

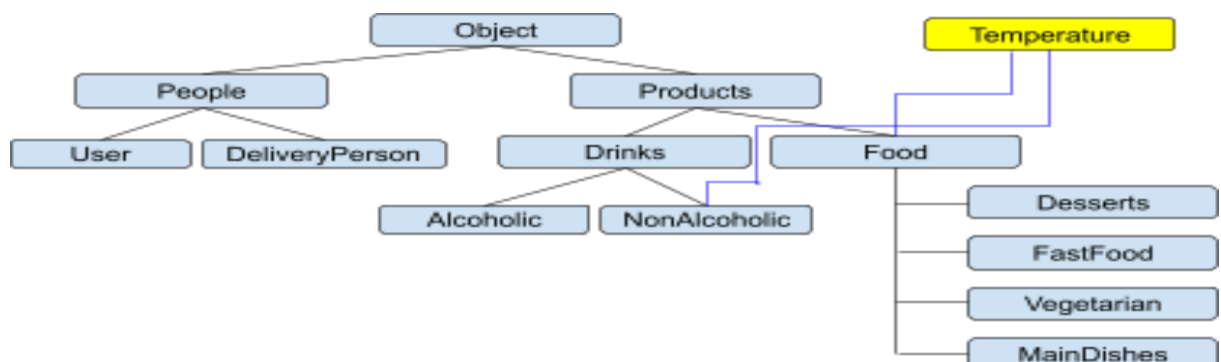
Project Idea Brief Overview:

Application, which allows a user to register/sign in, make an order from a restaurant, get delivery related information(delivery fee and time, courier name and ratings), and rate the courier once they finish ordering.

Sub Tasks:

- [Log In](#) Files GUI
 - [LogInPane](#)
 - [LogInActivity](#)
 - [User \(Class\)](#)
- [Register](#) Files GUI
 - [RegisterPane](#)
 - [RegisterActivity](#)
 - [MapPane](#)
- [Calculate Delivery Time](#) Algorithms
 - [GenerateGraph](#)
 - [FindTime](#)
- [Assign Delivery Person / Update Their Rating](#) Files
 - [GenerateDeiveryPeople](#)
 - [DeliveryPerson](#)
- [Generate Menu](#)
 - [GenerateMenu](#)
- [Display Main Menu](#) GUI
 - [GeneralMenuPane](#)
 - [Temperature \(Interface\)](#)
- [Display \(Food\) Menu](#) GUI
 - [MenuPane](#)
- [Check Out](#) GUI
 - [OrderPane](#)

Class Hierarchy:



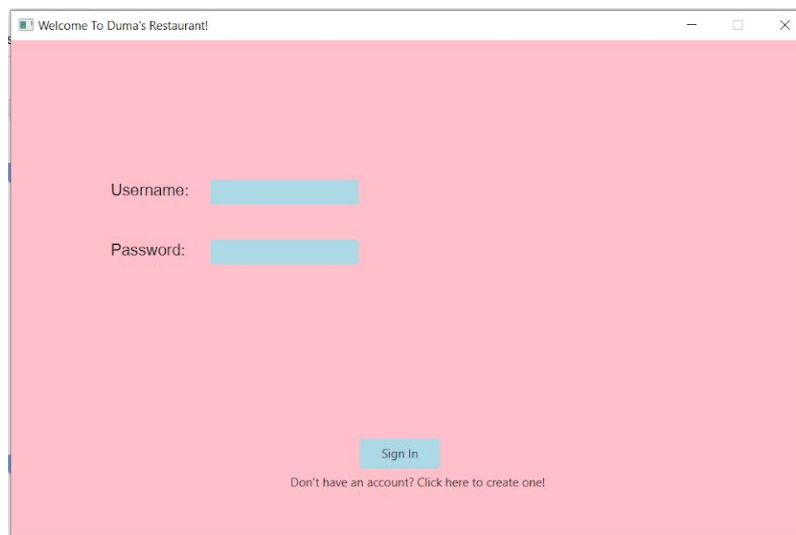
Project Description:

I used JavaFX to create a GUI for my application. In my project, I have Main.java class that is responsible for changing scenes(different panes, depending on the user's actions) for my primary stage and calling for different methods depended on the user's actions.

Log In

I created an instance of LogInPane in Main.java, and when the application is launched it this logInPane is set as a root of the scene for my primary stage.

In logInPane I have two fields text and password, where the user can type their username and password; I also have a clickable label, which is used in case the user doesn't already have an account(once it's pressed, it changes the scene root to registerPane - an instance of RegisterPane); And of course, sign-in button, which checks if sign-in was successful and brings the user to the main menu of the application.



To check if the sign-in was successful I use an instance of LogInActivity class and call boolean method `logInSuc(String username, String password)` which I have in LogInActivity. This method takes two string arguments, searches matching username and password in my data.txt file, where I keep the user's username, password, and address. I use modulus while reading data from my text file since I know that each user's information takes 3 lines, and the order of this information.

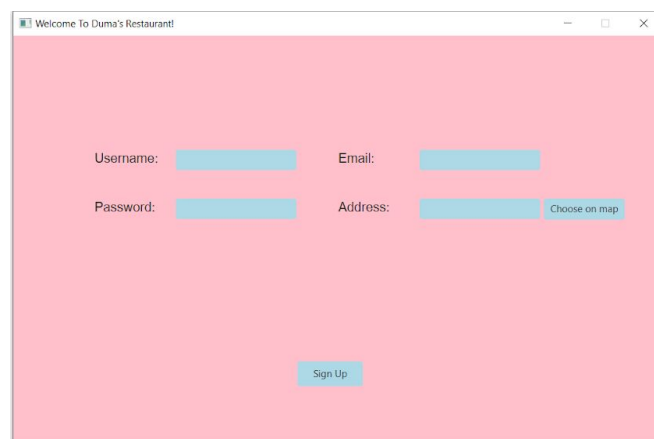
This method returns true if there is a user with a given username and password, and returns false otherwise. If the returned value is false, I create an alert dialog and let the user know that the entered username and password don't match. I also have a `getAddress()` method in `LogInActivity`. If the sign-in is successful I get the address of the user and instantiate a new object (user) of the `User` class. This makes it easier to get the user's data later in the app.

Since I also save information about couriers, and the user and courier have a common attribute - name - I decided to create an abstract class `People`. `User` class is a subclass of `People` class. In the user class constructor, I use the keyword 'super' to invoke the superclass's constructor.

Register

I have a `RegisterPane` class which extends the `Pane` class. I use text fields to let the user enter their information. I have two buttons one that displays the map and another one (sign up button) that takes the user to the main menu.

To check if the sign-up was successful I use an instance of `RegisterActivity` class and call the boolean method `signUp(String username, String password, String address)` which I have in `RegisterActivity`. This method takes three string arguments and writes them in my `data.txt` text file, but before that, I check if there is a user with the same username. For that, I have a boolean method `checkUsername(String username)`, I go through my file and see if any username (strings that are written on 3n lines, $n \geq 0$) is equal to the given one. If there is already someone in my database with the given username I create an alert and display it to the user, otherwise, I append my text file with the user's data.

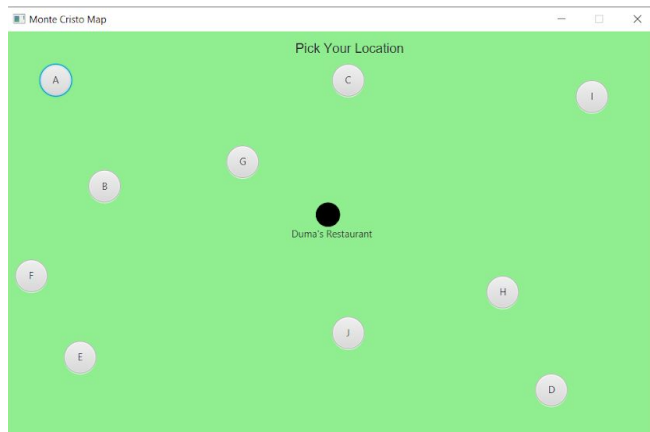


The screenshot shows a Java Swing window titled "Welcome To Duma's Restaurant!". The window has a pink background and contains the following elements:

- Username:
- Email:
- Password:
- Address:
- A button labeled "Choose on map" next to the Address field.
- A button labeled "Sign Up" at the bottom center.

MapPane/Location/Address

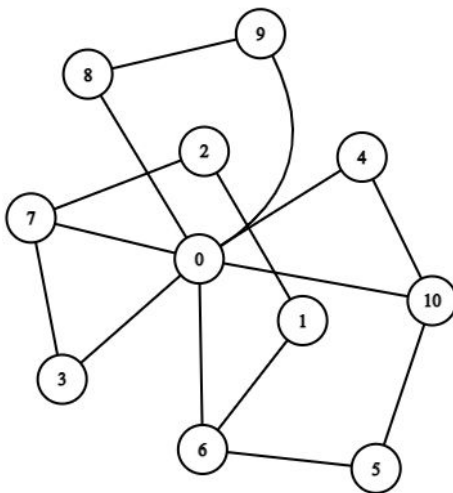
Since I wanted to implement Dijkstra's algorithm, I decided to turn this city into a graph, where delivery addresses are vertices. The address can be any character between A-J (I also check if the user's input is valid for this).



All the vertices are clickable on the map. For this, I used `ToggleButton`-s and grouped them in a `ToggleGroup`. This way, instead of creating event handlers for every button, now I added only one listener to the toggle group. After a button is clicked, I get the character of it and set it as a text in the address field.

Calculate Delivery Time

I had different ideas about implementing this part (creating a graph and setting values to the edges), so I finally chose to assign random values to the edge prices, but make the edges constant. Meaning that the graph will always look like this, but will have different edge values each time the app is launched:



Software used to draw this graph:

https://csacademy.com/app/graph_editor/.

For generating the graph I decided to have a special class for it (`GenerateGraph`). In this class, I generate the graph (2d array) in a constructor and I have a get method to get the graph (this 2d array). In my array, indices are the numbers associated with letters from A-J. Also, if there is no edge between two vertices for example i and j, then $g[i][j] = \text{INF}$ (big number).

`FindTime` is a class, where I use Dijkstra's algorithm. I have a static method in this class `static int getTime(int[][] g, int destination)`, which takes two

arguments: the 2d array of my graph and the integer equivalent for the address(A=1, B=1, .. J=10).

I have an array called mintime. Mintime[i] stores the minimum value it takes to get to i from the starting vertex, which is 0. I use a while loop to find the vertices that are connected to the current vertex and add them to the list to visit later. For each vertex, I visit I mark it as visited to don't discuss it again. For each vertex i, I check if the mintime[i] is greater than mintime[currentvertex] + g[currentvertex][i]. If yes, then I change the value of mintime[i]. This function returns mintime[destination].

Assign Delivery Person / Update Their Rating

I decided to have 7 different couriers, I save their name, ratings, and how many people rated them in a text file - deliveryPeople.txt.

I have GenerateDeliveryPeople class with four methods:

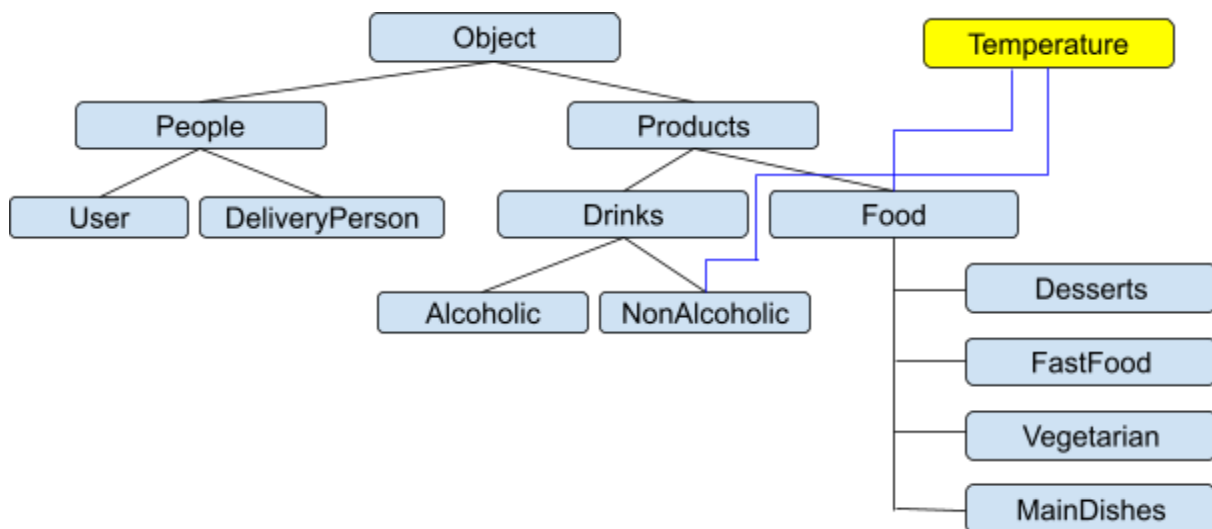
- `readData(BufferedReader aFile)` - which reads data from deliveryPeople.txt and returns an array list of delivery people. In this method, I also instantiate new objects of the DeliveryPerson class as I read.
- `getDeliveryPerson` - generates a random number n in the range of 0-6(inclusive) and returns the n-th member of the delivery people array list.
- `update(DeliveryPerson d, int r)` - this method takes two arguments. First is a DeliveryPerson object d (whose ratings must be updated) and the second argument is an integer, the ratings on a scale of 1-5 that the user gave to d. This method reads data from my text file, saves this data in a StringBuffer (info), and when it finds d in my data, it changes the ratings and the number of people that rated this courier. After the reading is done, it calls the writeData method.
- `writeData(String info)` - this method writes my StringBuffer info.toString() in the deliveryPeople text file.

DeliveryPerson is a subclass of People. I again use the keyword 'super' to invoke the superclass's constructor. For each delivery person, I have three attributes username, rating, ratingCnt.

Generate Menu

I have a class called GenerateMenu with a static method called generateMenu(). This method returns an array list of products.

I generate a random amount of products with random names, price, calories, ingredients(I have a list of ingredients and I choose random ingredients for different products), etc. The attributes depend on the type of product I am generating. To explain better I'll use my hierarchy tree.



So I have 6 different types of product: Alcoholic drinks, nonalcoholic drinks, desserts, fast food, vegetarian, and main dishes. Some of these products have many attributes in common, so I decided to create abstract classes Drinks and Food and group those 6 classes as subclasses of these two classes. Some examples:

Products class has 4 attributes that every product have: name, calories, price, ratings.

Drinks class extends Products class, and also has its attribute `int mls` - which saves the size of the drink in milliliters.

Alcoholic class extends Drinks class, and also has its own attribute `double Alper` - which stores the percentage of the alcohol in a drink.

For the Food class, I store ingredients. For the main dishes the cuisine, desserts - the level of sweetness, vegetarian - type of vegetarian.

I use the for loops to generate different kinds of products and I save this product in ArrayList <Product> product. I return this array list once generating is done.

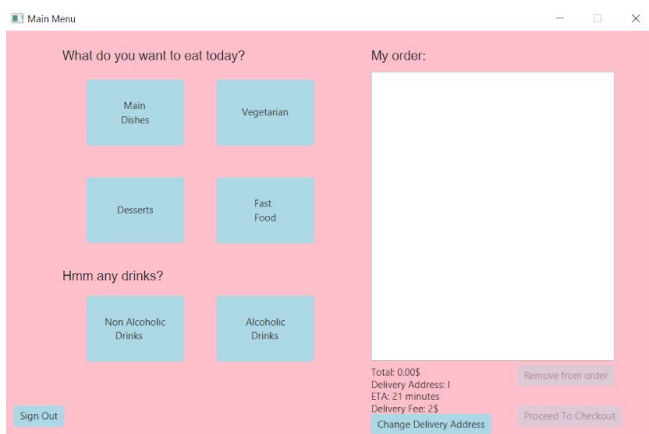
Also, apparently using one general ingredients list wasn't a good idea:

```
vegetarian7  
[turkey, chicken, pork]  
Calories: 126 Price: 16.72$ Ratings: 1.65/5  
vegetarian8
```

So, I created the second ingredient list specifically for vegetarian food to fix that not so little problem.

Display Main Menu

Once the sign-in/sign up is completed successfully, my app displays the general menu. Which looks like this:



I have a GeneralMenuPane class, which extends the Pane, class. I create an instance of this class in Main.java class, and once the sign-in button is pressed I change the root of the scene to generalMenuPane.

I decided to have different buttons for the different types of products. Once the button is pressed it calls the update method in [MenuPane](#) and the

scene root is changed to the menuPane. I also display the items that are added to the cart by the user.

Once the item is added, I make the check out button able to press. Also, if the user clicks the item on the 'my order' list, I make the remove button able to be clicked. I have the `removeFromCart(Product p)` method, which takes the Product object as an argument and removes it from the 'my order' list. It also updates the total price. I

also have the `addToCart(Product p)` method in GeneralMenuPane, which updates the 'my order' list and the total price.

I also have a change delivery address button, which displays MapPane and gives users a chance to choose a different address.

In this pane, I also added a sign out button. Once it is clicked, I empty the 'my order' list and disable relevant buttons.

Display (Food) Menu

I have a MenuPane that extends the Pane class. Using this pane, I display the concrete products(depended on the chosen button like Main Dishes, Fast Food, etc.). In this pane I have two buttons: go back to the main menu button, which simply changes the root of the scene back to generalMenuPane, and add to cart button, which calls [the addToCart](#) method in GeneralMenu Class and gives the selected item as an argument.



In this pane I have two methods:

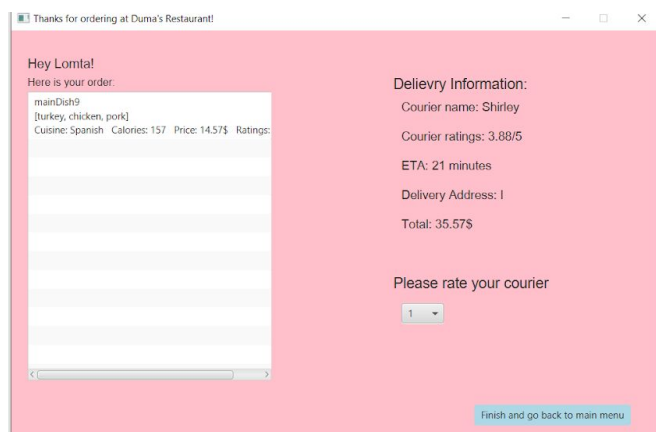
- `public void update(int a, ArrayList<Product> products)` - this method updates the displayProducts list, depending on the meaning of a. Argument a is an integer that is associated with the type of product. For example, if a == 1, it means the user wants to see desserts. I go through the products list and using instanceof operator, I find all the products that are instances of desserts and add them to displayProducts list.

- `public void sortList(ArrayList<Product> products, String selection)` - this method sorts the list based on the criteria that the user chose. I use a combo box, where the user can choose the criteria for sorting. I have 4 criteria: by name, by price, by calories, and by ratings. To sort the list, I use the Collection interface. Since I'm gonna need to sort any type of product instead of implementing this interface in each class, I implement it in the Product class. I have `compareTo(Object obj)` method and a static integer sortStrategy in the Product class. I change its value from MenuPane, depending on the sorting criteria. For the ratings, I sort the list in descending order.

Moreover, when the item is selected from the list I display an alert if the item might get cold on its way. For this, I use a Temperature interface, which has two methods: `int getColdTime()` and `boolean timeEnough(int time)`. Food and NonAlcoholicDrinks classes implement this interface, which means they both have this method. For those two classes, I have `private final int coldTime` and in the constructors, I assign them a random integer value returned by the private method `calculateTime()`. In the `boolean timeEnough(int time)` I compare the delivery time (which my method takes as an argument) and coldTime. `getColdTime()` returns the value of coldTime(what time does it take for this product to get cold).

Also for the toString() methods, since there were some similarities while displaying Desserts, Main Dishes, Fast Food, and Vegetarian products, I wrote a toString method in Food class, and I use the keyword 'super' to invoke superclass's toString() method.

Check Out



Once the check out button is pressed, I change the root of the scene to orderPane. orderPane is an instance of OrderPane class, which extends Pane class.

I have a listview, where I display the products that the user ordered. I also display all the data related to delivery. I have a combo box, using which the user can choose the ratings for the

delivery person. When the finish button is pressed I call the [update method](#) for the ratings and switch the root of the scene to [generalMenuPane](#).

Extension/Improvements and Summary:

As an extension, I was thinking about creating a delivery app not only for this restaurant but for many restaurants. For this, I would probably generate restaurant locations randomly and I'd have a restaurant class with several attributes like char address, Product menu, string name, etc. I actually could do that for this time too, but then I would need a bigger map and it might not be the best idea to implement Dijkstra's algorithm anymore. Also, as an extension, I wanted to add a timer that would count down the delivery time, and after delivery time is gone, I would make rating the courier possible only then.