GROUP NAME G1 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. User can see an empty log for today with default calories and weight. User can view the basic foods loaded.  TODO: Explain what you did here and where to find this feature in your code.  So everything is handled in the controller. We have a loadData method in the controller that calls the method read that both models(foods and logs) contain. The read method in the foods, takes each line and is passed as a parameter in the createFood method of the factory. Then based on the line factory creates either BasicFood or Recipe and returns it. Then the Load data just appends to the textArea the food.ToString with all the information about the basicfood or recipe. | (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.  The class named HandleAddToLogs implements the EventHandler interface to handle user actions, particularly adding food items to logs. It reads the selected item from the view, distinguishes between regular food items and recipes, extracts necessary information, creates a log entry, and writes it to a CSV file using the csvModel.write() method. The method getFormattedDate() is used to retrieve the current date in a specified format. This feature can be found in the controller package, particularly in the HandleAddToLogs.java 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.  The process of adding a food to foods.csv is handled in the HandleAddFood class. The handle addFood takes the Data from the textfields that are in the view and creates a new basic food. Then we call the method .write() from foods model that takes the Food as a parameter and writes it to the food.csv file. Our handle addFood also adds this information to the text area whenever a new food is added. | +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.  Everything is handled in the controller. We have a loadData method in the controller that calls the method read that both models(foods and logs) contain. The read method in the foods, takes each line and is passed as a parameter in the createFood method of the factory. Then based on the line factory creates either BasicFood or Recipe and returns it. The difference is that when there is an instance of recipe we are also adding food objects (we check if the foodnames from what we read from the csv file exists in the foods arraylist) and then we call .addFood method that adds the basicFood to the array along with the count.  Then the Load data just appends to the textArea the food.ToString with all the information about the basicfood or recipe. | +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.  This is very similar to level 2, the code we have for selecting a food for daily intake just distinguishes whether the selected food is a basic food or a recipe, and based on that a log object is created which is written down to the log.csv using the csvModel.write() method, and afterwards the Foods ListView and the Logs ListView is cleared and updated with latest information about foods and logs. This can be found in HandleAddLogs.java class in the handle method. | +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 process of adding a new recipe to the database aswell as having it immediately display on the screen is handled in the HandleAddRecipe class. The information that is needed to create a Recipe is received from the textfields and comboBox located in the view. Then we loop through the arraylist of foods that we get by calling the read method again and we check if compare the foodnames with the ones we got from reading the file. If they are equal, we add the food object to the Recipe. | +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.  We made it so the user can navigate through different days in the log using a DatePicker which is created in the View class, the logic for showing the logs for that certain date is in the Controller.java class specifically the handleDateSelection() method which just takes the selected date from the DatePicker field in the View class and then reads through the log.csv file to find a matched date which we parse in a format that is same as the datepicker format and then we update the Log ListView afterwards with the selected date and the data available for the selected date, a user can pick whichever date they want and if there is a match for that date in the log.csv it will show the data. | +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.  Whenever we add a basicfood, recipe or log to the food.csv file or log.csv file, the changes are displayed both in the screen aswell as in the database.  when adding one of the 2 files. We call the .convertTOCSV() method that returns a string applicable with the format of the csv file. Whenever we want to add to the screen we call the method .ToString that returns the applicable format that is user friendly to be displayed in the screen. | +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.