

# **CSE443 Object Oriented Analysis and Design**

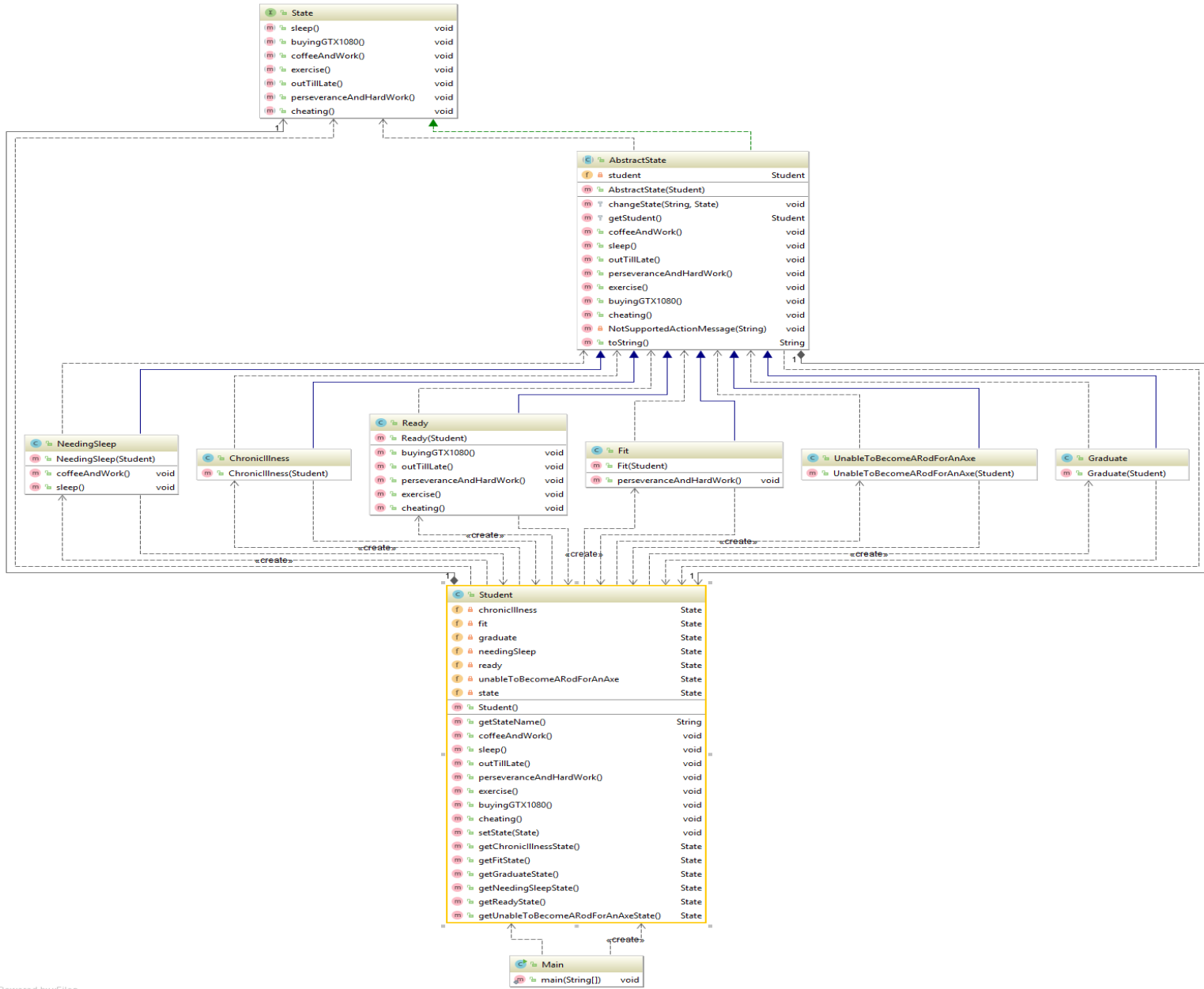
**Seyit Ahmet KARACA - 141044084**

## **HW4 Rapor**

1.Part 1 – State Design Pattern .....	2
1.1.Part1 UML Diyagramı .....	2
1.2.UML Diagram Açıklaması.....	2
1.3.State Design Pattern Sonuçları.....	3
2. Part2 - Proxy Design Pattern .....	4
2.1. Part2 - Proxy UML Diyagramı .....	4
2.2. UML Diagram Açıklaması.....	4
2.3. Part2 – Proxy Design pattern Çıktıları ve notlar .....	5

## 1.Part 1 – State Design Pattern

### 1.1.Part1 UML Diyagramı



### 1.2.UML Diagram Açıklaması

State'lerin bulunduğu bir interface oluşturuldu. Bu state'leri implement edip hepsine hata mesajı bastırılan ve stateleri değiştirilmesini sağlayan bir soyut sınıf oluşturuldu. Soyut sınıf içerisinde bir tane öğrenci sınıfı değişkeni tutulup bu öğrenci sınıfında ise öğrencinin yapabileceği aktiviteler tanımlandı ve aktivitelere göre state'i güncellendi. Her durum için soyut sınıftan türetilmiş şekilde sınıf oluşturuldu ve bu durum sınıflarından hangi durumlara geçilebiliyor ise ilgili durumlar override edildi. Her durum sınıfında bulunduğu durumdan geçebileceği duruma ait olan method çağrıldığında durum güncellemesi yapıldı.

### 1.3.State Design Pattern Sonuçları

---ready to unable to become a rod for an axe-----  
State ChronicIllness is not supporting action buying GTX1080  
Current state : Ready + Cheating  
New state : UnableToBecomeARodForAnAxe

---ready to needing sleep to ready-----  
Current state : Ready + Out till late  
New state : NeedingSleep

Current state : NeedingSleep + SLEEP  
New state : Ready

---Errors-----  
State Ready is not supporting action coffee & work  
Current state : Ready + Out till late  
New state : NeedingSleep

State NeedingSleep is not supporting action cheating  
State NeedingSleep is not supporting action perseverance and hard work  
State NeedingSleep is not supporting action buying GTX1080  
State NeedingSleep is not supporting action exercise

---ready to fit to graduate-----  
Current state : Ready + Exercise  
New state : Fit

Current state : Fit + Perseverance and hard work  
New state : Graduate

---ready to graduate-----  
Current state : Ready + Perseverance and & work  
New state : Graduate

---ready to needing sleep -> chronic illness-----  
Current state : Ready + Out till late  
New state : NeedingSleep

Current state : NeedingSleep + Coffee & work  
New state : ChronicIllness

```

classDiagram
    class Edge {
        +int from
        +int to
        +double weight
    }
    class Graph {
        +int v
        +Edge[] edges
    }
    class Account {
        +String name
        +String creditCard
    }
    class RegistrationService {
        +Map accounts
    }
    class BulutCizgeRegistrationService {
        +Map accounts
    }
    class ClientGUI {
        +JTextField usernameField
        +JTextField cardNumberTextField
        +JButton registerButton
        +JTextArea responseArea
        +JTextField pathField
        +JButton constructGraphButton
        +JButton minimumSpanningTreeButton
        +JButton incidenceMatrixButton
        +JPanel card
        +JTextField creditField
        +JButton loadCreditButton
        +JButton showGraphButton
        +JButton GetCreditButton
        +JTextField userCredit
        +JButton listAccountsButton
        +JComboBox pathFieldComboBox
        +JComboBox graphService
        +JComboBox registrationService
        +JComboBox graph
    }
    class ServiceInvocationHandler {
        +GraphService service
        +BulutCizgeRegistrationService bulutCizgeRegistrationService
    }
    class Server {
    }

    Edge "1" -- "*" Graph
    Graph "1" -- "1" GraphService
    Account "1" -- "1" RegistrationService
    RegistrationService "1" -- "1" BulutCizgeRegistrationService
    ClientGUI "1" -- "1" ServiceInvocationHandler
    ServiceInvocationHandler "1" -- "1" Server
    ServiceInvocationHandler "1" -- "1" GraphService
    ServiceInvocationHandler "1" -- "1" BulutCizgeRegistrationService
    
```

The diagram illustrates the following classes and their relationships:

- Edge**: Attributes include `from` (int), `to` (int), and `weight` (double). Methods include `Edge()` and `Edge(int, int, double)`.
- Graph**: Attributes include `v` (int) and `edges` (Edge[]). Methods include `graphToMatrix()` and `graphToMatrix(String)`.
- Account**: Attributes include `name` (String) and `creditCard` (String). Methods include `Account(String, String)`, `equals(Object)`, `hashCode()`, and `creditCard`.
- RegistrationService**: Attribute includes `accounts` (Map). Methods include `register(Account, int)`, `loadCredit(Account, int)`, and `getCredit(Account)`.
- BulutCizgeRegistrationService**: Attribute includes `accounts` (Map). Methods include `getCredit(Account)`, `hasCredit(Account)`, `decreaseCredit(Account)`, `setCredit(Account, int)`, `register(Account, int)`, and `loadCredit(Account, int)`.
- ClientGUI**: Attributes include `usernameField`, `cardNumberTextField`, `registerButton`, `responseArea`, `pathField`, `constructGraphButton`, `minimumSpanningTreeButton`, `incidenceMatrixButton`, `card`, `creditField`, `loadCreditButton`, `showGraphButton`, `GetCreditButton`, `userCredit`, `listAccountsButton`, `pathFieldComboBox`, `graphService`, `registrationService`, and `graph`. Methods include `ClientGUI()`, `getCreditMethod()`, `finder(String)`, and `main(String[])`.
- ServiceInvocationHandler**: Attributes include `service` (GraphService) and `bulutCizgeRegistrationService` (BulutCizgeRegistrationService). Methods include `ServiceInvocationHandler(GraphService, BulutCizgeRegistrationService)` and `invoke(Object, Method, Object[])`.
- Server**: Method includes `main(String[])`.

Relationships are shown with dashed arrows and multiplicity:

- Edge** (1) to **Graph** (\*): Association.
- Graph** (1) to **GraphService** (1): Association.
- Account** (1) to **RegistrationService** (1): Association.
- RegistrationService** (1) to **BulutCizgeRegistrationService** (1): Association.
- ClientGUI** (1) to **ServiceInvocationHandler** (1): Association.
- ServiceInvocationHandler** (1) to **Server** (1): Association.
- ServiceInvocationHandler** (1) to **GraphService** (1): Association.
- ServiceInvocationHandler** (1) to **BulutCizgeRegistrationService** (1): Association.

Generalization relationships are indicated by solid arrows with hollow triangle heads:

- GraphService** is a generalization of **Graph**.
- BulutCizgeRegistrationService** is a generalization of **RegistrationService**.

Creation relationships are indicated by dashed arrows with open circle heads and the label `<<create>>`:

- GraphService** creates **Graph**.
- RegistrationService** creates **Account**.
- BulutCizgeRegistrationService** creates **Account**.
- ServiceInvocationHandler** creates **GraphService**.
- ServiceInvocationHandler** creates **BulutCizgeRegistrationService**.
- Server** creates **ServiceInvocationHandler**.

Client tarafında ise grafik arayüzü bulunmaktadır. Grafik arayüzü başlatılmadan önce Naming.lookup ile servisler tanıtılmış ve gerekli butonlara servislerle haberleşmesi için kodlar yazılmıştır.

Programın sağlık çalışabilmesi için client.jar ile aynı klasörde bulunması gereken graph'ların bulunduğu text dosyaları bulunmalıdır. Text dosyaların içeriği ilk satırda vertex sayısı , alt satırlarda ise from(int) to(int) weight(int) şeklinde satırlardan oluşması gerekmektedir.

Client

User Name

Seyit

1

Credit Card No

123

2

Register

List Accounts

Credit

3

Get Credit

100

4

Load credit

5

Build Graph

Show Graph

Minimum Spanning Tree

Incidence Matrix

Account registered

3 numaralı alan “Get Credit” butonuna veya herhangi bir servis kullanıldığında kalan bakiyeyi göstermektedir. 4 numaraları alanda bulunan sayı “Load Credit” butonuna basılarak kalan bakiyeye 4

numaralı alanda yazan miktar kadar yükleme yapılır. 5 ile işaretlenmiş alanın kullanım şekli ise , aşağı ok işaretinden önce 5 numaralı alana bir kez tıklanarak client.jar ile aynı klasörde bulunan text dosyalarını tarar ve bu combobox doldurulur. Sağındaki ok'a basılarak istenen içerisinde graph olan dosya seçilip build graph , show graph , minimum spanning tree , incidence matrix servisleri kullanılabilir. Her servis 2 kredi ile kullanılmaktadır ve başlangıçta her kullanıcıya 20 kredi verilmektedir. List Account servisi ise servera kayıtlı kullanıcıları alttaki panelde listelemektedir ve kullanıcıya bağlı bir servis olmadığından herhangi bir kredi ücretine tabii tutulmamıştır.

Çıktılar:

The screenshot shows the 'Client' application window. The 'User Name' field contains 'Seyit' and the 'Credit Card No' field contains '123'. The 'Credit' field shows '0'. The 'Get Credit' button is disabled. The 'Build Graph' button is active. The 'Show Graph' button is disabled. The 'Minimum Spanning Tree' button is disabled. The 'Incidence Matrix' button is disabled. The 'List Accounts' button is disabled. The 'Load credit' button is disabled. The 'Register' button is disabled. The 'Incidence Matrix' button is disabled. The message 'Account couldn't registered' is displayed in the main area.

Sisteme kayıt olan bir kullanıcı tekrar kayıt olamaz mesajının bulunduğu görsel

The screenshot shows the 'Client' application window. The 'User Name' field contains 'Ali ata bak' and the 'Credit Card No' field contains '852'. The 'Credit' field shows '0'. The 'Get Credit' button is disabled. The 'Build Graph' button is active. The 'Show Graph' button is disabled. The 'Minimum Spanning Tree' button is disabled. The 'Incidence Matrix' button is disabled. The 'List Accounts' button is disabled. The 'Load credit' button is disabled. The 'Register' button is disabled. The message 'User already registered' is displayed in the main area.

Yetersiz bakiye bilgisinin bulunduğu görsel.

The screenshot shows the 'Client' application window. The 'User Name' field contains 'Seyit' and the 'Credit Card No' field contains '123'. The 'Credit' field shows '0'. The 'Get Credit' button is disabled. The 'Build Graph' button is active. The 'Show Graph' button is disabled. The 'Minimum Spanning Tree' button is disabled. The 'Incidence Matrix' button is disabled. The 'List Accounts' button is disabled. The 'Load credit' button is disabled. The 'Register' button is disabled. The message 'User don't have enough credit to build graph.' is displayed in the main area.

Sistemdeki kayıtlı kullanıcıların bulunduğu görsel.

```
graphToString invoked at 07:29:02 (Account: qwe) (Credit left: 18) (Thread 19)
buildGraph invoked at 07:29:17 (Account: qwe) (Credit left: 16) (Thread 19)
buildGraph tooks 0,011 ms (Account: qwe) (Thread 19)
-----
```

```
graphToString invoked at 07:29:19 (Account: qwe) (Credit left: 14) (Thread 19)
graphToString tooks 0,326 ms (Account: qwe) (Thread 19)
-----
```

```
getMinimumSpanningTree invoked at 07:29:21 (Account: qwe) (Credit left: 12) (Thread 19)
getMinimumSpanningTree tooks 0,944 ms (Account: qwe) (Thread 19)
-----
```

Server’da hangi servisin hangi kullanıcı tarafından saat kaçta kullanıldığı ve ne kadar sürdüğü bilgisinin çıktısıdır.

Not :Derste gösterdiğiniz konsoldan “start rmiregistry” komutu server sınıfı içerisine gömülmüştür. Bu yüzden kontrolden “start rmiregister” komutu yapılmasına gerek yoktur. Executable dosyaları ödev klasörü içerisinde executablePart1 ve executablePart2 dosyalarında bulunmaktadır. Görsellerin orjinalleri ödev klasöründe bulunmaktadır. odevPart1 ve odevPart2 ‘ de ise proje kodları bulunmaktadır.

Not2: Bu ödevin kodlarında bazı yerlerde odev5 görebilirsiniz. Ödeve başladığımda bu ödevi 5.ödev sanıyordum o yüzden öyle bir yanlışlık yaptım. Kodlar düzgün çalışıyor, sadece bir isim karışıklığı oluştu.

Not 3 : Çalıştığım makinedeki java –version çıktısı;

```
java version "1.8.0_191"
```

```
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)
```

```
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)
```