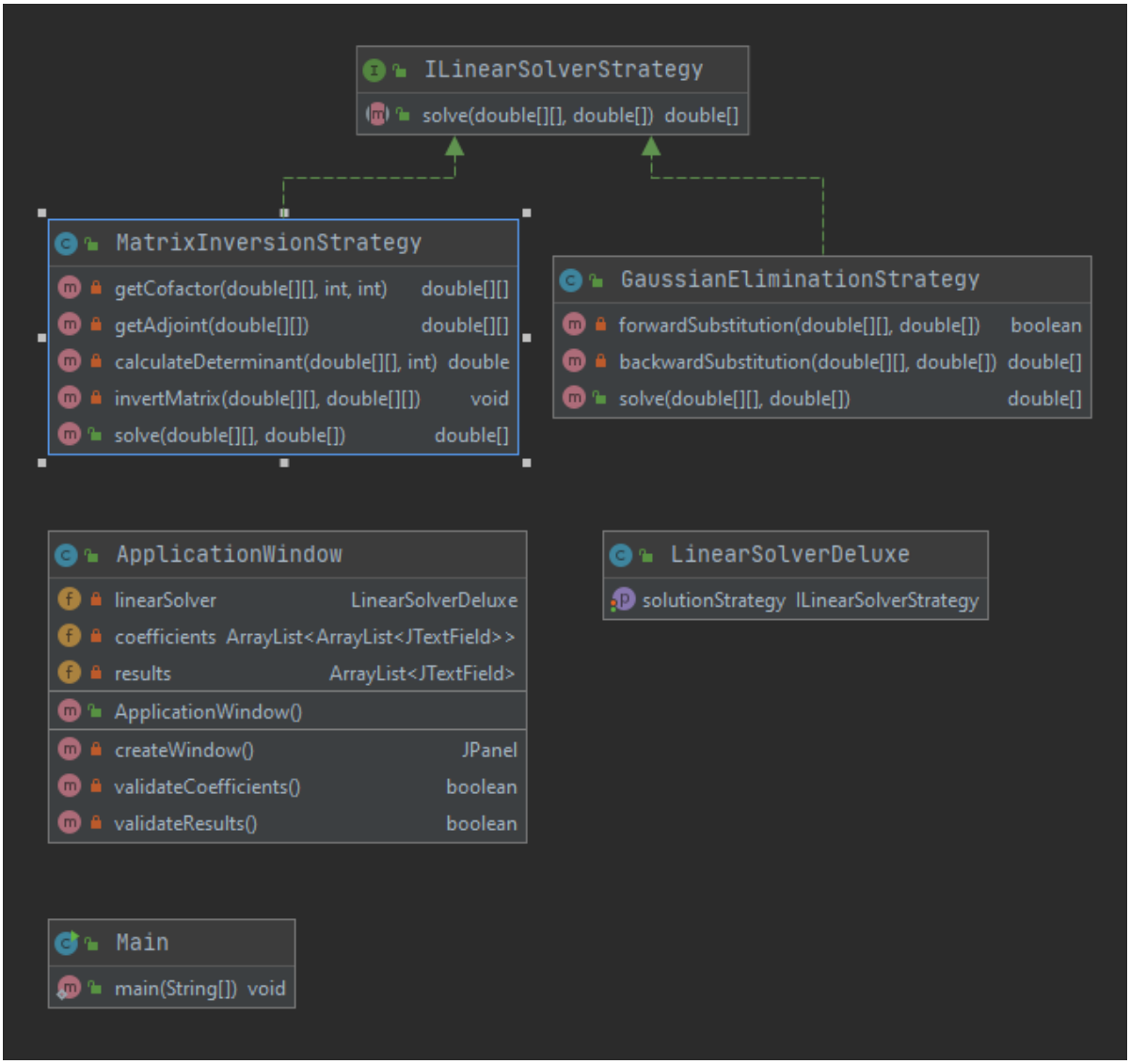
CSE443 – Object Oriented Analysis and Design Homework #1 – Report

Gökçe Nur Erer – 171044079

Question #1

Class Diagram



What kind of design supports to solve this problem?

In this question, strategy pattern has been implemented to solve the problem because there are multiple behaviors of solving an equation. An interface is defined to represent the linear solving strategies and Gaussian, Matrix Inversion strategy classes implemented this interface which has solve method.

How to use the graphical user interface?

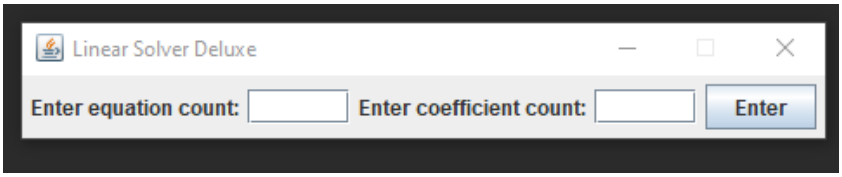
First equation count and coefficient count per equation is entered. To keep in mind every equation has to have the same number of coefficients. After entering the equation and coefficient counts, coefficient and results has to be entered, they can be positive or negative or 0. After entering the values from the dropdown either Gaussian or Matrix Elimination strategy can be selected. Then clicking on calculate will show the solutions of the equations.

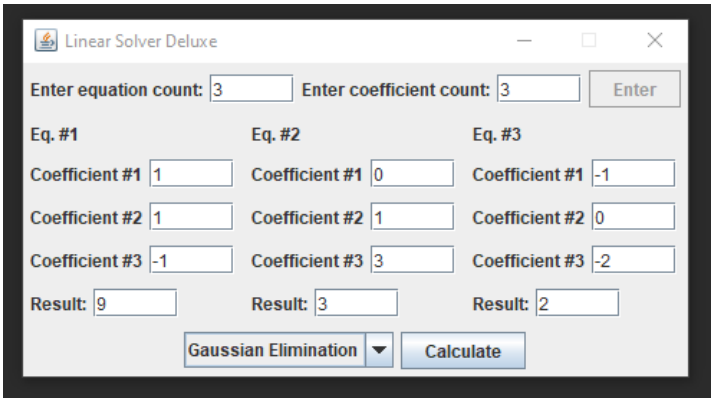
Example run for equations:

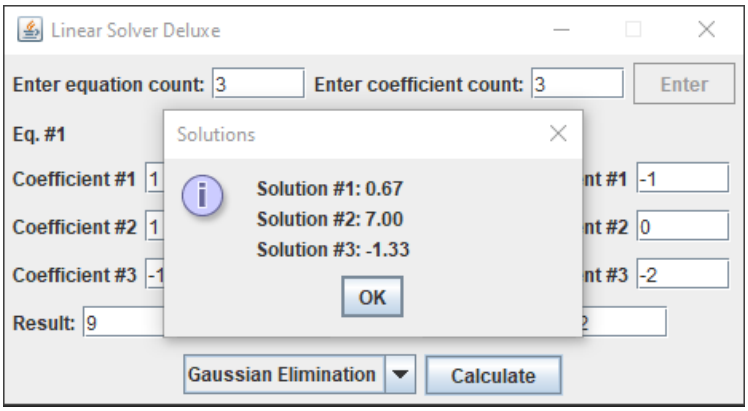
$x + y - z = 9$

$y + 3z = 3$

$-x - 2z = 2$







Question #2

What kind of design supports to solve the problem?

In this question, observer pattern has been implemented to solve the problem. This pattern contains an observable class and single or multiple observers which can get notified if something changes in the observable class.

To solve the question first the interfaces ***"IObservable"*** and **"IObserver"** are created which creates a baseline for the concrete classes which is created later. ***"IObservable"*** interface includes method definitions for adding, removing and notifying subscriptions. ***"IObserver"*** interface includes the method definition for updating itself.

Then the content classes which are ***"TextContent"***, ***"AudioContent"*** and ***"PhotoContent"*** are created to represent the data contents in the websites. Each class has a data instance which is a String, Clip or Image respectively.

Later, ***"User"*** class is created to represent users in the system. Users contain name, surname and subscription list only.

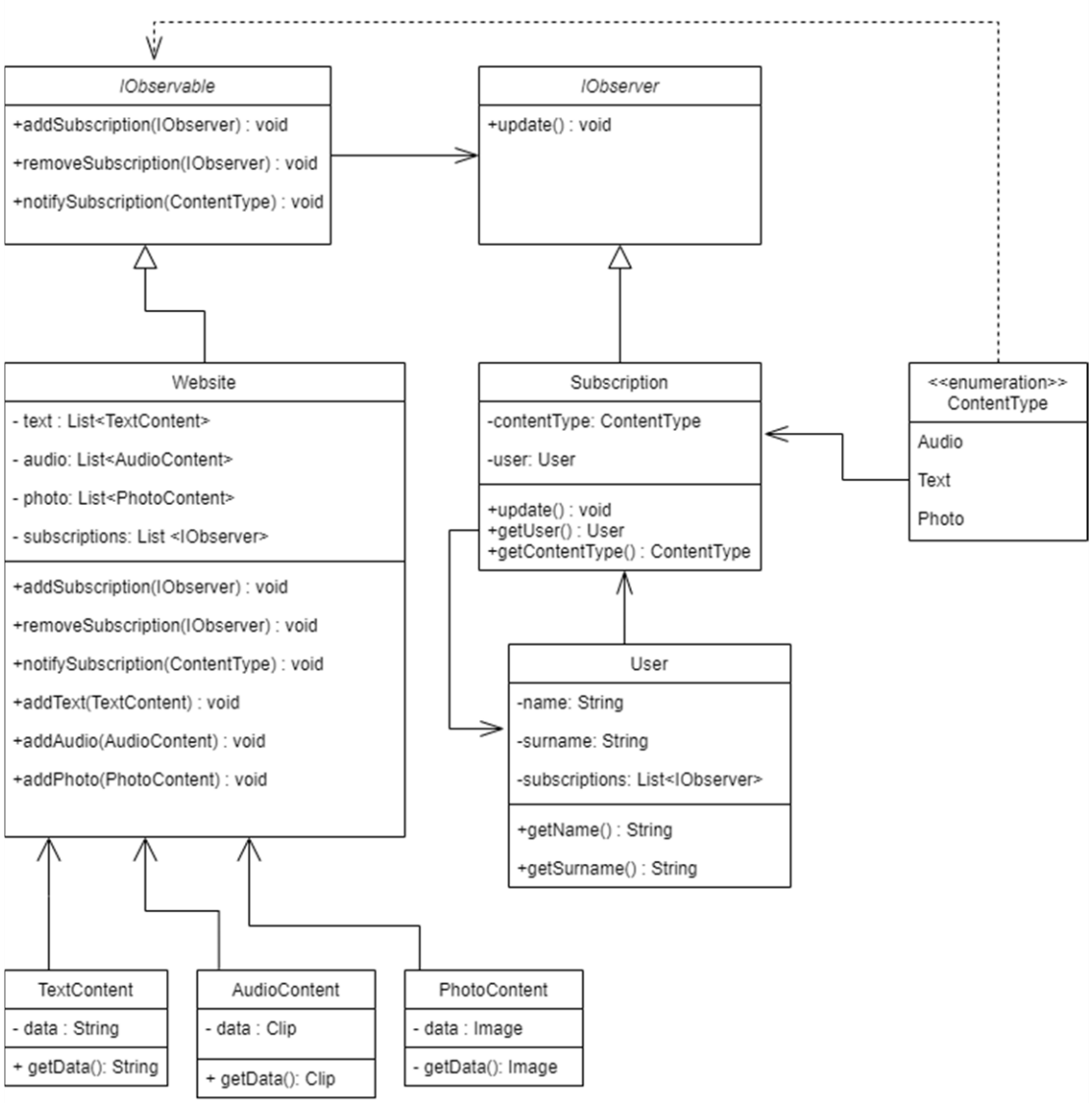
At next, the ***"Subscription"*** class is created to represent the subscriptions as a concrete implementation of ***"IObserver"*** interface. ***"Subscription"*** class contains a content type which is defined as an enum type (which can either be audio, text or photo) and a user which represents the user who subscribed the website. It implements the update method coming from "IObserver" interface.

After that the "Website" class is created to represent websites as a concrete implementation of ***"IObservable"*** interface. ***"Website"*** class has lists of ***"TextContent"***, ***"AudioContent"*** and ***"PhotoContent"*** to represent contents existing in the website. Besides that, it has a list of ***"IObserver"*** which represents the current subscriptions to that website. It also implements the add, remove and notify methods coming from ***"IObservable"*** interface. In addition to those adding text content, adding audio content and adding photo content methods are added. In those methods the notify method is called with the content type so that only the subscriptions to that type get notified.

What if users or websites demand your software to support a fourth type of content? Will it be easy to modify?

If the users or websites demand the software to support a fourth type of content it will be easy to modify because only a single list of new content to the ***"Website"*** class and a new content class representing that content type needs to be added to the software.

Class Diagram



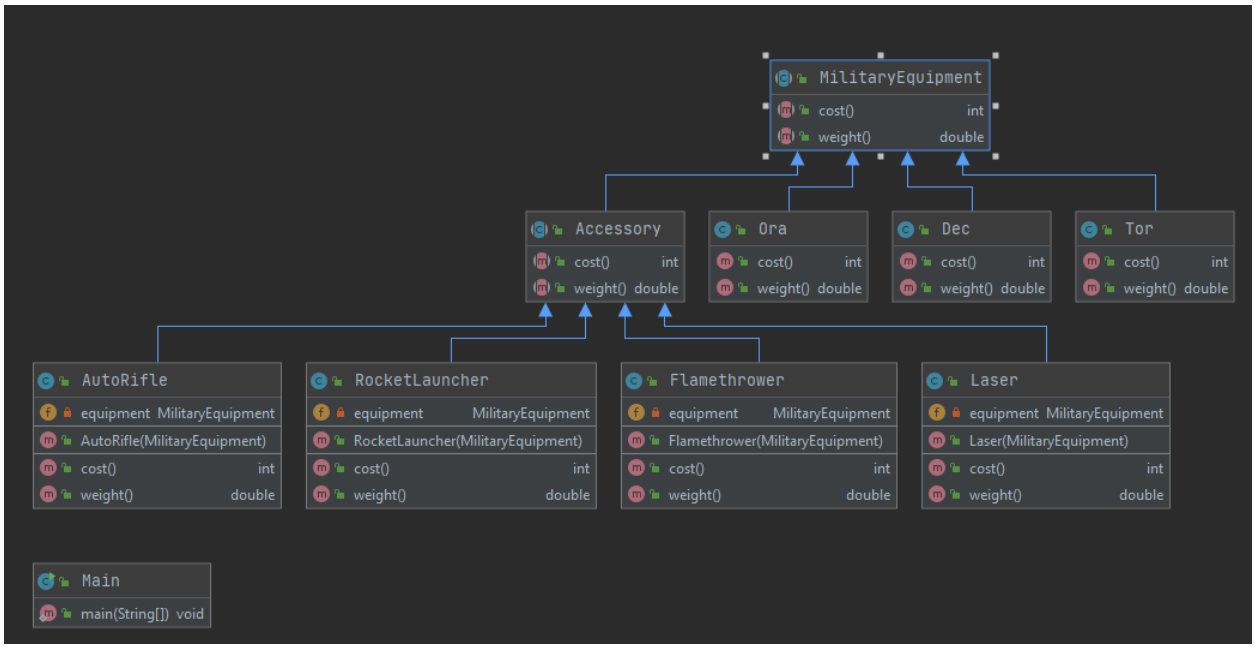
Question #3

What kind of design support to solve the problem?

In this question, decorator pattern is used to solve the problem. Because decorator pattern gives the ability to wrap objects around other objects creating more complex objects. Which is what is needed to solve this question. An abstract class called MilitaryEquipment is created which has cost and weight methods. From that class Dec, Ora and Tor suit's classes are extended which implement abstract methods of Military Equipment class which are cost and weight. Another abstract class which is called Accessory is extended from MilitaryEquipment class which represents the Flamethrower, Laser etc. accessories to the base suits. From

Accessory class Flamethrower,Laser, RocketLauncher and Autorifle classes are extended and they all implement the cost and weight methods of MilitaryEquipment as well. They contain a MilitaryEquipment object inside so they can add on their own weight and cost attributes to the base object they recieve which enables the wrapping of the object which was mentioned before.

Class Diagram



Additional Information

To execute the .jar files

"java -jar 171044079_CSE443_HW1_Q1 "

"java -jar 171044079_CSE443_HW1_Q3"

commands can be used.