Palate Tailor

Technical Report

Created by:

Emma Lurie and Dorothy Sun

1. ADTs

a) Hashtable

The **Hashtable** is created in the **createHashtable**() method in PalateTailor.java and is utilized in the **createMenu**() method in the DiningHall.java class. The Hashtable holds data from a tab separated value file for a specific dining hall, and the data is comprised of popular dining hall menu options and their student-reviewed score which is on a scale from one to ten. The Hashtable **key** is a menu option's **String** name, and the **value** is a **Dish** object representing that menu item.

If the resulting Hashtable contains a menu item listed on that day's menu, then the Dish object value from that key is retrieved from the Hashtable and added to that **DiningHall** object's daily menu. We chose to use a Hashtable because the retrieval time of a key in the Hashtable is O(1), which is efficient for searching.

b) Priority Queue

We had difficulty implementing the Java implementation of a **Priority Queue**, so we elected to use the Java Foundations implementation. Therefore, our Priority Queue uses a Max Heap, which turned out to be ideal for our program.

In our program there are two Priority Queues. The first Priority Queue contains a DiningHall object's menu. This Priority Queue is filled in the **createMenu**() method in **DiningHall.java**. We chose to use a Priority Queue primarily because we wanted an easy way to retrieve the highest rated element as we do in the **getTop()** method in the **DiningHall.java** class. Also, we wanted a structure that gave us flexibility in how many elements were added to it. The second Priority Queue is comprised of DiningHall objects in the **PalateTailor.java** class, and it is used in the **getTopTwoDiningHalls**() method. The first two items that are dequeued from the Priority Queue are the top two highest rated DiningHall objects for that day. The Priority Queue makes finding the max of the DiningHall objects or

menu items extremely easy and efficient.

b) Linked List

We used a Linked List to hold all of the String names of the menu options on the Wellesley Fresh menu in the **readWellesleyFresh**() method in the **PalateTailor**.**java** class. We chose a Linked List because of its flexible size, as well as the ease of adding and removing elements.

2. Important classes (Back-end classes)

a) Dish.java (implements Comparable<Dish>)

i. Each Dish object represents a menu item or "dish" on the Wellesley Fresh menu. The Dish class implements the Comparable interface. Every Dish object contains a name and a score. A Dish is greater than another Dish if it has a higher score.

ii. The Dish class contains getters and setters, a **compareTo**() method, a **toString**() method, and a method to **shortenName**() method that returns the first forty-five characters of the name.

b) DiningHall.java (implements Comparable<DiningHall>)

i. Each DiningHall object represents a dining hall at Wellesley College. All DiningHall objects contain a name for the object, a menu represented by **PriorityQueue<Dish>** for a given meal, and the average score, stored in a double, of the items on the menu for that given meal. The **createMenu**() method takes the Hashtable created in **PalateTailor.java** and the **LinkedList<String>** of menu options available that day produced in **readWellesleyFresh**() and creates a menu for that DiningHall object. The

calcScore() method initializes the average score instance variable once the menu has been created. This **averageScore** variable is used to compare DiningHall objects to each other. Both the **calcMenu**() and **calcScore**() methods are run in a method called **initializeDiningHall**(). These two methods cannot be completed in the constructor because when the DiningHall is constructed the user still has not decided which meal they would like to see recommendations for. So, one of the parameters **initializeDiningHall**() is the meal name.

ii. Important Methods:

1) private void addToMenu(Dish d)
2) private PriorityQueue<Dish> copyMenu()
3) public void createMenu(String todaysMenuFile, String dataFile, String mealName)
4) public double calcScore()
5) public Dish getTop()
6) public int compareTo (DiningHall d)
7) public void initializeDiningHall(String todaysMenuFile, String dataFile, String mealName)

c) PalateTailor.java

i. Each time the program is run, a **PalateTailor** object is created representing the process of choosing the best dining hall at a given meal. In the constructor, five dining halls are created as representations the five dining halls at Wellesley. This class contains a **readWellesleyFresh**() method that reads a weekly menu from a text file (**.txt**) as well as a **createHashtable**() method

that reads data from a tab separated value file (**.tsv**) that holds common Wellesley Fresh menu options and their student reviews. The **initializeAll**() method performs the **intializeDiningHall**() method for all of the five dining halls at Wellesley represented by DiningHall objects. This class also uses **Java Calendar** to return an integer representing the days of the week at a given date. This class also creates helper methods that neaten text and help retrieve the day of the week.

ii. Important Methods:

1) public void initializeAll(String mealName)
2) public static LinkedList<String> readWellesleyFresh (String inFileName, String meal)
3) public static Hashtable<String,Dish> createHashtable(String inFileName)
4) public DiningHall[] getTopTwoDiningHalls()

3. GUI classes

a) SelectorGUI.java

The driver class that establishes JFrame with three tabbed panels: InstructionsPanel, TodaysChoice, JustForYouPanel.

b) InstructionPanel.java

The InstructionsPanel is the first panel that users view. It contains the general instructions of the program and lovely pictures of all the dining halls.

c) TodaysChoice.java

The TodaysChoice panel contains both a lunch panel and dinner panel for users to click on the “See My Lunch” or “See My Dinner” buttons to see the corresponding results generated through the program. The results are calculated based on comparing today's menu

to the user scores in the Hashtable. The program will generate two best dining halls for each meal as well as menu option to try at the first-choice dining hall.

d) JustForYouPanel.java

The JustForYouPanel contains both a lunch panel and dinner panel. Each panel contains checkboxes of the top-rated dish from each dining hall so that users may check the ones they enjoy eating. The average score of the dining hall increases by one point if a meal option at that location is checked. The program generates the two best dining halls for each meal as well as a menu option to try at the first-choice dining hall.

4. Survey

In an attempt to get more accurate data, we decided to ask CS230 students for their dining hall dish preferences. The results were incorporated in tab separated value files (**.tsv**) in the data directories.

5. Testing Methods

The **Java Calendar** tool was extremely useful in developing our program. The **getDow()** returns the day of the week, which made it simple to read the weekly menu text files starting with the current day of the week and stopping reading when the next line was the next day of the week. However, this simple tool makes it very difficult to test Monday's output on Thursday. So, we created overloaded helper methods that take a new parameter "day," which is an integer one through seven. We have overloaded methods at the bottom of each file it is needed for. The overloaded methods are: **createMenu()**, **initializeDiningHall()**, **initializeAll()**,**getStartDow()**, **getEndDow()**, and **readWellesleyFresh()**. All the testing is documented in **PalateTailorTesting.java**.