HACETTEPE UNIVERSITY DEPARTMENT OF COMPUTER ENGINEERING
BBM104

Prepared by Burak YILMAZ
Number: 21627868
E-Mail: burak040898@gmail.com
Subject: Assignment 3

MAIN GOAL

In this assignment, our main goal is using inheritance, polymorphism, how classes interact and basic input-output operations in Java. Also design our code to a specific pattern called "Decorator Design Pattern". Moreover, while writing the output again using a specific pattern called "Data Access Objects(DAO)" to manage our data.

PROBLEM

In this problem, we need to write a Java code that develop a Pizza Restaurant System. Our code should provide following features.

-AddCustomer

-RemoveCustomer

-ListCustomer

-CreateOrder

-AddPizza

-RemoveOrder

-AddDrink

-PayCheck

ALGORITHM

According to my algorithm firstly, I tried to design my decorator pattern because the most important part of this assignment was that. I created an abstract class which name is "Pizza". This class has-a-relationship with "Pizzatopping" class because "Pizza" class has a variable type of "Pizzatopping" and then created a decorator called "Pizzatopping". After that I extend this "Pizzatopping" abstract class with "Pizza" class, so they interact each other just the way I want. After that for each topping class (Onion, Salami, Soudjouk, HotPepper, NoTopping) I extend "Pizzatopping". They all have a variable type of "Pizzatopping" except "NoTopping" class. Therefore, If I add any topping later to my code it is very easy to add without a few changes. Also, I create two class (AmericanPan, Neapolitan) which these extends "Pizza" class. Besides I create "Order" class to keep an order, and "Customer" class to keep each customer. With all that classes I completed my decorator pattern and a huge part of my code.

Moved on with Data Access object part. I created three Interface for that and three implementations of these Interfaces. Each implementation only busy with specific operation like "OrderDaoImplementation" class just for writing orders to order.txt. After finished that part only thing left was reading the input txt and do the right operations for each command. Finally, I created a "Command" class to read the input file and for each command It continues with correct If/else statement.

!!! Command line argument must be java Main input.txt

!!! HotPepper, AmericanPan, Neapolitan, Onion, Soudjouk, Salami classes so the pizzas and toppings in input should be like this because code is working with Class.forName():

Classes and Interfaces

Pizza Class: This is an abstract class it has got three methods and a variable. This class has a relationship with pizza topping class.

Neapolitan, AmericanPan Classes: These classes extends Pizza class so they can access all methods and variables. Also, these classes have overridden methods of Pizza class.

Soudjouk, Salami, NoTopping, Onion, HotPepper Classes: These classes extend the PizzaTopping class so by extending that class they override the methods of Pizza class and can access the variable of Pizza class.

OrderClass Class: This class keeps each different order. It has got several private variables and get methods.

OutputDao, OrderDao, CustomerDao Interfaces: These interfaces have got own different methods.

OutputDaoImplementation, OrderDaoImplementation, CustomerDaoImplementation Classes: These classes implement their own Interfaces so they have overridden methods.

SoftDrink Class: This class has got only two private variables and get methods of these variables.

PizzaTopping Class: This abstract class actually does not have any methods or variables. Actually, this class is decorator by implementing Pizza class it can access the methods and variables.

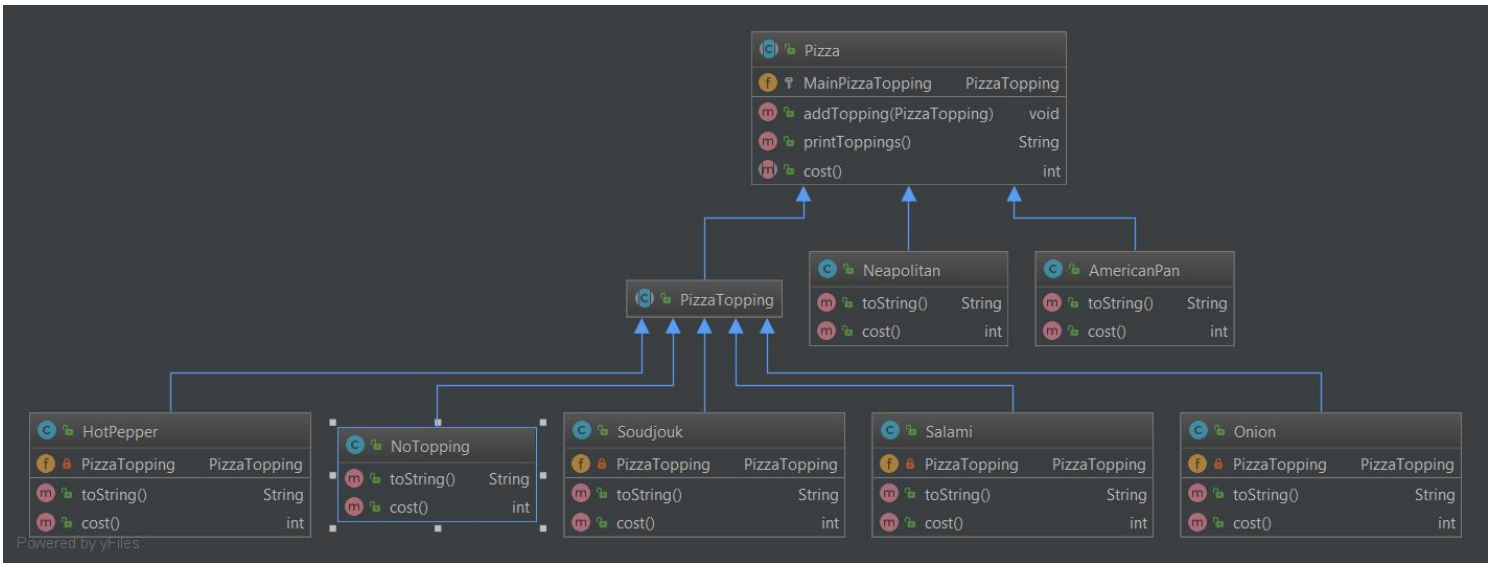
Command Class: This class reads the input.txt and decided which methods are going to use or which variable are going to create.

Main Class: It runs the whole code.

Customer Class: This class keeps every different customer. It has got several private variables and get methods.

UML DIAGRAMS

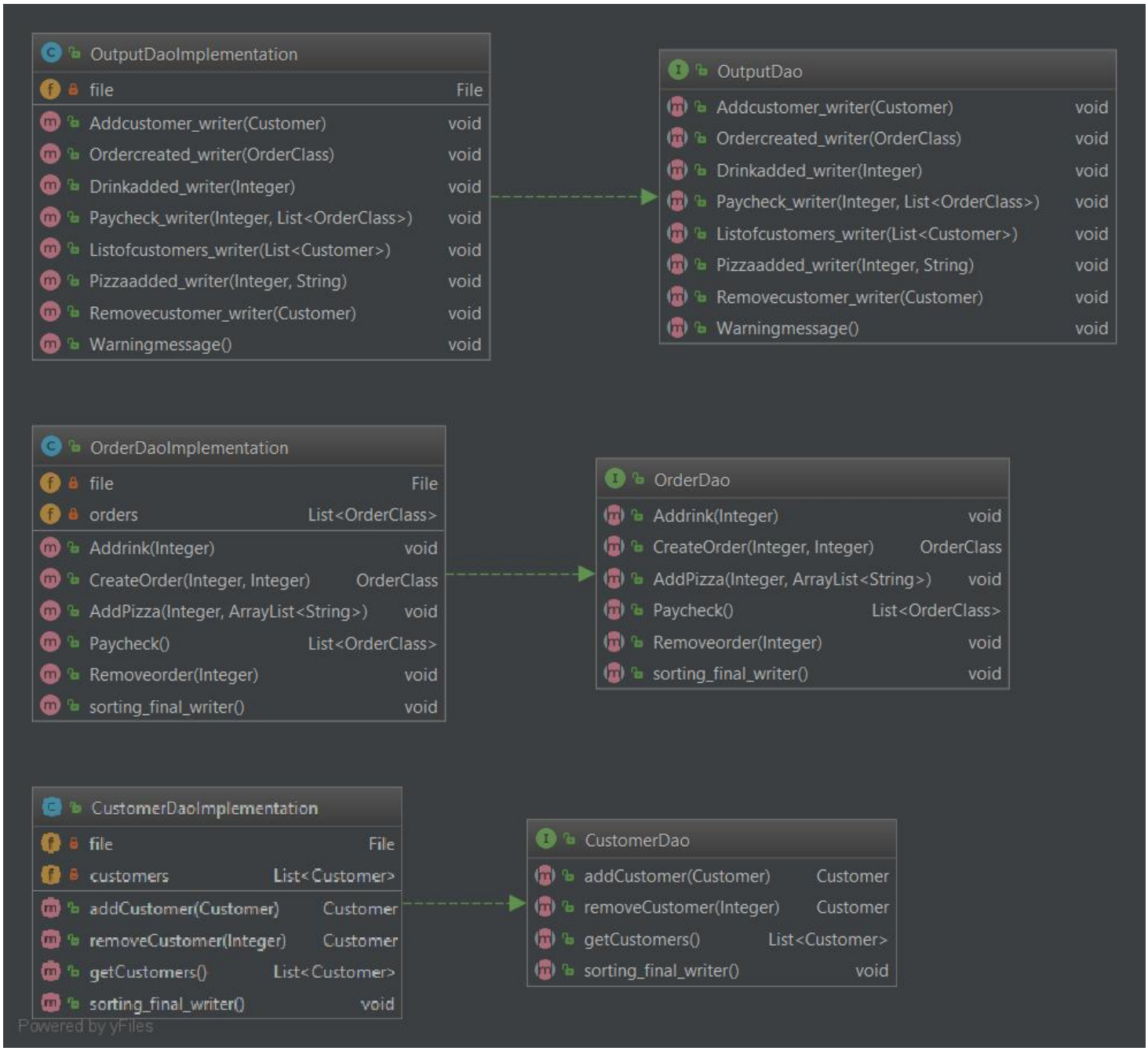
Decorator Design Pattern Uml Diagram



This uml diagram is showing the decorator design pattern. I created an abstract class called "Pizza" and a decorator called "PizzaTopping" extends class "Pizza". Also, I created two types of pizza class called "Neapolitan" and "AmericanPan" therefore these types of pizza classes can access the methods and variables of their abstract class "Pizza". These classes extend also abstract class "Pizza". Finally created topping classes ("Salami, Soudjouk, NoTopping, Onion, HotPepper) and these

classes extends "Pizza" class because of that these toppings classes can access the methods and variables of abstract "Pizza" class.

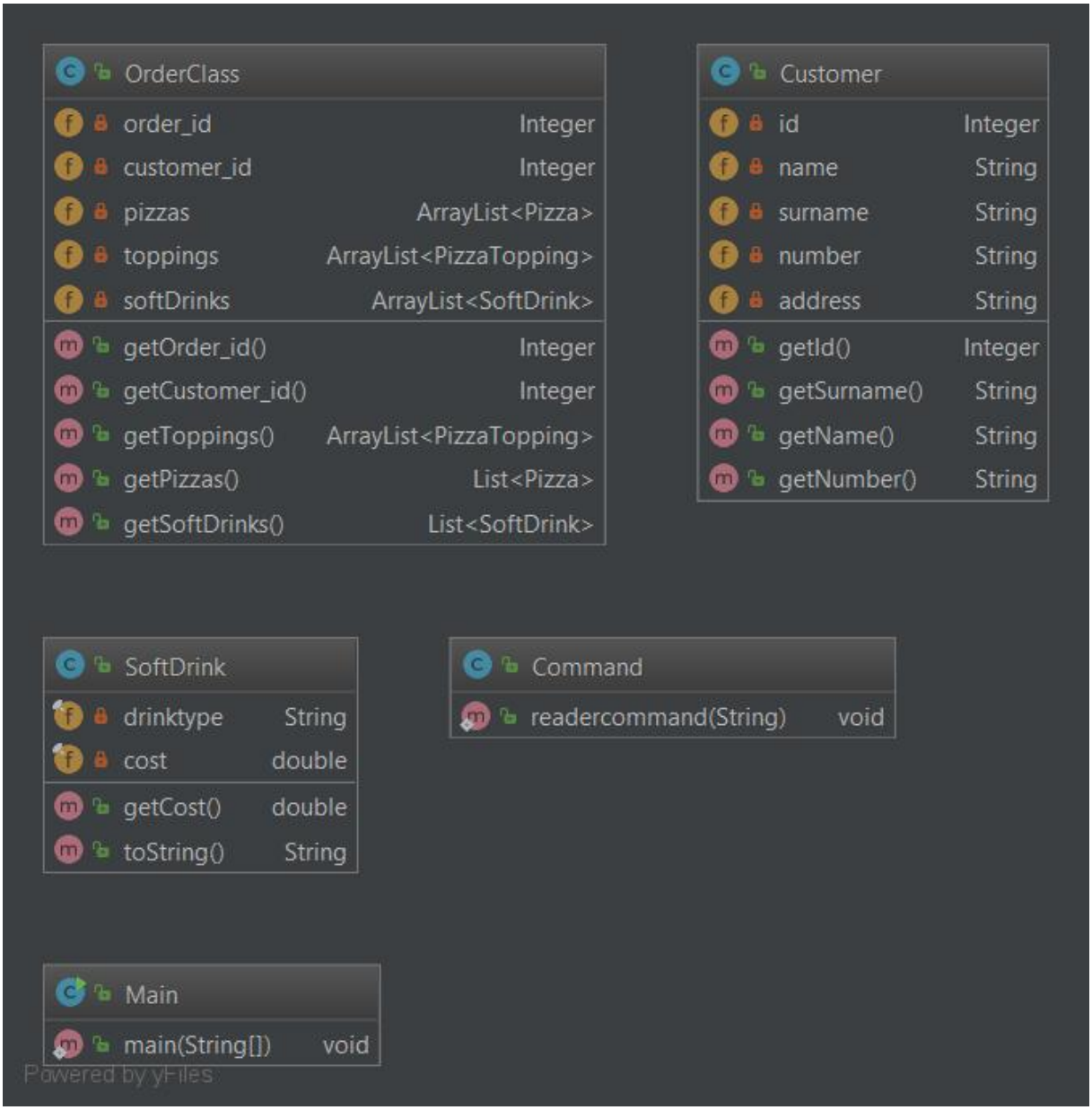
Data Access Object Uml Diagram



This uml diagram is showing the Data Access Object (DAO). I created three interfaces and three Implementations of these interfaces. Each interface has

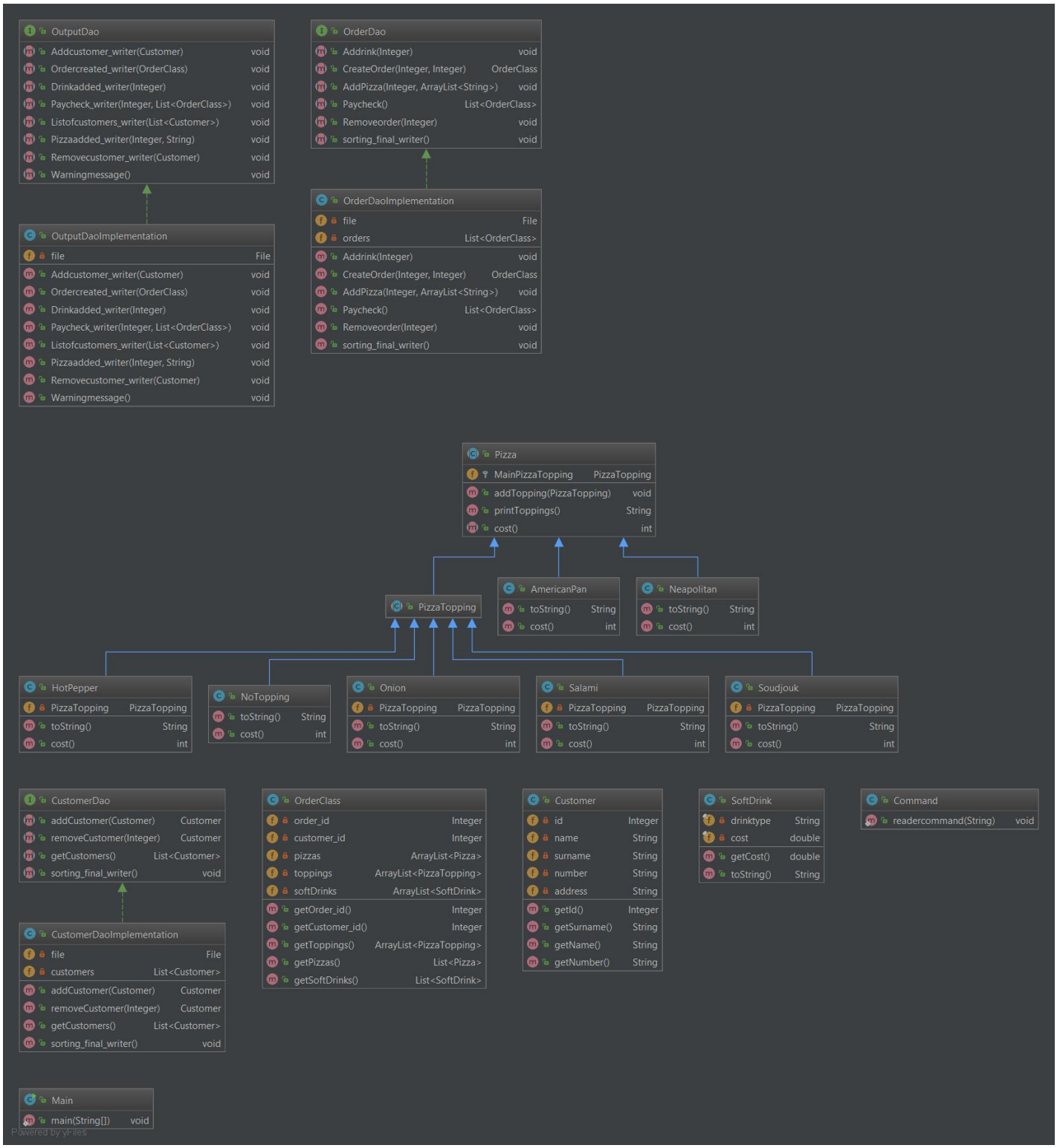
got own methods. By implementing these interfaces with the implementation classes each class can access the methods of their interfaces.

Dependent Classes Uml Diagram



These classes are independent classes. They did not implement any interface or extend any classes. They have their own variables and some of these classes have their own methods.

Complete Uml Diagram of Code



This diagram is showing the full uml diagram of this code. Above diagrams I clarify every part of code.