22.11.2020

CSE 443 Object Oriented Analysis and Design

Homework 1 Report

Ahmet Yuşa Telli
151044092

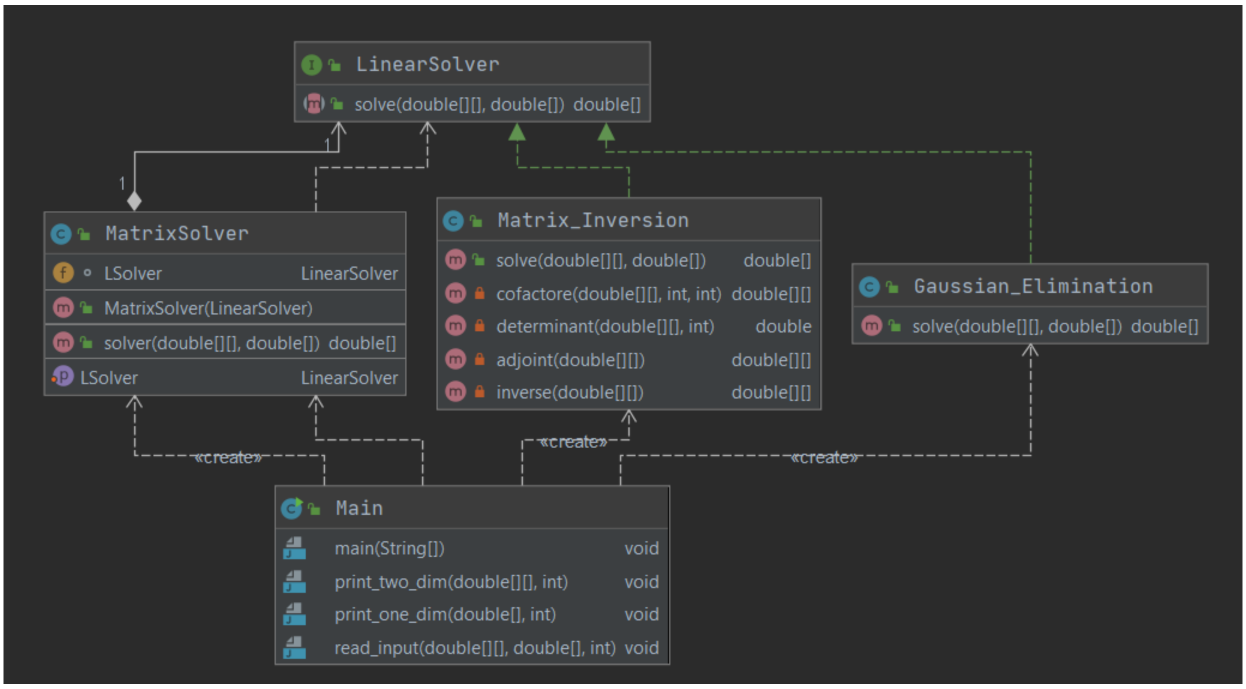
All parts were made with Java8. Jdk 1.8.0.271

Part 1:

For his part, Strategy Design Pattern has used for this solution. First, we have a LinearSolver interface which has a solve method. And we have two classes, which are implement from this interface, Gaussian_Elimination and Matrix_Inversion. We have also another class which name is MatrixSolver.

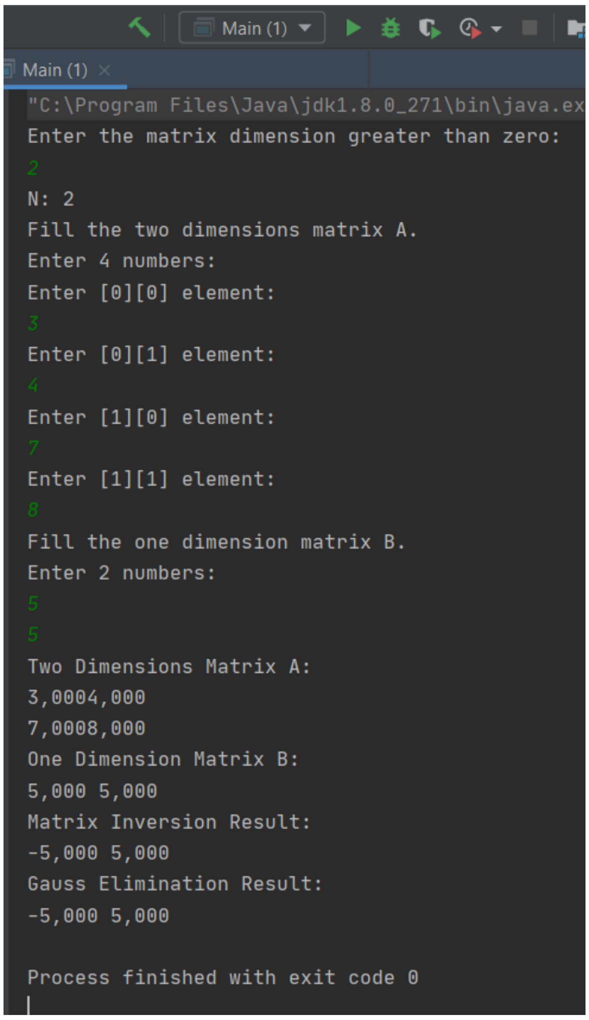
This MatrixSolver class decides which way to use for solving the input matrices. In Main method, if we call MatrixSolver with Gaussion_Elimination object, the MatrixSolver object calls Gaussian_Elimination's solve method. If we give Matrix_Inversion object to this MatrixSolver class, it calls the Matrix_Inversion's solve method.

Class Diagram:



LinearSolver
solve(double[][], double[]) double[]
MatrixSolver
LSolver LinearSolver
MatrixSolver(LinearSolver)
solver(double[][], double[]) double[]
LSolver LinearSolver
Matrix_Inversion
solve(double[][], double[]) double[]
cofactore(double[][], int, int) double[][]
determinant(double[][], int) double
adjoint(double[][]) double[][]
inverse(double[][]) double[][]
Gaussian_Elimination
solve(double[][], double[]) double[]
Main
main(String[]) void
print_two_dim(double[][], int) void
print_one_dim(double[], int) void
read_input(double[][], double[], int) void
«create»
«create»
«create»
1
1

Example Run:



```
"C:\Program Files\Java\jdk1.8.0_271\bin\java.ex
Enter the matrix dimension greater than zero:
2
N: 2
Fill the two dimensions matrix A.
Enter 4 numbers:
Enter [0][0] element:
3
Enter [0][1] element:
4
Enter [1][0] element:
7
Enter [1][1] element:
8
Fill the one dimension matrix B.
Enter 2 numbers:
5
5
Two Dimensions Matrix A:
3,0004,000
7,0008,000
One Dimension Matrix B:
5,000 5,000
Matrix Inversion Result:
-5,000 5,000
Gauss Elimination Result:
-5,000 5,000

Process finished with exit code 0
```

For run this part, in main method right click and click "Run Main.main()". Because while doing homework, I chanced Package name.

Part 2:

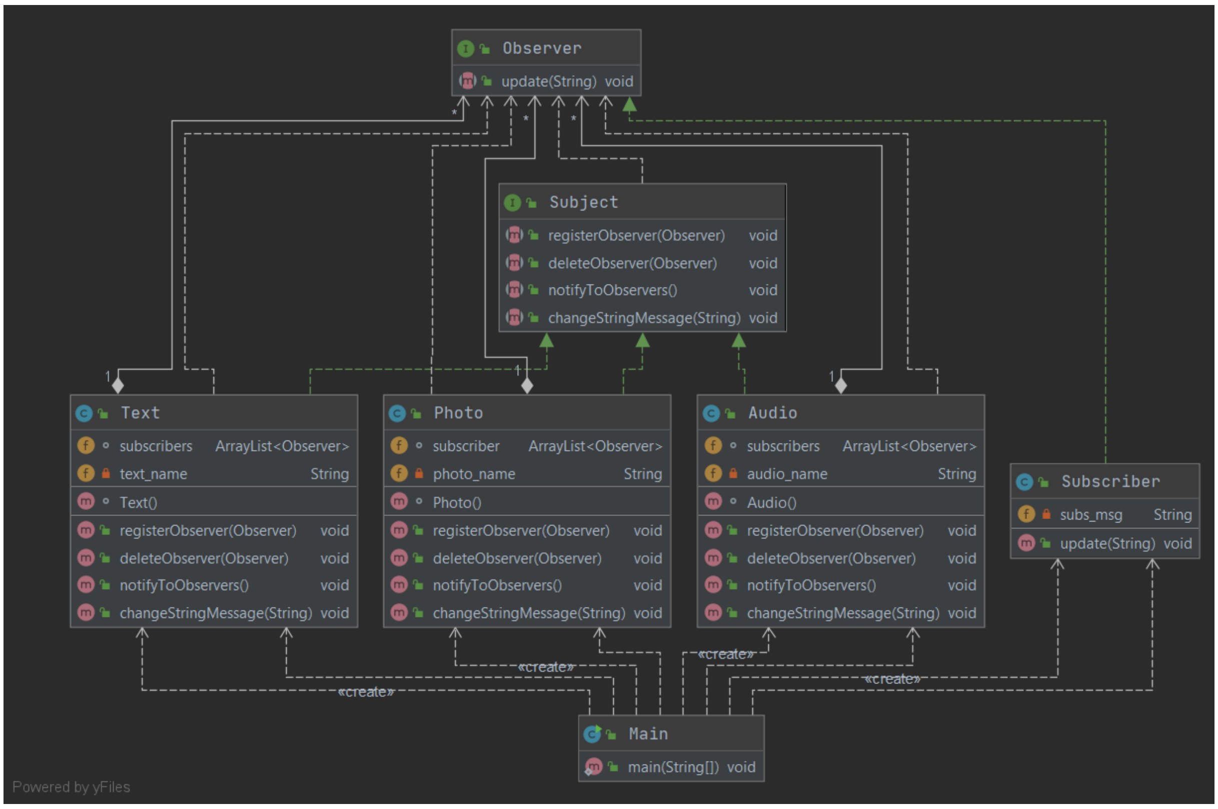
In the second part, we used Observer Design Pattern. We have three subjects and one observer. Content types (text, photograph, audio) are our subjects. These can register a subscriber, delete a subscriber, notify their own subscribers and change their own strings.

We have an interface which is Subject. In subject interface we have four methods: register a subscriber, delete subscriber, notify subscribers and change string. The classes which are implements from Subject, have overridden in their class.

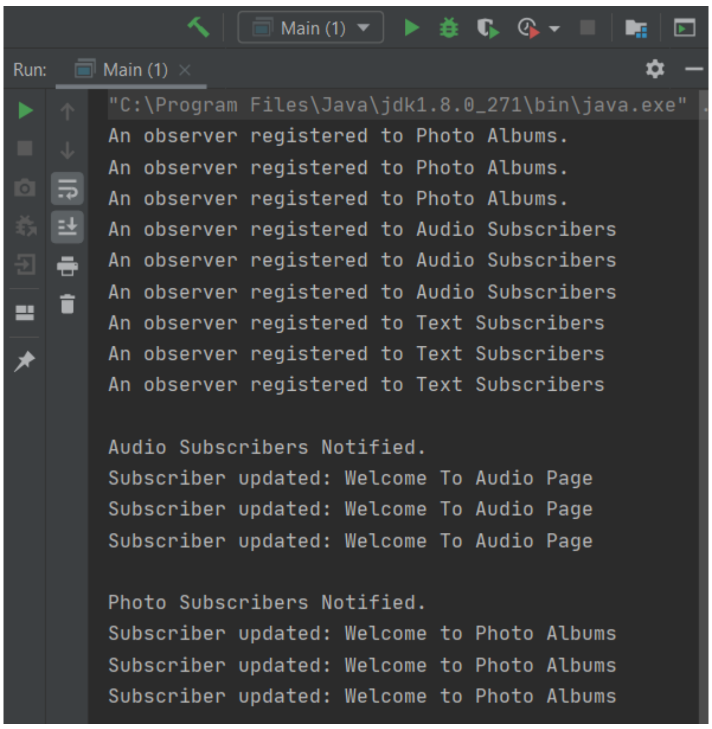
The subscriber is implemented from Observer interface. Observers have only update method and subscriber override this method.

In main, we have three different objects from text, photograph and audio classes. And we have five subscribers. The content types are register subscribers using their own register method. They chance their strings and notify their subscribers using their own methods.

Class Diagram:


Observer
update(String) void
Subject
registerObserver(Observer) void
deleteObserver(Observer) void
notifyToObservers() void
changeStringMessage(String) void
Text
subscribers ArrayList<Observer>
text_name String
Text()
registerObserver(Observer) void
deleteObserver(Observer) void
notifyToObservers() void
changeStringMessage(String) void
Photo
subscriber ArrayList<Observer>
photo_name String
Photo()
registerObserver(Observer) void
deleteObserver(Observer) void
notifyToObservers() void
changeStringMessage(String) void
Audio
subscribers ArrayList<Observer>
audio_name String
Audio()
registerObserver(Observer) void
deleteObserver(Observer) void
notifyToObservers() void
changeStringMessage(String) void
Subscriber
subs_msg String
update(String) void
Main
main(String[]) void
«create»
«create»
«create»
«create»
1
1
1
*
*
Powered by yFiles

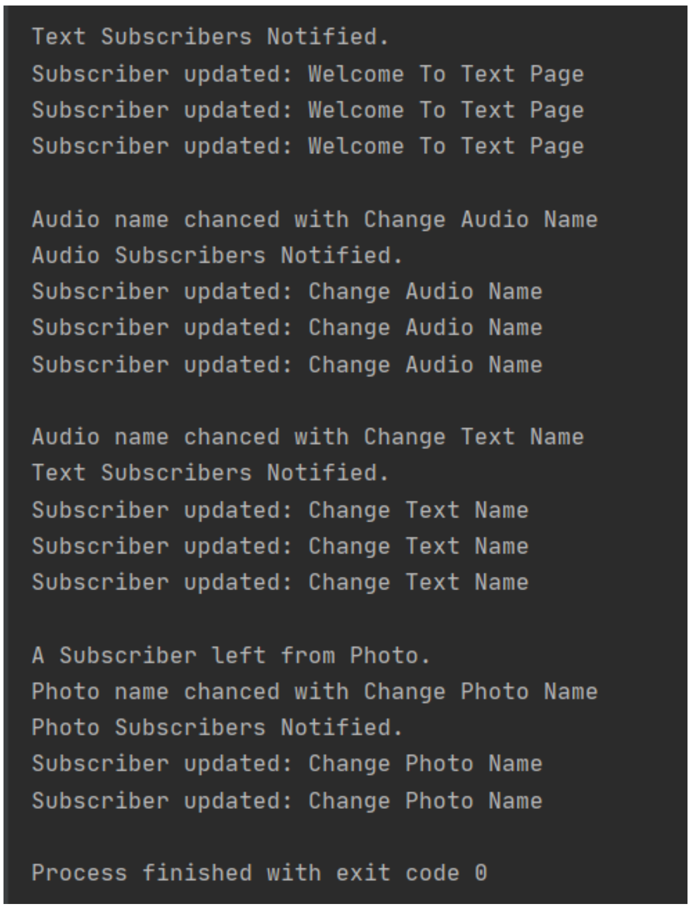
Example Run:



```
"C:\Program Files\Java\jdk1.8.0_271\bin\java.exe"
An observer registered to Photo Albums.
An observer registered to Photo Albums.
An observer registered to Photo Albums.
An observer registered to Audio Subscribers
An observer registered to Audio Subscribers
An observer registered to Audio Subscribers
An observer registered to Text Subscribers
An observer registered to Text Subscribers
An observer registered to Text Subscribers

Audio Subscribers Notified.
Subscriber updated: Welcome To Audio Page
Subscriber updated: Welcome To Audio Page
Subscriber updated: Welcome To Audio Page

Photo Subscribers Notified.
Subscriber updated: Welcome to Photo Albums
Subscriber updated: Welcome to Photo Albums
Subscriber updated: Welcome to Photo Albums
```



```
Text Subscribers Notified.
Subscriber updated: Welcome To Text Page
Subscriber updated: Welcome To Text Page
Subscriber updated: Welcome To Text Page

Audio name chanced with Change Audio Name
Audio Subscribers Notified.
Subscriber updated: Change Audio Name
Subscriber updated: Change Audio Name
Subscriber updated: Change Audio Name

Audio name chanced with Change Text Name
Text Subscribers Notified.
Subscriber updated: Change Text Name
Subscriber updated: Change Text Name
Subscriber updated: Change Text Name

A Subscriber left from Photo.
Photo name chanced with Change Photo Name
Photo Subscribers Notified.
Subscriber updated: Change Photo Name
Subscriber updated: Change Photo Name

Process finished with exit code 0
```

For run this part, in main method right click and click "Run Main.main()". Because while doing homework, I chanced Package name.

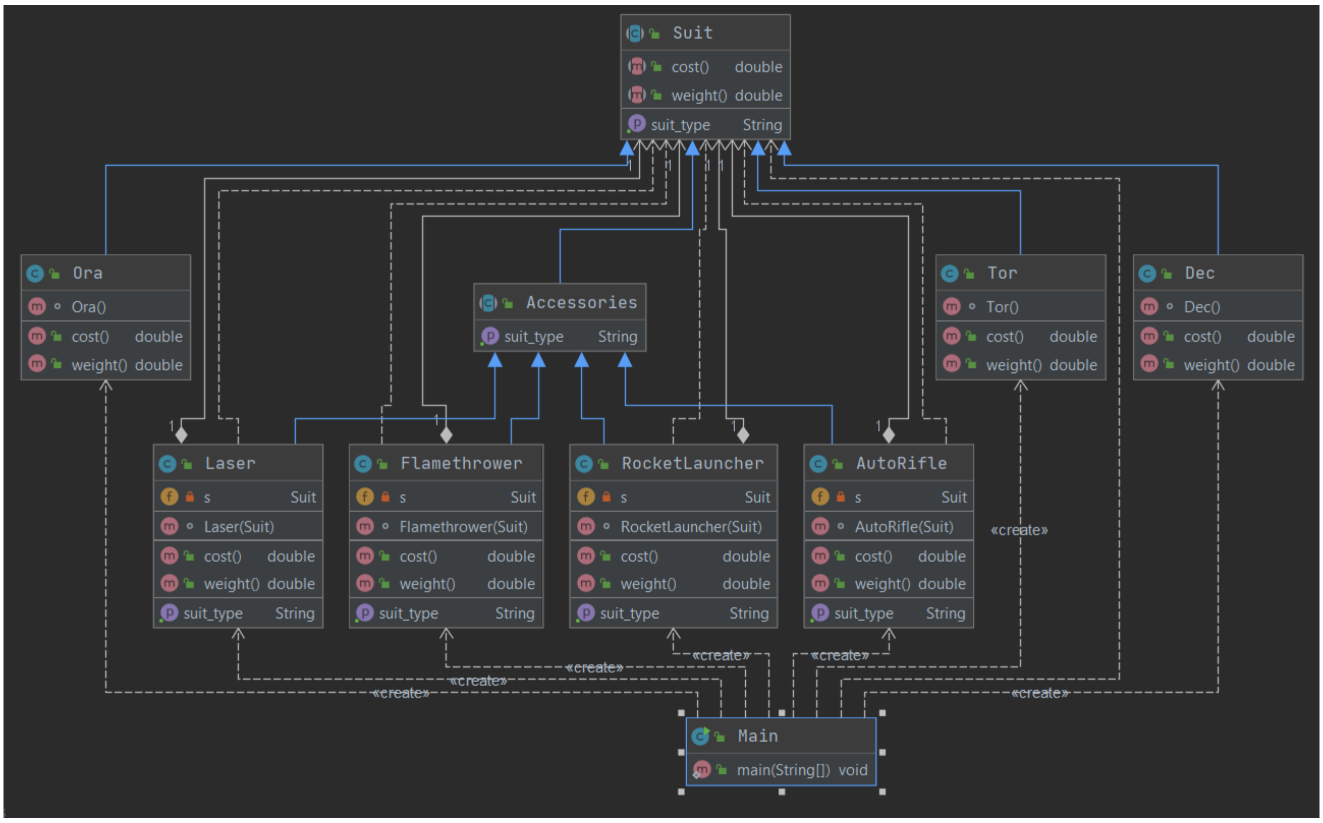
Part 3:

For the solution, we used Decorator Design Pattern. The basic of suits must be the same Dec, Ora, Tor. Therefore, we have an abstract class which name is Suit. These basic classes extend from this Suit class. These three classes overrides cost(), weight() methods from Suit class.

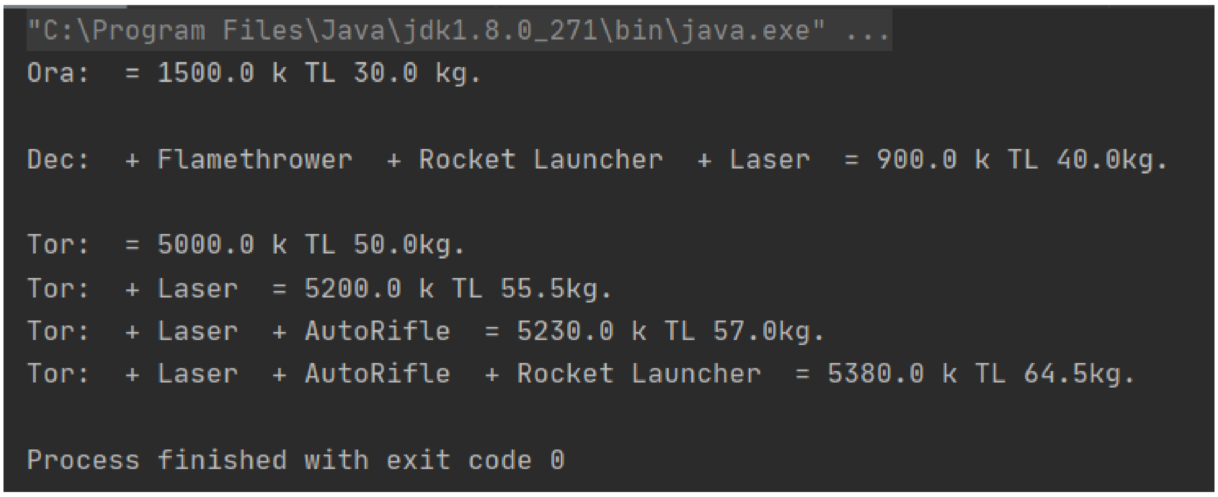
We have four different types of accessories. We create an abstract class which name is Accessories and these types are extended from Accessories class. This class must be abstract and it abstracts getSuit_type() method of Suit. This class is decorator of Suit class. These types of accessories classes (AutoRifle, RocketLauncher, Flamethrower, Laser) must override cost(), weight(), getSuit_type() methods from Suit class. These classes are also composition relation with Suit class.

In main method, we create an object which real type is Ora class but declared type is Suit class. Then, we call these objects methods and it calls Ora's methods. Second, we create an object which real type is Dec class but declared type is Suit class. And we add some accessories and call Flamethrower, RocketLauncer and Laser. When we call these objects methods, it comes to us by adding to their costs.

Class Diagram:



Example Run:



```
"C:\Program Files\Java\jdk1.8.0_271\bin\java.exe" ...
Ora:  = 1500.0 k TL 30.0 kg.

Dec:  + Flamethrower  + Rocket Launcher  + Laser  = 900.0 k TL 40.0kg.

Tor:  = 5000.0 k TL 50.0kg.
Tor:  + Laser  = 5200.0 k TL 55.5kg.
Tor:  + Laser  + AutoRifle  = 5230.0 k TL 57.0kg.
Tor:  + Laser  + AutoRifle  + Rocket Launcher  = 5380.0 k TL 64.5kg.

Process finished with exit code 0
```

Ahmet Yuşa Telli

151044092