**Food Ordering System**

Nguyen Tan Alex Le

Vanier College

420-SF2-RE: Data Structure and Object Orienting Programming

Yi Wang

May 11th, 2025

**Tables of contents**

1. Project Description..……………………………………………………………………3
   1. Scenario….……………………………………………………………………………………..……..3
   2. Desing Paradigm: User…………………………………………………………………………….3
   3. Expected Output…………………………………………………………………………………….3
   4. Specification………………………………………………………………………………………….4
      1. Hierarchies…………………………………………………………………………………..4
      2. Interface……………………………………………………………………………………...4
      3. Runtime-polymorphism…………………………………………………………………4
      4. TextIO………………………………………………………………………………………....4Comparator and Comparable………………………………………………………….4
      5. Class diagram……………………………………………………………………………….5
2. Program Feature and Screenshots…………..…………………………………….6
   1. Exception with file handling………………………………………………………………………6
   2. Hierarchies (Order and User)…………………………………………………………………….6
   3. Methods working (without export)…………………………………………………………..…7
   4. Refund method working (without export)…………………………………………………….7
   5. TextIO……………………………………………………………………………………………………7
   6. Comparable and Comparator…………………………………………………………………..8
   7. Unit testing…………………………………………………………………………………………….8
   8. Interface with at least one abstract method………………………………………………..8
3. Challenges Encountered………………………………………………………………9
4. Learning Outcomes……………………………………………………………………10

**1. Project Description**

**1.1: Scenario**

The Food Ordering System (FOS) is a small-scale application that allows any restaurant to manage operations more easily. The system supports multiple roles including the worker, customer and driver. The customer can create an order and pay for the order. They can also view the menu. The workers and driver manage orders and handle deliveries.

**1.2: Design Paradigm: User**

Worker:

* Check if the food is on the menu
* Refund order if it is wrong
* Keep track of each category food is ordered
* View the menu
* Add and remove food from the order

Customer:

* Search foods with keywords in name
* Search food by exact name
* Create order
* Add and remove food from the order
* View the menu
* Pay for food

Doordash driver:

* Complete orders
* View orders

**1.3: Expected Output**

The worker can also view the menu. The food can be searched by its category of food and its name. The system will also allow them to keep track of the orders. The worker and customer can search for the food on the menu whether it is there, they can also refund the customer in case of a mistake. They are able to create orders, add and remove food to order. It allows them to process orders from the customer. They check if the food is on the menu to manage their requests. Customers can also search for the food based on category and name. They are also able to view the menu. The customer will be able to order too and add or remove food from the order. Lastly, the drivers can manage the orders they get, and they can view the order they receive. They can complete orders. This is a food ordering and delivery platform with roles for customers, workers and drivers.

**1.4: Specification**

1.4.1: Hierarchies:

To specify the program, there would be a User class and that class is the superclass or the Parent class. This class will allow us to put in all common data fields and common methods for all the Child class to be able to implement these methods without rewriting the whole thing every time. The Child classes of User class would be Worker, Customer and Driver since these classes will inherit from the User class. The other class that would be a parent class is Order, and its child classes would be DeliveryOrder and InRestaurantOrder. These two subclasses inherit from Order class. There will also be the Food class, but this class is its own independent class just like Order class. It won’t be related to the User class.

1.4.2: Interface:

For the interface, we will have an Orderable interface, so that this allows the workers and customers to create orders. This interface will also allow them to add or remove food from their order. The reason why there is an interface Orderable is because multiple classes will implement the charge method, but not all classes. It would be long to rewrite it for each class that implements it.

1.4.3: Runtime-polymorphism:

There are a couple methods that will use runtime polymorphism. For example, viewing the menu or searching for specific food or displaying their info too. The add and remove food method will also apply runtime polymorphism, because there are two classes that will use that method and so we’ll need to specify which one specifically.

1.4.4: TextIO:

For the text IO, it would be in a Restaurant class, because it would export information of completed delivery orders and normal orders in a csv file. In the text IO file, it will add the name of the customers, the foods of their order and the number of their order, the time they paid for it and the order type. It also shows the orders that the workers refunded. This will allow the workers to find old orders and possibly refund orders using the text IO to check old orders to verify. Multiple methods will export the orders to the csv files like the pay method.

1.4.5: Comparator and comparable:

The customer class would implement both comparator and comparable, because keeping track of customer data is more important. The customers wouldn’t care about the data, but the restaurant and the admins would. It would be easier for them to keep track of data. The Food and Order class will definitely implement comparator and comparable to sort the food and the number of order. It will be easier to organize the data. Workers could implement a comparator since we might also need information on the workers to keep their data and make it easier to sort and organize the data.

1.4.6: Class diagram: //TODO

A grid of paper with text

AI-generated content may be incorrect.

**2. Program Feature and Screenshots**

2.1: Exception with file handling

A computer screen shot of a program

AI-generated content may be incorrect.

2.2: 2 hierarchies (Order and User)

**A screenshot of a computer

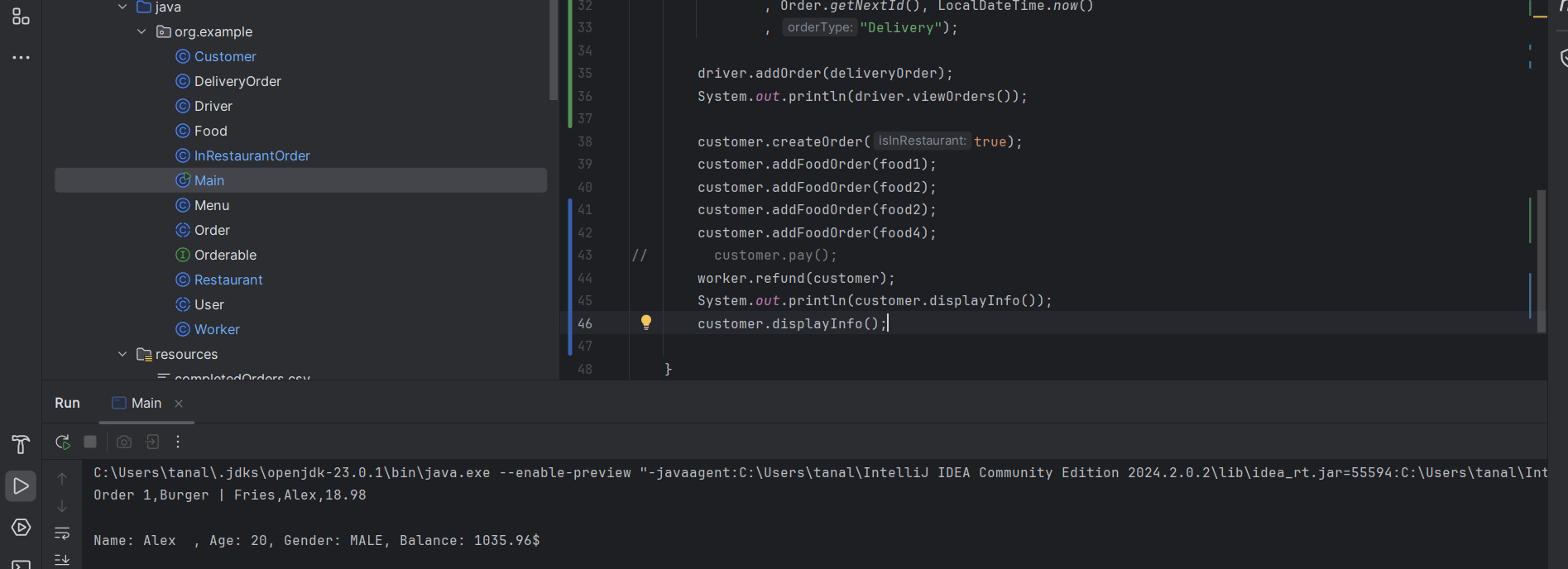
AI-generated content may be incorrect.**

2.3: Methods working (without export)

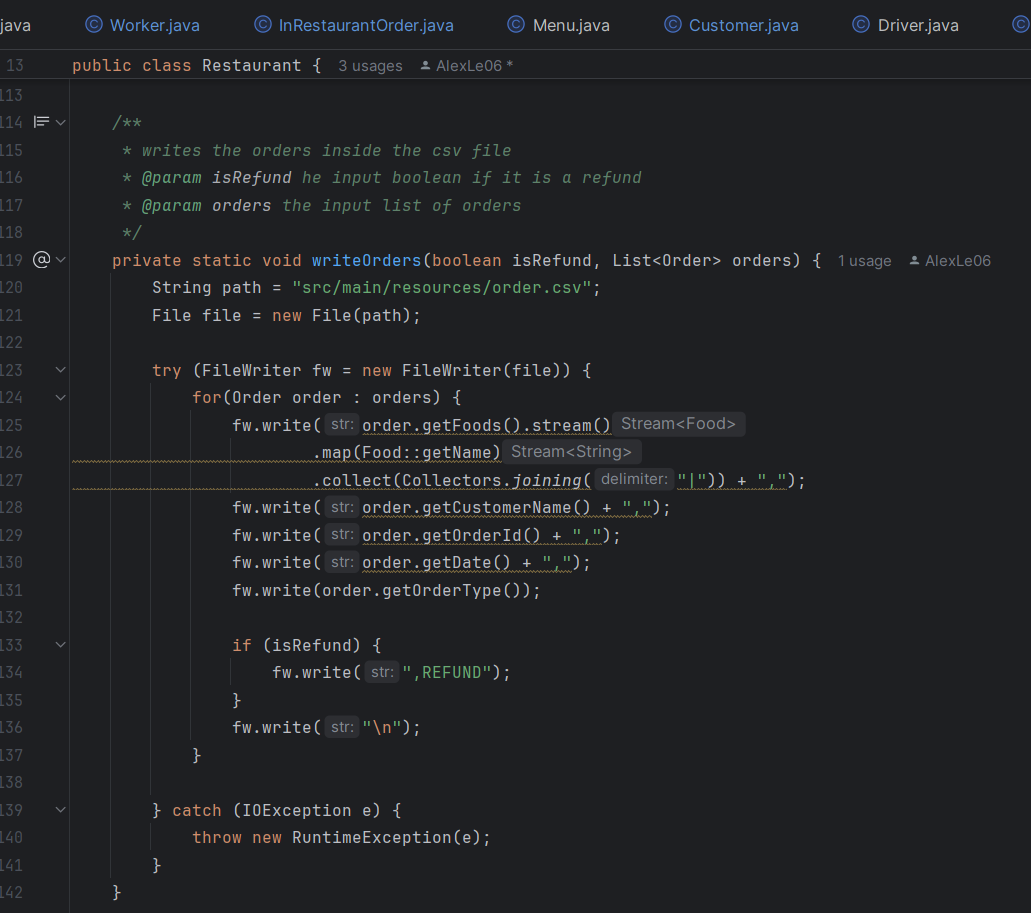
A screenshot of a computer program

AI-generated content may be incorrect.

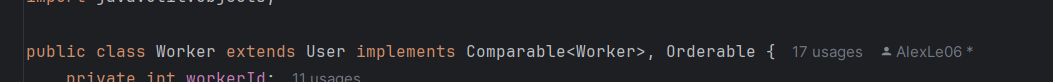
2.4: Refunding method working (without export)

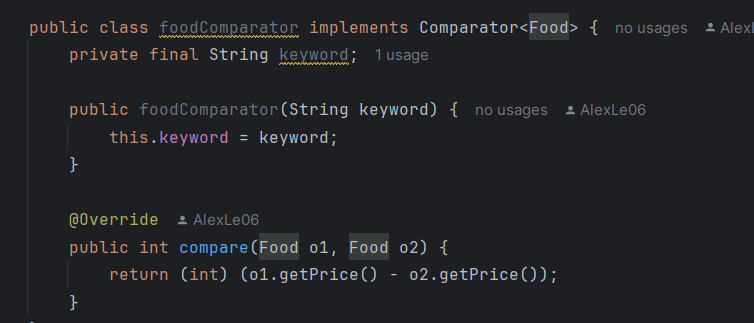


2.5: TextIO

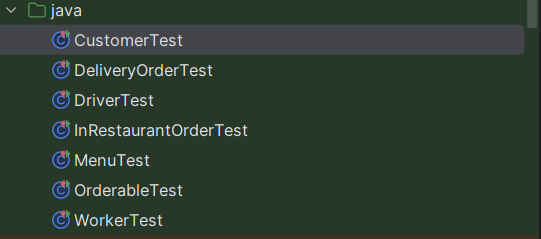


2.6: Comparable and Comparator



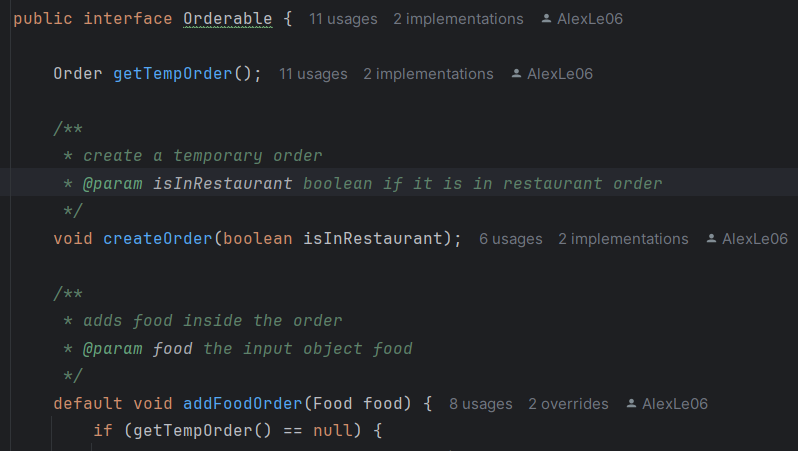


2.7: Unit testing





2.8: Interface with at least one abstract method



**3. Challenges Encountered**

During this project, I’ve faced many obstacles like methods not working, missing details, no connections between some classes and many more. For example, in my Deliverable 1, I specified that there would be a Chargeable interface, but I realized later that this interface would be useless if Customer class already had a pay method. This interface would only work if the worker and driver would make them pay later and not at the same time as the customer paying for their order. So, I had to remove this method, so that my project would make sense. Furthermore, I added a whole new class named Restaurant so that it makes it easier to store the information of the orders inside a csv file, because at first I tried to get the orders to go inside a csv file, but it would probably only just get one since there were no List of orders that I could export into that file. So, I had to create four whole new methods that would allow to load orders and write orders inside the csv file. I needed to also get a method for customers and workers that allowed them to create a new order, because in my Deliverable 1, there wasn’t any way to get them to have specific orders to their name. I thought it would’ve been easy to have a Driver method where they could manage orders. They would be set the order status to refused, accepted or have it just pending, but it would’ve been confusing so I changed it so that, it would just be pending or completed. Unfortunately, there is still a problem with the text IO that I couldn’t resolve. When the customer pays, it adds the order to the restaurant order and exports the information, but it seems that there is an array out of bound exception with the method load order. I think it is because of the way I export and get the order from a csv file. The food List parameter of the order is a list of object Food, but in the csv file there’s only the food names and so there isn’t any information about the food. I’ve encountered many other challenges including challenges with unit testing. Another challenge that I met was when I had to create the method createOrder, I was suggested to add a data field of Orderable operator, but I didn’t understand what it would do. Sometimes, I noticed that my unit testing didn’t really correspond with what I was supposed to compare. The assertEquals line wouldn’t make sense sometimes. Lastly, one of the challenges I faced was managing to do all methods. There were some methods I missed like the comparator methods, because I had forgotten to write //TODO even though I implemented it.

**4. Learning Outcomes**

In the end, I gained many things from this project. I learned many things for example, planning a project is very hard, but it will be very useful later on. This allowed me to be organized and have an idea of what to do and what it would be like. There were big changes in this project, but this doesn’t stop the fact that it allowed me to get an idea of what I had to do. If I just started programming right away, I would’ve been lost. This experience allows me to do the same thing soon when I would like to create a project for myself or some kind of event like Hackathons. This project also helped me refine my skillset in everything like inheritance, polymorphism, unit testing, OOP and file handling. This project also made me debug more than I am used to. I don’t debug often when doing assignments which allows me to have a better understanding of debugging. Debugging helped me overcome challenges and bugs. It made me use my critical thinking skills to solve problems I encountered. I also learned how to use git hub more and how to initialize repositories, add, commit and push in files. This will help me in the future when I have group projects to do. This project also helped me realize that time management is very important. For example, for Deliverable 2, I did the project a bit too late in the week and this made me realize that making projects isn’t easy especially when I had to do unit testing. Some of the unit testing I did when doing Deliverable 2 didn’t make sense or didn’t work. This made me understand for Deliverable 3 that I would need more time to work on this. This project taught me that unit testing is also important. It makes sure that the code is reliable and that it should work. For example, in the Restaurant class, I didn’t have unit testing for it and so it was harder to work on than the rest of the methods and classes. Moreover, I learned that file handling a list where I try to export only the food names isn’t a good way to do it. It also gives me a glimpse of how I may have to program in real life situations. This gives me experience and possibly will help me in the near future.