GROUP NAME G3 [Section 800] Diet Manager V1 – Rubric

SWEN.383 SW Design Principles and Patterns

Final Java Solution for Diet Manager V1

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Tasks** | **Points** | **Score** |
| 1 | Load a foods.csv file with only basic foods and an empty log.csv file. Users can see an empty log for today with default calories and weight. Users can view the basic foods loaded.  TODO: Explain what you did here and where to find this feature in your code.  Csv files are loaded using the load methods found in the FoodHandler and LogHandler classes. Loading is called from the DMModel class, which serves as the model of the project.  A log file is created and loaded inside the LogHandler. Inside the load method (lines 16 to 83, the class is located inside the io package) we first instantiate a Scanner, a String representing the file path to our lg.csv, and finally an array of Strings which will hold the attributes read from a log.csv file, if such exists. Inside of the try block (lines 21 tol 75) is where we use an if statement to check whether log.csv already exists and stores some data or whether it needs to be created anew (lines 25 to 29). The Log object is manipulated by the controller in order for it to be passed from the model to the view (LogListener inside the controller package, line 35) Here, we call getLogOnDate method, to which we pass in the date from the view which uses DatePicker in order to display the log for the exact date passed in and filling in other necessary details inside the actionPerformed method (lines 21 to 60).  A list of basic foods from the food.csv is displayed in the ComboBox, which is populated with foods once the csv file has been read and loaded in the app’s model. The code for loading all food is found in the DMView class and it takes a list of all the food names that have been read from the csv file by the handler inside the io package that is set to deal with inputs and outputs within the program. The method in the view is called updateComboBoxes (lines 390 to 397). It is called from the DMController (which is inside the controller package, line 41). | (60) |  |
| 2 | Level 1 PLUS Select a basic food(s) for the daily intake. The selection is stored in the database (log.csv) the log view is updated with the dietary information about the nutrients consumed.  TODO: Explain what you did here and where to find this feature in your code.  After selecting all the needed information and pressing the log button, the TextArea is updated with the new log and log.csv is updated to have the newest data. Additionally, dietary information is updated upon the same action. This all happens in the LogListener class (in the “io” package), which is responsible for manipulating the logs once an action has been performed (suitably, inside the actionPerformed method on lines 21 to 60). TextArea is updated by calling the addLogFood method, which is located in the DMView and called inside the LogListener’s actionPerformed method on line 58. Logging is done by calling the saveLogs method (LogListener line 60), which is located in the LogHandler class (lines 131 to 160). All of the methods called to update the dietary information are located in the DMView class. | +10 (70) |  |
| 3 | Level 2 PLUS Add new basic food(s) to the food database. This implies the ability to then add such basic food(s) to today's log as in level 2  TODO: Explain what you did here and where to find this feature in your code.  Once the user has given all the necessary information for adding a new food entry, on a button press, a new BasicFood is created and added to the storage file which is displayed through a comboBox featured in the UI. This action is the responsibility of the SaveNewFoodListener class located inside the controller package. Inside this class, we created a method called actionPerformed, in which all the necessary methods are called in order for the action to run smoothly (lines 20 to 53). New BasicFood is created using the addNewFood method, which is located in the DMModel class (lines 41 to 45). Parameters of the method take in all of the inserted information. ComboBox is updated with the updateComboBoxes method inside the DMView (lines 390 to 397), which has the newest data. New BasicFood is saved by calling the saveFoodsFile method from the FoodHandler class (lines 86 to 121 inside FoodHandler, method is called from the aforementioned action listener’s class actionPerformed method). | +5 (75) |  |
| 4 | Level 3 PLUS Loading and viewing a foods.csv file with recipes.  TODO: Explain what you did here and where to find this feature in your code.  FoodHandler class has a load method that performs the loading for both basic food and recipes (lines 16 to 83). First, we check if a file with the given name exists in order to know if we need our program to fill it with data or create from scratch for the user to be able to fill it with only new data. Then, we use if statements to differentiate basic food instances from recipes by checking the “key” character in the string that has been read, which is “b” for basic food or “r” for recipe. Depending on which letter it is, the objects are filled with the data accordingly (lines 37 to 72). Finally, all items are added to the foods collection. The loaded recipes can also be viewed and selected in the combobox. | +5 (80) |  |
| 5 | Level 4 PLUS Select recipe(s) as well as basic food(s) for the daily intake. The selection is stored in the database (log.csv) and the log view is updated with the dietary information about the nutrients consumed.  TODO: Explain what you did here and where to find this feature in your code.  Once the user has selected all the necessary information, on button press, the TextArea is updated with the newly defined log, and log.csv is updated to have the latest data. Also, dietary information is updated. This all happens in the LogListener class, which is responsible for creating new logs. For this, the actionPerfomed method that the class contains calls all the necessary methods from the application’s subsystems (lines 21 to 60).The TextArea is updated by calling the addLogFood method, which is located in the DMView (lines 496-498). Logging is done by calling the saveLogs method, which is located in the LogHandler class (lines 131 to 160). All of the methods called to update the dietary information are located in the DMView class. | +5 (85) |  |
| 6 | Level 5 PLUS add new recipe(s) to the food database. This implies the ability to  then add such recipe(s) to today's log as in level 5.  TODO: Explain what you did here and where to find this feature in your code.  The selected ingredients are added by pressing the save ingredient button. After each press, the IngredientListener is called (located inside the controller package), which updates the log with all the ingredients (actionPerformed method on lines 15 to 31). After pressing the save button, a new Recipe is created, and all ingredients are added. Moreover, the recipe is also added to the comboBox.  All of this occurs in the SaveRecipeListener class (located inside the controller package). The Recipe object is created under the name provided by the user. The Log with the ingredients is set to have all the ingredients. Later that data is used to add ingredients to the recipe. The addIngredient method is called on the new recipe object (inside Recipe class, inside the model package, lines 49 to 52), which takes ingredient name and quantity (method is called inside the SaveRecipeListener, line 45). After that is done, a new recipe is added to the list of all foods and the combo boxes are updated. Lastly, the recipe is saved by calling the saveFoodsFile method from the FoodHandler class (lines 86 to 122). | +5 (90) |  |
| 7 | Level 6 PLUS the ability to read a non-empty log.csv file, to navigate to  different days in the log, and to select foods for the intake for the days other than today.  TODO: Explain what you did here and where to find this feature in your code.  Users can click on the DatePicker and select a different date. All of the data is then updated to display the data set for that specific date entry. DateListener class (inside the controller package) is used to reflect these changes. | +5 (95) |  |
| 8 | Level 7 PLUS the ability to save the log and food database back to the log.csv and foods.csv files.  TODO: Explain what you did here and where to find this feature in your code.  Logging is done by calling the saveLogs method, which is located in the LogHandler class (lines 131 to 160). Food is saved by calling the saveFoodsFile method from the FoodHandler class (lines 86 to 123). These methods are called inside the action listeners which are triggered by the user’s actions which reflect their responsibility (e.g. the user wants to save a recipe entry, the SaveRecipeListener.java is triggered). | +5 (100) |  |
| SUBTOTAL: | | **100** |  |

To receive any credit for level N, the preceding levels must be sufficiently functional to test level N. In general, this means previous levels must work without failure when the user enters normal (non-error) data.

Remember, the solution should apply design patterns as required. The final grade will be adjusted based on how good you have applied the pattern. You may lose up to 25% of the final project grade in this regard.