Department of Computer Science & Engineering

BBM104 Introduction to Programming Laboratory

Experiment 3

EMRE HANCI - 21604552

Programming Language : Java

Submission Date : 11.04.2018

Due Date : 25.04.2018

Advisors : Dr. Gönenç Ercan, Dr. Öner Barut, Dr. Cumhur Yiğit Özcan, Dr. Ali Seydi Keçeli, R. A. Pelin Canbay

-What was the problem?

In this assignment, the expectation from us write Java code of pizza restaurant system in this system user shuold be able to create an order, add a pizza with toppings or without topping, add a drink, paycheck or user should be able to create, add or remove customer. Program should take the command of user from input.txt, the customer information from customer.txt and the order information from order.txt and write the output of program in to input.txt. When program does something like add an order or add an pizza this progress should been proccesed by Decorator Design Pattern and the proccess about customer or order should been proccesed by Data Access Object. Before the end of program, program should write the information of customer to customer.txt and also write the information of order to order.txt

input.txt

In this file, there are command lines which are the progress the program. There are eight type commands.

CreateOrder <OrderID> <CustomerID> : When this command execute, program create order and add it to ArrayList which is store the orders.

RemoveOrder <OrderID> : When this command execute, program remove the order whose id given with command from ArrayList.

AddPizza <OrderID> <PizzaType> <Toppings> : When this command execute, program add a pizza which is type and toppings given with order id. Number of toppings can be 0-3 but there ara four types of toppings which are Hot Pepper, Soudjouk, Salami, Onion, and there are two types of pizza which are American Pan and Neapolitan.

AddDrink <OrderID>: When this command execute program should be able to add a drink to given order id. There is one type of drink and it is value 1$

PayCheck <OrderID>: When this command execute program should calculate the cost of order and print it in output.txt like a bill.

AddCustomer <cutomerID> <name> <surname> <phone number> <address>: When this command execute, the program should add the customer which is informaiton given with this command.

RemoveCustomer <cutomer ID>: When this command execute, the program should remove the customer whose customer id given with command, and also program should remove the order which is given by removed customer.

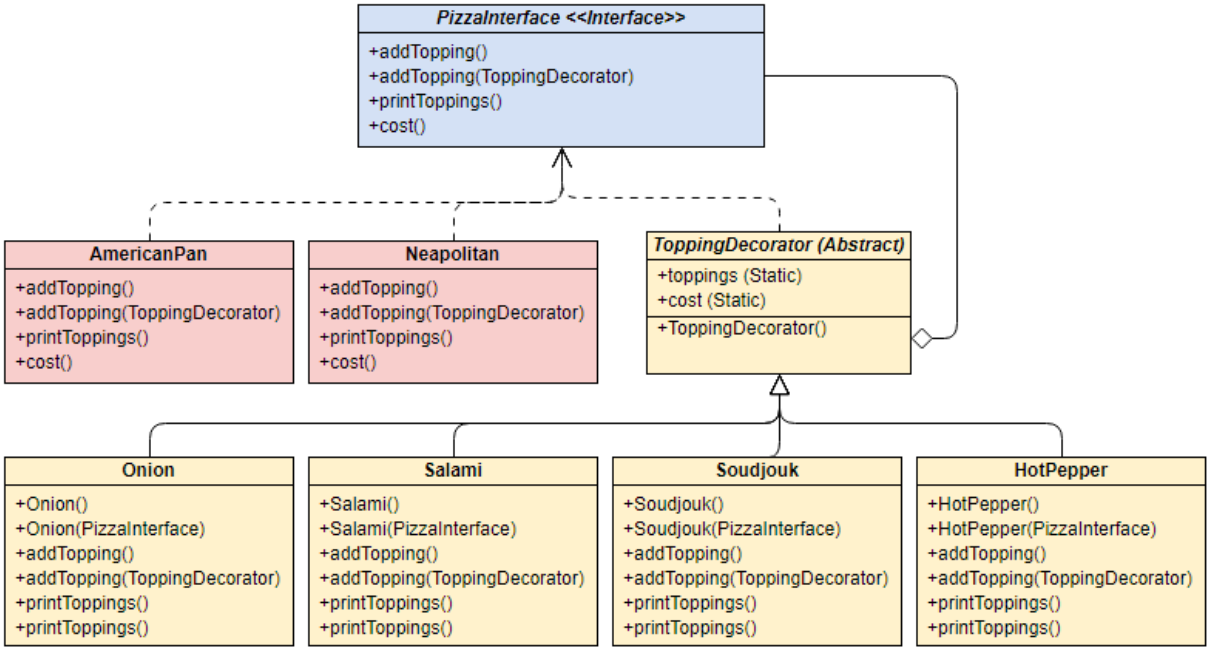
ListCustomers: When this command execute, program should write the customer list ordered by name.

-My solution

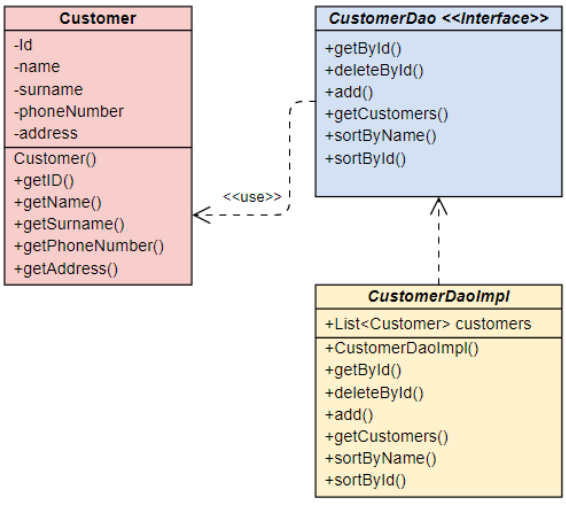
For the solve this problem I used decorator pattern and data access object. My solution has sixteen classes.

Main.java: The brain of the program, in this class program read the command from input and

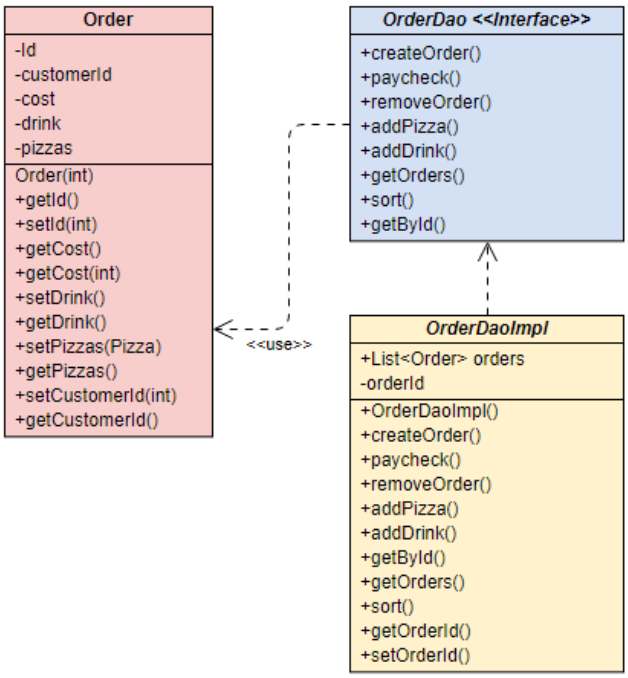
call the method needed. I use reflection for the process line
pizza.addTopping(new Salami(new HotPepper(new Soudjuk())));
Pizza.java: This is for writing code more readable, it is store information of added pizzas.
PizzaInterface.java: This is an interface which is implemented by AmericanPan.java, Neapolitan.java and ToppingDecorator.java.
AmericanPan.java and **Neapolitan.java**: Those two java file implements PizzaInterface.java and overrides the methods.
ToppingDecorator.java: It is an abstract file, it is implements PizzaInterface and store static topping variable, static cost variable.
HotPepper.java, **Soudjouk.java**, **Onion.java** and **Salami.java**: Those four java files extends the ToppingDecorator.java file, and this classes modify topping variable in ToppingDecorator.java
-UML Class Diagram of Decorator Design Pattern:



Customer.java: The purpose of this java file is create a customer.
CustomerDao.java: It is an interface for using Data Access Object Pattern.
CustomerDaoImpl.java: This java file implements CustomerDao.java read and store the customer information from customer.txt and proccess methots when they are called.
-UML Class Diagram of Data Object Pattern:



Order.java: The purpose of this java file is create a order.
OrderDao.java: It is an interface for using Data Access Object Pattern.
OrderDaoImpl.java: This java file implements OrderDao.java read and store the order information from order.txt and proccess methots when they are called.
-UML Class Diagram of Data Object Pattern:



-The algorithm of my program

The algorithm of my codes;
- customerDao object created from CustomerDao.java file and assign to new CustomerDaoImpl().
- CustomerDaoImpl() called and it's create a customer arraylist from customer java file
  - Read the customer information from customer.txt line by line and customer to customer list.
- orderDao object created form OrderDao.java file and assign tı new OrderDaoImpl().
- OrderDaoImpl() called.
  - Order information from order.txt loading to arraylist.
- Output file opened.
- Input file opened.
- Reads input file line by line
  - When AddCustomer command given.
    -customerDao.add(line) called.
      -New customer created and added to customer arraylist.
      -Customer arraylist sorted by name.
      -Return an string.
  - When RemoveCustomer command given.
    - Search the order id of given customer id when it founded, orderDao.removeOrder() called.
      - orderDao.removeOrder remove the order from arraylist.
    - customerDao.deleteById() delete the customer from arraylist.
    - sort arraylist by id.
  - When CreateOrder command given.
    - orderDao.createOrder called.
      - New order created and added to arraylist.
  - When RemoveOrder command given.
    - orderDao.removeOrder called.
      - Order removed from arraylist.
      - Sort the arraylist by id.
  - When AddDrink command given.
    - orderDao.addDrink called.
      - Drink added to given order id.
  - When ListCustomers command given.
    - customers from customer arraylist print.
  - When AddPizza command given.
    - Program check the number of toppings given then directs to case.
      - Create a pizza and add toppings with given command.
-Input file closed.

-Customer file opened.
-Customer ArrayList sorted by Id.
-Customers wrote to customer.txt.
-Customer file closed.
-Order file opened.
-Order ArrayList sorted by Id.
-Orders wrote to order.txt.
-Order file closed.
End of Program.

-Program will be handle with these situations;

-When adding a customer with an id which is already taken by a customer.

Program prints;

-*There is a customer with same unique id therefore this customer could not added.*

-When adding a customer with missing field.

Program prints;

-*There is some missing field.*

-When removing a customer which is not exist.

Program prints;

-*There is no customer with this id.*

-When creating an order with an order id which is already taken by a order.

Program prints;

-*There is an order with this id.*

-When creating an order with an customer id which is not exist.

Program prints;

-*There is no customer with this id.*

-When adding a drink to an order which is not exist.

Program prints;

-*There is no order with this id.*

-When adding a pizza to an order which is not exist.

Program prints;

-*There is no order with this id.*

-When paycheck command given with an order id which is not exist.

Program prints;

-There is no order with this id.

-When adding more than three topping to a pizza.
Program prints;
-A pizza can include max 3 toppings.

-When adding a pizza which is not exist.*
Program prints;
- There is no type pizza.

-When adding a topping which is not exist.*
Program prints;
-You can not add this toppings.

-Things program check when it execute.

-If in order.txt has an order whose order id is same with another order id program will not load it to program.

-If in order.txt has an order which is has a pizza with more than three toppings, program will not load it to program.

-If in order.txt has an order which is has a pizza with a topping which is not exist, program will not load it to program.

-If in order.txt has an order which is has a pizza that base type* is not exist, program will not load it to program.

-If in customer.txt has more than one customer with same customer id program only load first one to program.

*: If another type of pizza class created or another type of toppings created, those rules not need to update because they check them on runtime.