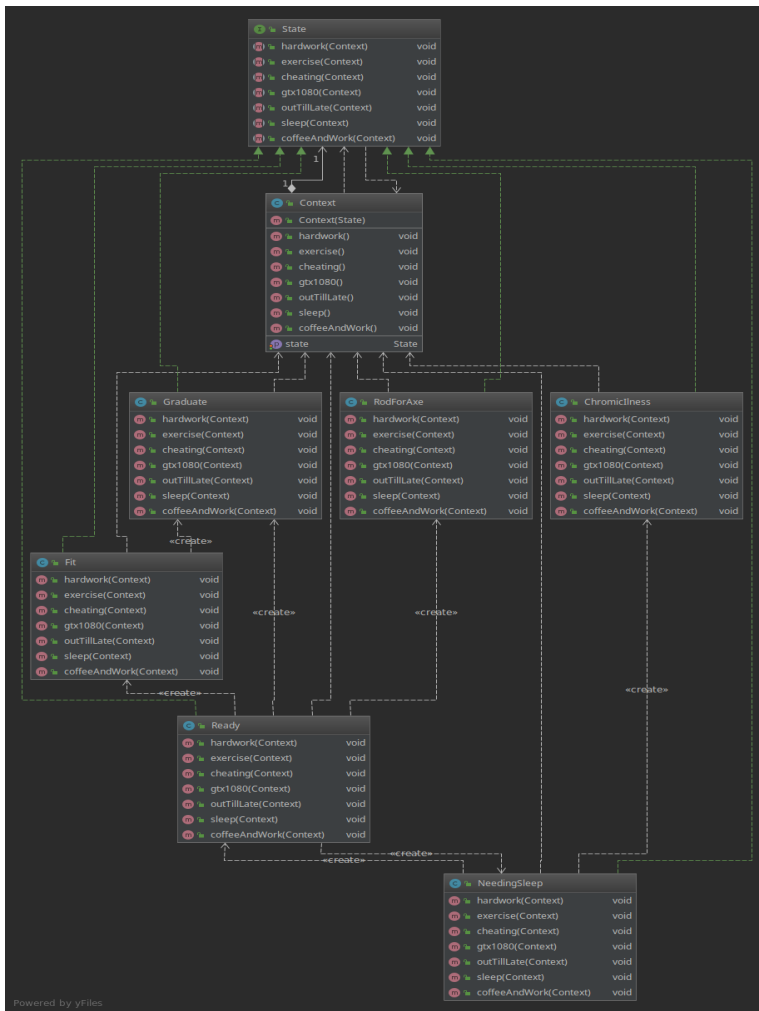


CSE443  
OBJECT ORIENTED ANALYSIS AND  
DESIGN  
HW4  
REPORT

HAKKI ERDEM DUMAN  
151044005

## Question 1



In this part, state design pattern is used. All the states are generated as classes and these classes implement “State” interface’s methods.

These methods are the connections between states.

Each of these methods are implemented in state classes in different ways. For example, if there is no way to go from a state, that state’s methods are implemented with a message like “You can’t do that”. Otherwise, if a state has a way like a method, which will be implemented, that method is going to involve a state change.

In this output, there are 4 tests.

First test is the way of “Ready → Fit → Graduate” and that’s why **exercise** and **hardwork** methods are called.

Second test is the way of “Ready → Needing Sleep → Chronic Illness” **outTillLate** and **coffeAndWork** methods are called for it.

Third test is the way of “Ready → Unable to become a rod for an axe” **gtx1080** method is called here.

And the fourth test is a wrong case. The way was “Ready → Sleep” but there is no connection between these two states. That’s why an error message is printed on the screen.

```
Run: Main x
/usr/lib/jvm/java-8-openjdk-amd64/bin/java ...
Test 1
Fit
Graduate

Test 2
Needing Sleep
Chronic Illness

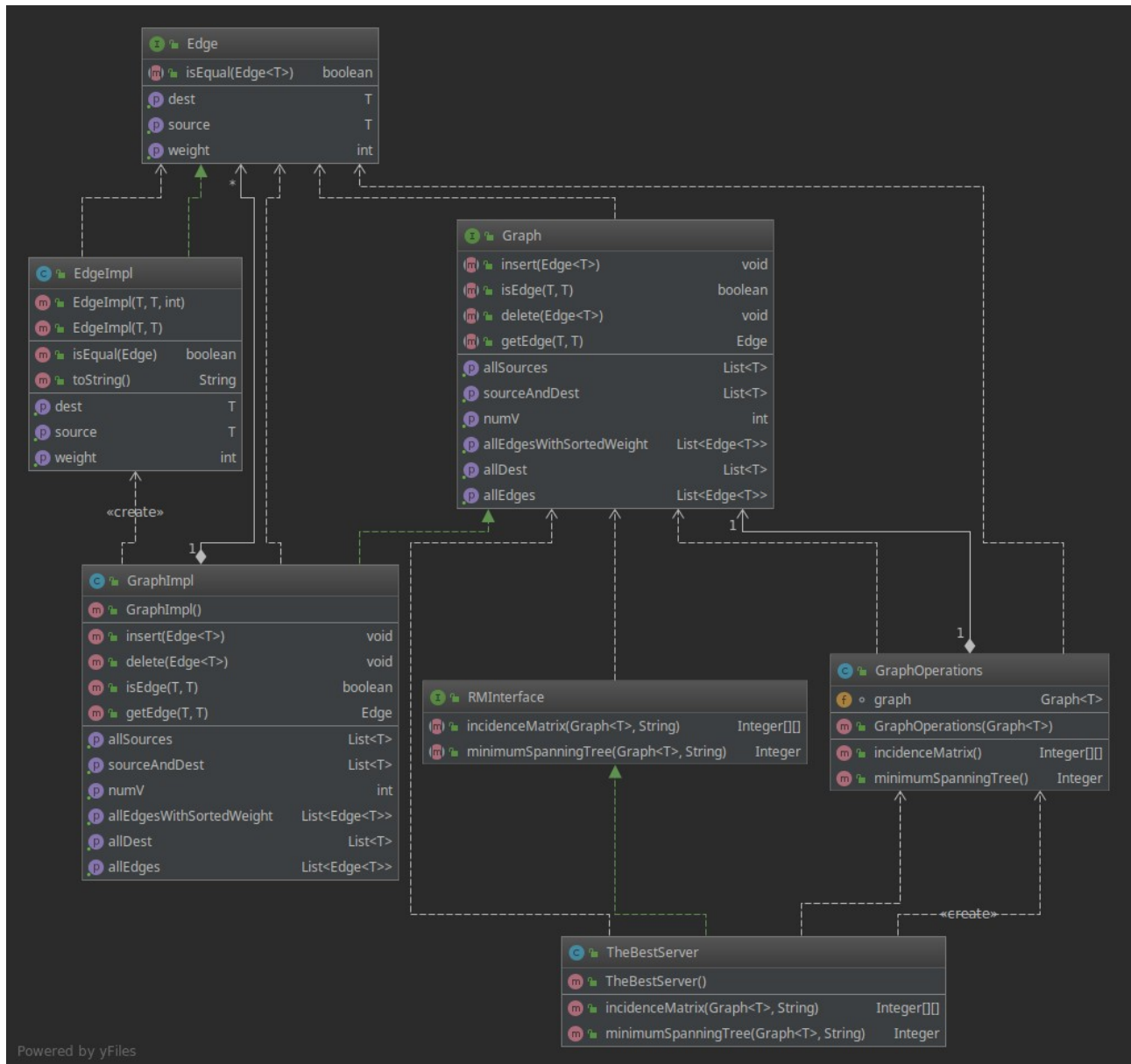
Test 3
Unable to become a rod for an axe

Test 4
You can't do that

Process finished with exit code 0
```

## Question 2

This question is separated into two parts, which are called server and client. In server part, a server is created with Java RMI. This server waits for a client to connect, generate a graph and send a job request to it. These job requests can be either minimum spanning tree of the given graph or incidence matrix of it. After getting graph information and job request from client, an answer is sent to, according to its request. Server's class diagram can be seen below.

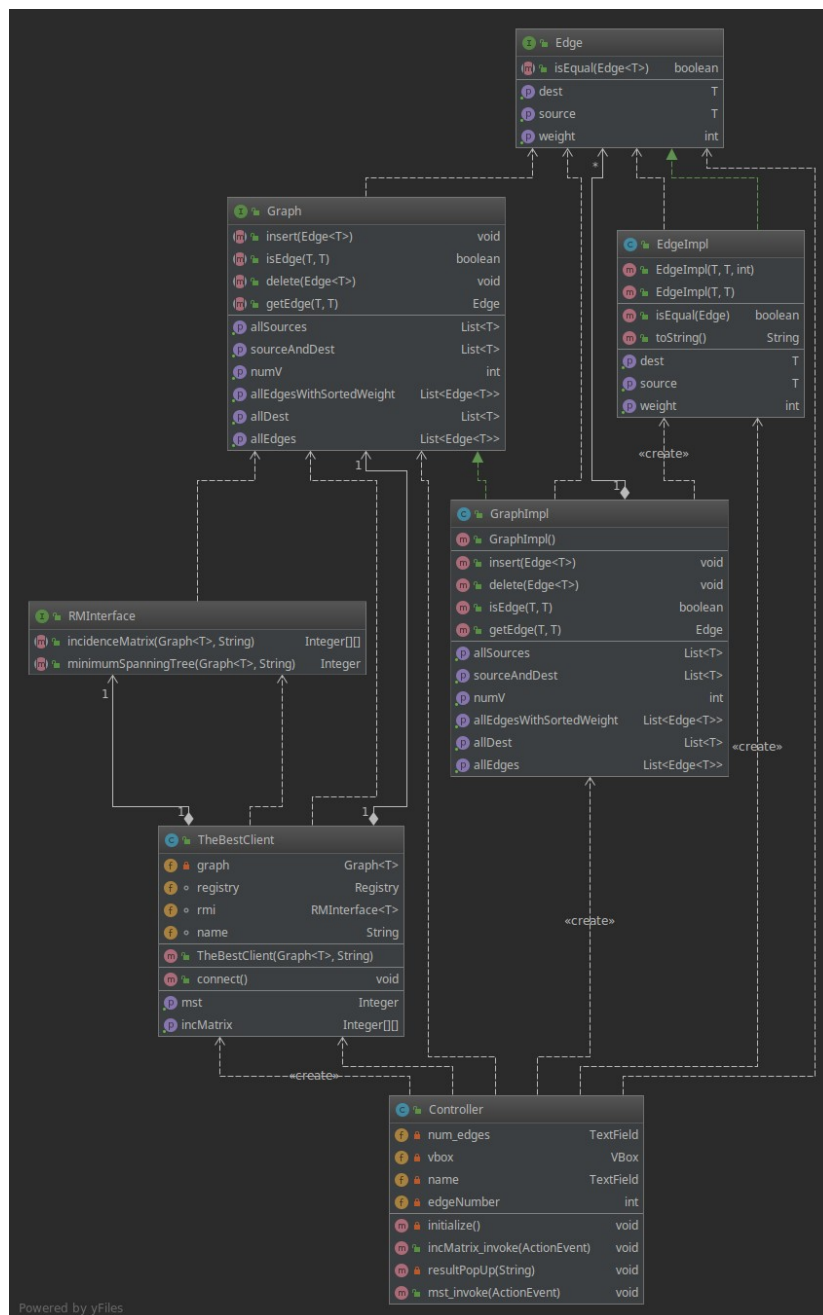


As I told above, client sends its graph information and method to server with using a graphical user interface. In GUI; name of the user, edge number and edge information must be entered and method type must be clicked after these. Edge structure must be something like:

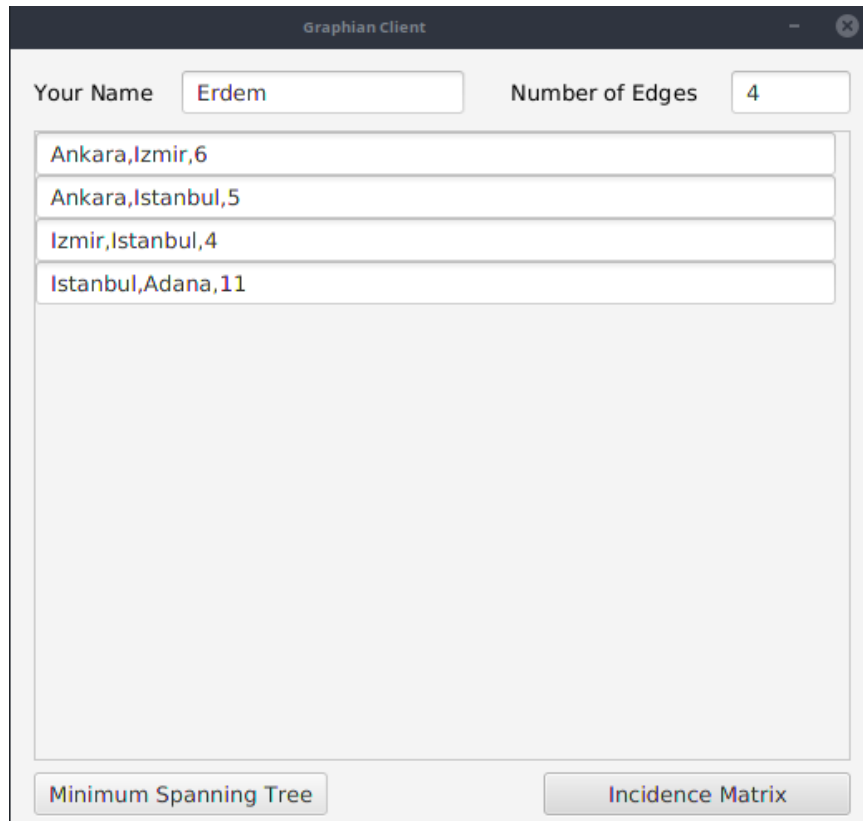
Istanbul,Adana,11  
Izmir,Istanbul,5

shown above. There should be commas between source, destination and weight.

Class diagram of client side is here:

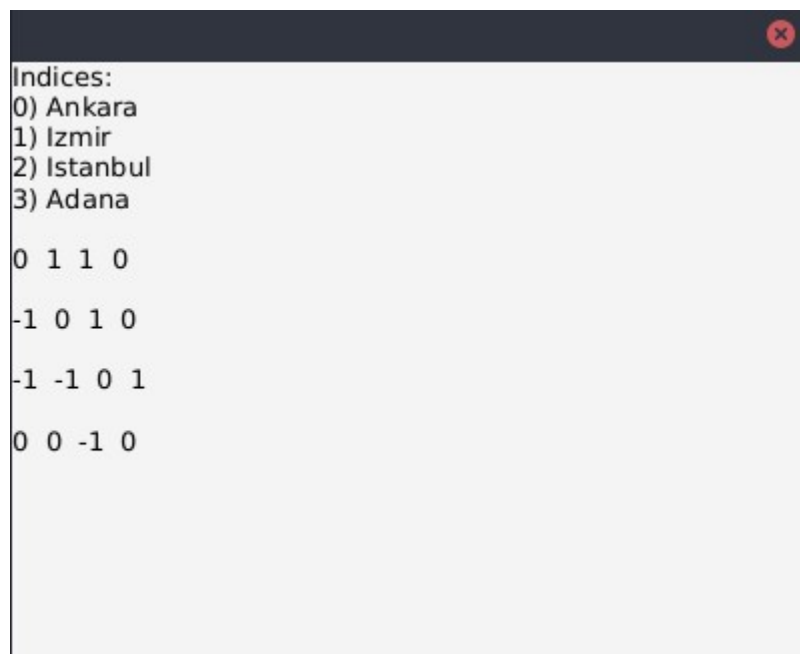


Screenshots of server and client:



The screenshot shows the Graphian Client application window. At the top, the title bar reads "Graphian Client". Below the title bar, there are two input fields: "Your Name" with the value "Erdem" and "Number of Edges" with the value "4". Below these fields is a list of four entries, each in a separate box: "Ankara,Izmir,6", "Ankara,Istanbul,5", "Izmir,Istanbul,4", and "Istanbul,Adana,11". At the bottom of the window, there are two buttons: "Minimum Spanning Tree" and "Incidence Matrix".

*GUI of Client*



The screenshot shows the output of the Incidence Matrix. It displays the indices of the nodes and the corresponding incidence matrix.

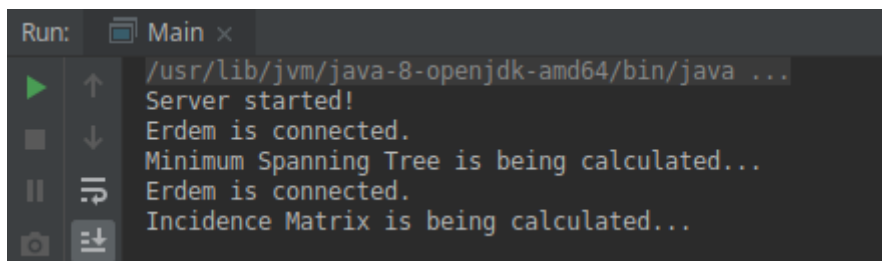
Indices:  
0) Ankara  
1) Izmir  
2) Istanbul  
3) Adana

0	1	1	0
-1	0	1	0
-1	-1	0	1
0	0	-1	0

*Output of Incidence Matrix*



*Output of Minimum Spanning Tree*



*Server side of the program*

**NOTE 1:** Even though there are two messages of “Erdem is connected”, client connects only once. I want to indicate that I can get name from client.

**NOTE 2:** Minimum Spanning Tree is calculated with Kruskal’s Algorithm. That means I consider these edges as undirected for this problem. In incidence matrix problem, edges are handled as directed, which is the expected scenario.