SFWRENG 2XB3: Software Engineering Practice and Experience: Binding Theory to Practice

System Design Specification Document

ezHealth

Rehan Theiveehathasan - 001416031
Edwin Do - 400079331
Timothy Chen - 400129081
Brian Kibazohi - 400077789
Ridhwan Chowdhury - 400075426

14 April 2019

TABLE OF CONTENTS

Class Breakdown

Note to Reader:

Please see the generated java documentation found in the javadoc folder for an in depth breakdown of classes, and functions, and section 2 of the Requirements Specifications Document to see class decomposition tables. All descriptions of semantics, functions, variables are found in the previously mentioned references.

The classes have been decomposed such that there are five classes, Read.java, Food.java, DirectedEdge.java, Bag.java, and EdgeWeightedDigraph.java. Please see section 2, and 4.3 of the Requirements Specifications Document to see the requirements trace to class interface.

The reasoning is to only have critical files necessary for the program to run efficiently and simply to allow for maintainability,

With this class structure we can then a main class that handles all data processing of the database, a class that constructs objects to contains the data from the database, and lastly any data structure classes necessary to help process the data into valuable input.

As such the UML class diagram of ezHealth is as follows:

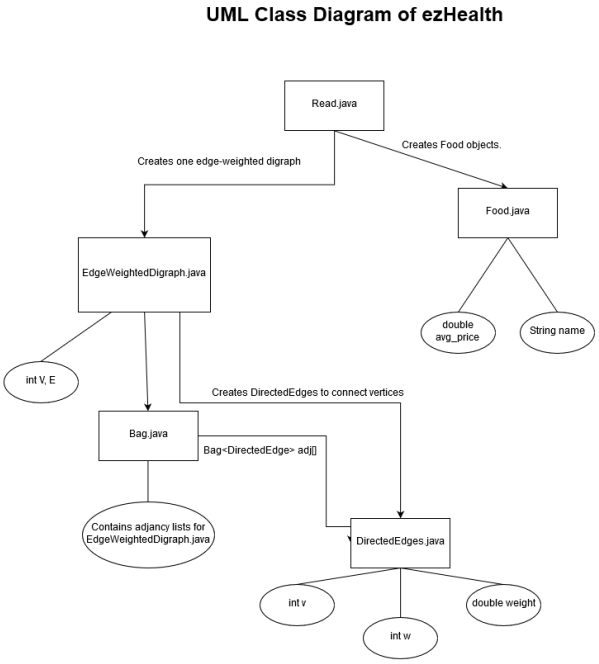

Figure 1: UML Class Diagram of ezHealth

1.1 Read.java

Read.java is the class that holds the main function of the project. The class serves the following purposes:

First - guide the user through using the application with console print statements.
Second - Parse the wholesale food database into useful Food objects.
Third - Sort Food objects by price parameter from least to greatest.
Fourth - Output most varied grocery list of wholesale food items based on user's budget constraint.

1.2 EdgeWeightedDigraph.java

EdgeWeightedDigraph.java is the class that construct the graph data structure that is necessary in holding the connected and weighted DirectedEdge objects that show the connections between Food objects. This class also holds the important function of simpleLongestPath() that determines the longest path possible between Food objects given a user budget.

1.3 Food.java

Class that constructs Food objects which contain data related to the price of a food and the name of the food.

1.4 DirectedEdge.java

Class that constructs DirectedEdge objects user to link food objects via direction and weights. Contains data related to: vertex v -> vertex w, and the cost of this traversal (weight).

1.5 Bag.java

Class that constructs Bag objects that hold data related to the adjacency matrix of the entire EdgeWeightedDigraph object. This helper class exists as a data structure purely for DirectedEdge.java and EdgeWeightedDigraph.java.

State Diagrams

The two most interesting classes that exist in the ezHealth project are: Read.java and EdgeWeightedDigraph.java.

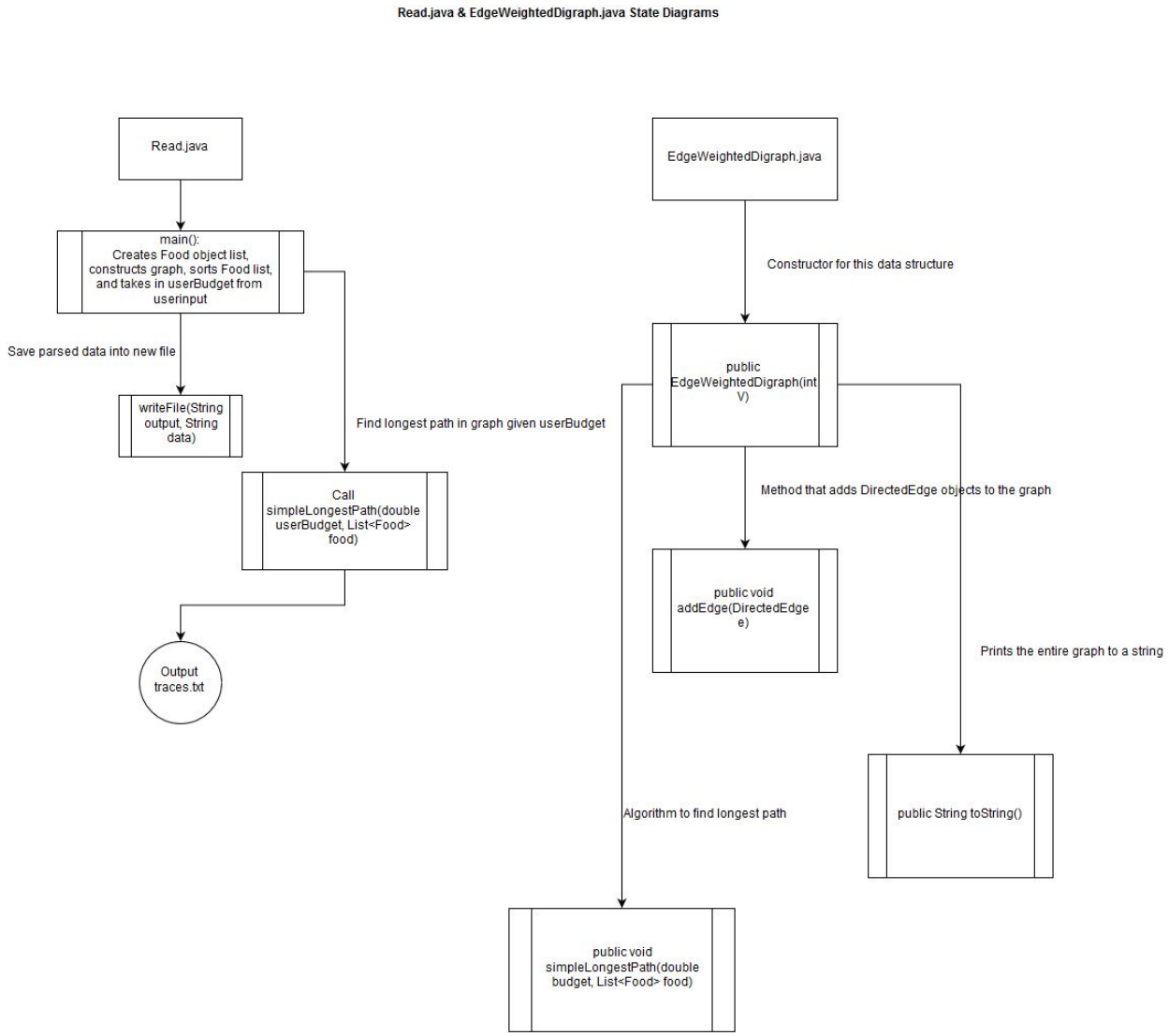

Figure 2: State-Process Diagrams showing control flow of Read.java and EdgeWeightedDigraph.java.

Internal Evaluation

In terms of features, the project could have had more features such as a feature to ignore food items the user knows they dislike and will not purchase. Another useful feature that the user may find useful is information on where to purchase the foods on the generated grocery list. However this is difficult to do without having access to stock information directly provided by brick and mortar stores.

In terms of algorithms, the project could have had faster though more complicated algorithms to parse CSV databases, which would become more useful if the database was orders of magnitudes larger than it currently is. Lastly, a complexLongestPaths algorithm could be created to give more interesting traces of potential longest paths, however this directly conflicts with our Non Functional requirements of the application running relatively quickly. This is because such an algorithm has a non polynomial run time.