2. Software Using Documentation

2.1. Software Usage

The program works by taking three different files. Program reads **“customer.txt”** and **“order.txt”** directly and the program takes “input.txt” as command line argument.

- “customer.txt” an input file with 5 different personal information of each person. (<customerID>,<customerName>,<customerSurname >,<phoneNumber>,<adress>) .
- “order.txt” contains a list of orders.
- The last file is “input.txt” and the program applies this commands line by line.

3. Software Design Notes

3.1. Description of the Program

3.1.1. Problem

In this experiment, we are expected to write a program that works with commands from an input file. The program is expected to hold, edit, and process customer lists and their orders, and the program consists of two different parts. In the first part, transactions about the customers are done. In the second part, the orders are taken and the pizzas are being processed.

- Actions for customers include adding a new customer, deleting an existing customer, and printing the last version of customer list.
- Order operations : Creating a new order, adding a pizza to the order, adding a beverage to the order, and learning the final price of the order.

We are expected to use DataAccess Objects (DAO) to manage the data and Decorator pattern to add new toppings to Pizzas.

3.1.2. Solution

I created separate classes for pizza types and toppings and associated them through an interface. In the same way, I opened two classes named customer and order to manage the data, and I contacted the DataAccessObject interface I created.

I have two structures which hold the data.

CustomerList which contains all the customers program takes, is an ArrayList and each customer has its own customerID, name, surname, phonenum and address. Using this list is an ideal way to make it easier to access customer information when needed.

```
String customerID,customerName,customerSurname,phoneNum,address="";
static ArrayList<Customer> customerList = new ArrayList<Customer>();
```

OrderList which contains all the orders program takes, is an ArrayList.

```
String orderID,customerID;
boolean drink = false;
ArrayList<String> pizzaType = new ArrayList<String>();
static ArrayList<Order> orderList = new ArrayList<Order>();
```

4. Class Diagram