Your task is to use that menu in conjunction with e.g. a JFileChooser to allow the user to select the text file to which the data is to be saved.

Requirement 9 **Loading the Food and Drink Menu from a text File**

- The program should allow the user to load the details of the food and drink items available at the shop to a text file.
- The **Menu** class has a method that can be used to read from a file.
- Your task is to use that menu in conjunction with e.g. a JFileChooser to allow the user to select the text file to which the data is to be saved.

Customer Order Management

Requirement 10 Starting a new customer order

- The program should allow the user to initiate a new Customer order.
- Upon selecting a new order option (e.g. via a JButton, or a JMenuItem) the user should be requested to provide the name of the new customer.
- The system should then create a new Order object.

Requirement 11 Displaying customer order details

- The program should display the details of the current customer order, including the customer name, the order ID, the total cost of the order, and a list of the items ordered by the customer.
- These should be updated automatically when a new order is started, or whenever items are added to the order.

Requirement 12 Adding Items to the Customer Order

- The program should allow the user to select items from the Food and Drink menu, and to add the selected item to the order of the customer.
- If a drink that contains alcohol is selected, then the system should display a dialog asking the user to confirm that the customer is 18 years or older [a simple Yes or No is sufficient], and should only add the item to the Order if they are 18 or over.

Advanced System Features

Requirement 13 Images of the Menu Items

- The program should allow the user to select an image file for each new food or drink item and associate it with the MenuItem.
- Upon selecting the item on the main window, the image should be displayed on a JLabel before the item is added to the customer.

Requirement 14 Sortable Food and Drink Menu(s)

- The program should allow the user to decide if the items displayed on the food and drink selection menu are ordered by item name, or by price
- The menu from which items are selected should be updated automatically when the sorting is requested by the user.

Requirement 15 Changing Bread Type for a Sandwich in the Customer Order

- The program should allow the user to select a Sandwich from the Customer Order, and select and change the type of bread used for the sandwich.

Requirement 16 Add/Change sauce for a Side in the Customer Order

- The program should allow the user to select a Side from the Customer Order, and select and change the type of sauce used on the side dish.

END OF EXERCISE – see page 3 for Instructions on submitting your Netbeans project