Health Check

Group Members:

Lawrence A. Buljanovic

Alexa Doda

Pieter Rudovic

Mentor and professor:

Domagoj Tolic The Wise

Phase 2

Explanation

Let's start with the MVC pattern for the classes :

The HealthCheck class extends the HealthCheckView class - Serves to start the app.

HealthCheckController class extends the HealthCheckView - Controller class keeps tabs on both the model and view classes.

HealthCheckModel class extends the HealthCheckView - The Model class is in charge of managing and manipulating all of the backend data.

HealthCheckView class extends the application and implements the event handler - The view class is in charge of everything GUI-related.

Recipe class extends implements CompositeRecipe - The recipe class is a leaf of the composite class, it takes data from the file and displays it.

ComponentRecive class - Abstract component class which defines the basic behavior.

CompositeRecipe class - composite class holds the food recipes.

We created the factory pattern where it is responsible about the types of the food. We didn’t link all together, because lack of time. But, we linked them in the diagrams. The gui runs and the user can interact with it.

Rationale :

We wished to create the project as simple as possible and to create that aesthetic we have decided to do the whole project in an MVC pattern in order to have a clear separation of concerns. Using the model, view, and control pattern we have made it so that the model holds the responsibility of archiving the data and specifying in which way the data is handled and used. The view part of the MVC pattern in our project is responsible for all the GUI talks related. The controller is tasked with connecting the model and view, this helps decrease the coupling and increase the cohesion in order to make the app easier to maintain. This also ensures that the application is more suitable for multiple people to work on the application. Besides the MVC pattern, we also use the composite pattern which is implemented around the food items and recipes where each recipe is a composite of other recipes and foods. The composite pattern is used in order to make calculations on the entire collection easier to be done.

For the additional two patterns we did the facade and factory pattern. The facade pattern is there just so the user can move around the app easier. The rest of the code is unawre of the facade pattern and therefore work induvidually without the information about the facade.

The final patter implemented was the factory pattern which is used in order to create recipes and food.

The shortcomings of the presentation are :

1) We think that the model and view patterns could have been done better in order to prevent the coupling and enhance the cohesive perspective of the code.

2) There is a lack in the running of certain functions because they have been created successfully but however they have not been implemented correctly in the project due to top the lack of time and therefore do not run as planned.

3) The GUI could have been done better in order to catch the user's eye with more work on its esthetics but however as previously mentioned due to the lack of time we have been unable to pour in more effort into the design and aesthetics of the GUI.

4) we apologize for the misunderstandings with the diagrams and code but this is to be expected as we are a group and have the work divided among the three of us. Each of us has named certain functions differently before sending them to the group leader in order to connect all of the code into this magnificent project.

Skeleton Implementation

The skeleton implementation is available on the repository Aleksa&Lawrence on Github.