

**Gebze Technical University
Computer Engineering**

**Object Oriented Analysis and Design
CSE443 – 2018 Autumn**

HOMEWORK 4 REPORT

**Yunus ÇEVİK
141044080**

Course Teacher: Erchan Aptoula

Course Assistant: Muhammet Ali Dede

Answer – 1

1 METHOD

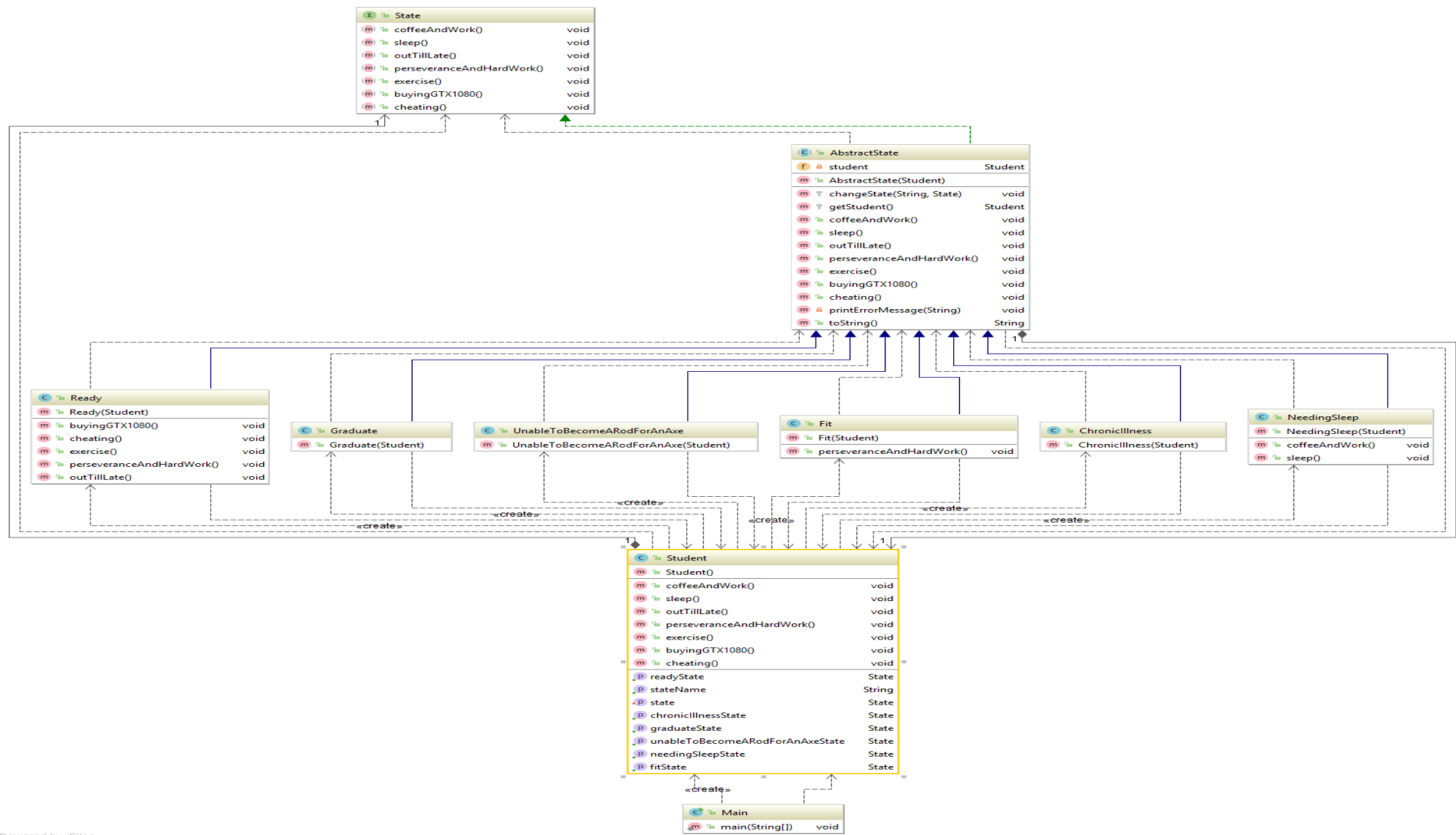
State projesini tasarlariken State Tasarım Desenini kullanarak tasarladım. Durum(State) tasarım deseni, behavioral tasarım desenlerinden biridir. Nesnenin durumu değiştiğinde, davranışının değişmesine imkan tanır. Nesne farklı durumlarda, farklı davranışlar gösteriyorsa bu tasarım desenini kullanabiliriz. Farklı durumlara sahip nesnenin, yeni bir duruma geçtiğinde davranışının yani yaptığı işlemin değişmesidir.

State tasarım deseni için, öncelikle bir State interface'i oluşturulur. Daha sonra oluşturulan interface'i implement edecek somut State sınıfları yaratılır. Genelde somut State sınıfları Context sınıfı türünden constructor'a sahip olurlar. Context sınıfı yaratılır. Bu sınıf içerisinde state interface türünden durumların set edilmesi için bir metod bulunur. Bu metod sayesinde Context sınıfı state nesnesini tutmuş olur. Son olarak bir Client(Main) sınıfı oluşturulur. Bu sınıf Context sınıfından ve state sınıflardan nesneler üreterek işlemlerin yapılmasını sağlar.

State tasarım deseninin faydası:

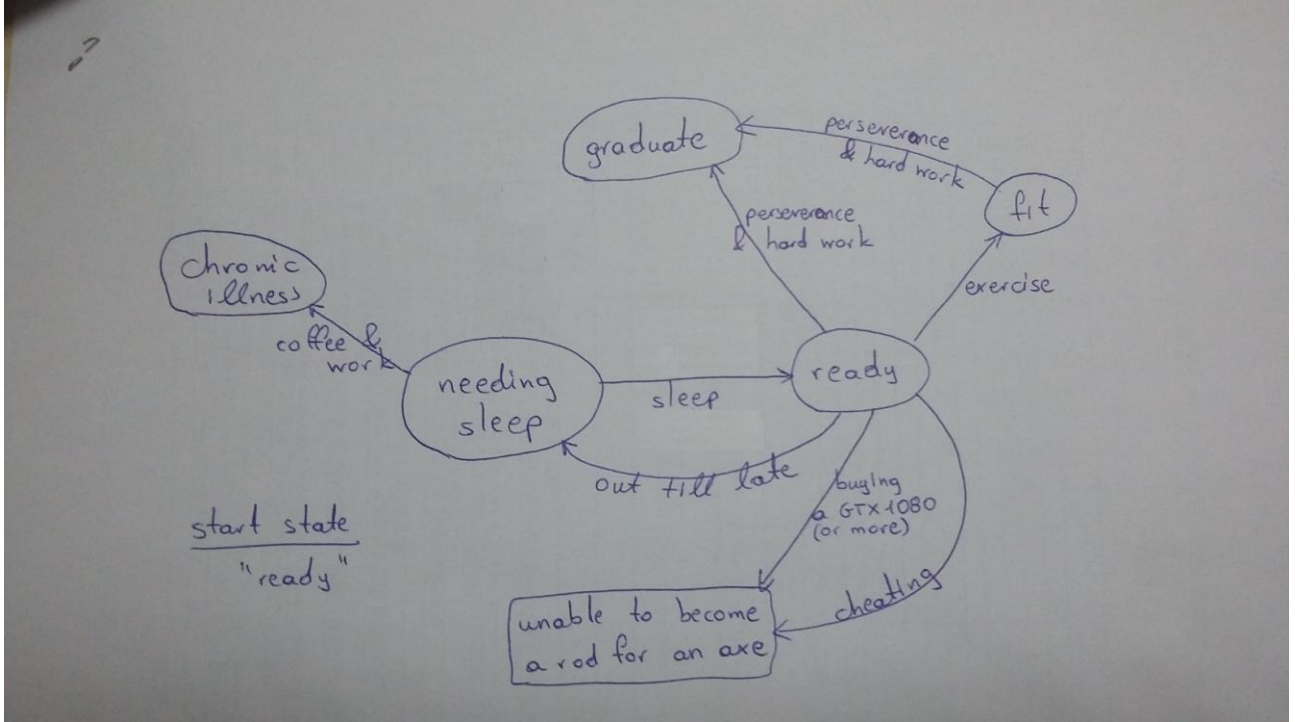
Nesnelerin duruma bağlı davranışlarının karmaşık if-else veya switch ifadeleri ile kontrol edilmesini önler.

Class Diagrams



2 Result

State Tasarım Deseni ile aşağıda belirtilen finite state machine resminde yer alan durumun ekran çıktısı aşağıda yer almaktadır.



OUTPUT:

```
*****
* Test: ready -> unable to become a rod for an axe *
*****
Current state: Ready
Action process: Buying GTX1080
New Target state: UnableToBecomeARodForAnAxe

Current state: Ready
Action process: Cheating
New Target state: UnableToBecomeARodForAnAxe

----- END STATE -----

*****
* Test: ready -> fit -> graduate *
*****
Current state: Ready
Action process: Exercise
New Target state: Fit

Current state: Fit
Action process: Perseverance and hard work
New Target state: Graduate

----- END STATE -----

*****
* Test: ready -> graduate *
*****
Current state: Ready
Action process: Perseverance and hard work
New Target state: Graduate

----- END STATE -----
```

```
*****
* Test: ready -> needing sleep -> chronic illness *
*****
Current state: Ready
Action process: Out till late
New Target state: NeedingSleep

Current state: NeedingSleep
Action process: Coffee and work
New Target state: ChronicIllness

----- END STATE -----

*****
* Test: ready -> needing sleep -> ready *
*****
Current state: Ready
Action process: Out till late
New Target state: NeedingSleep

Current state: NeedingSleep
Action process: Sleep
New Target state: Ready

----- END STATE -----
```

```
*****
* Test: unsupported actions *
*****
Ready state doesn't support coffee & work action
.....
Current state: Ready
Action process: Exercise
New Target state: Fit

Fit state doesn't support cheating action
.....
Current state: Fit
Action process: Perseverance and hard work
New Target state: Graduate

Graduate state doesn't support sleep action
----- END STATE -----

Process finished with exit code 0
```

Answer – 2

1 METHOD

Proje tasarımımda Proxy(Vekil) Tasarım Desenini kullandım. **Proxy** tasarım deseni **structural** tasarım desenlerinden biridir. Bir nesneye erişimi kontrol etmek için, **proxy** nesne kullanır. Bu nesne erişilecek nesneyi kontrol eder. Erişilecek nesne remote, yaratılması pahalı olan veya erişim için yetki isteyen gibi türlere sahip olabilir.

1. Remote nesne kullanmak istediğimizde kullanılır.(**Remote proxy**, farklı adres uzayında bulunan bir nesneye lokal temsilci sağlar)
2. Masraflı nesnelerin ihtiyaç duyulduğunda yaratılmasını sağlamak için(Örneğin resim yüklenmeden önce yükleniyor yazısı göstermek için. Bazı resimlerin boyutu büyük olduğu için yüklenmesi geç olabilmektedir. Burada belirtilen masraflı ifadesi büyük resim nesnesini temsil eder) kullanılır. Bu tarz proxy'lere **virtual proxy** denir.
3. Client'ın yetkisine göre işlem yapılıp yapılmamasını belirlemek için kullanılır. (**Protection proxy**)

Not: Üç temel **proxy** türü vardır. Bunlar: **Remote**, **Virtual** ve **Protection** proxy'dir.

RMI NEDİR?

Remote Method Invocation yani RMI farklı sunucularda olan metodların başka bir sunucudan çağırılarak kullanılmasında kullanılmaktadır. Java Nesnelerinin kullanmak için farklı bir sunucuya/sanal makina'ya bağlanıyorsa buna Dağıtık Programlama denir. RMI, Dağıtık Programlama yapılmasına olanak sağlamaktadır. RMI mimarisi, katmanlardan oluşmaktadır. Bu katmanlar istek, cevaplama, yönlendirme işlemlerini yapmaktadır.

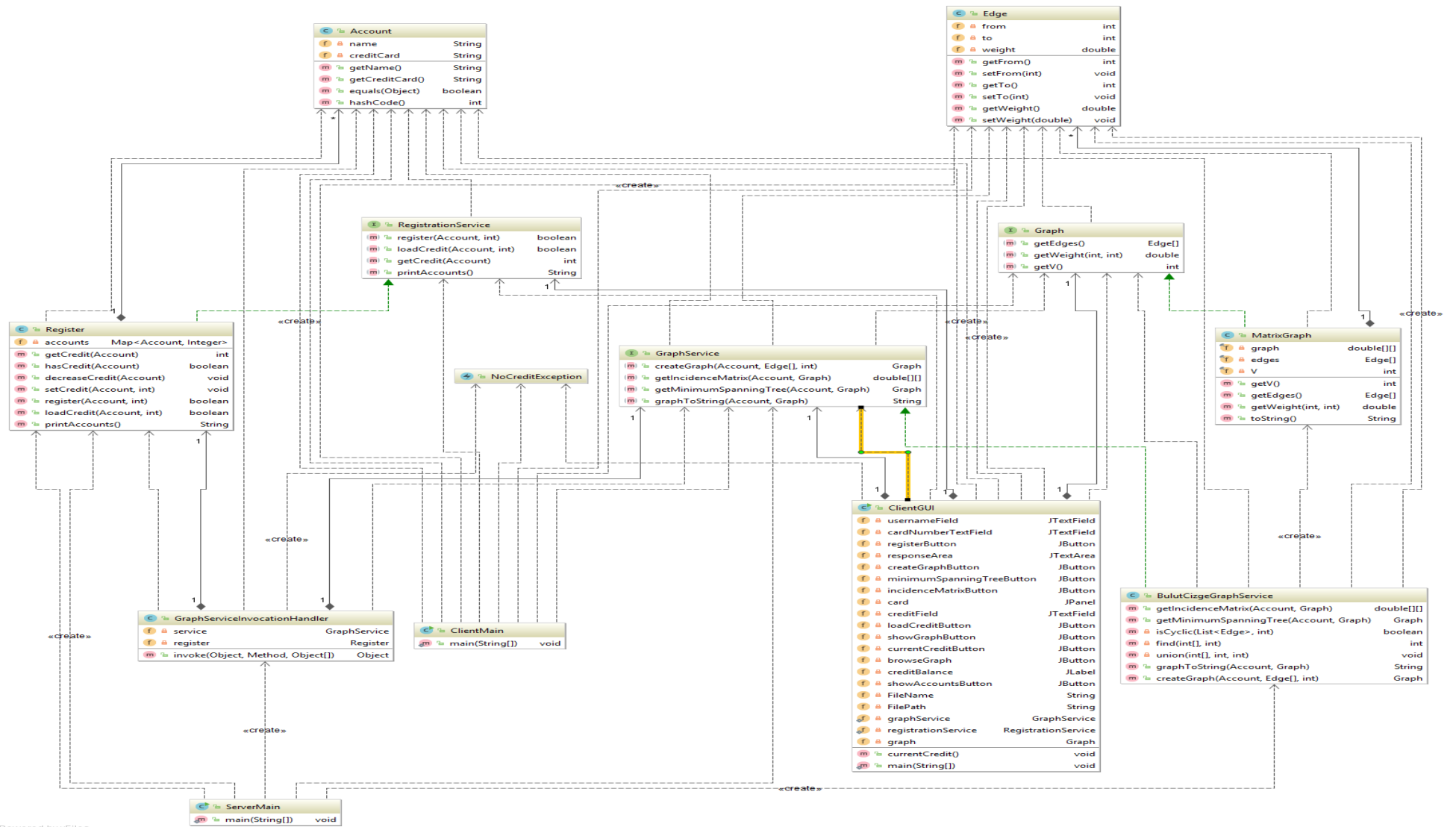
Stub: Uygulamadan gelen isteklerin ulaştığı yer Stub'dur. Stub Bağlantı ile başlar, JVM (Java Virtual Machine) ile iletişime geçer. Hataları yakalar, Skeleton'dan gelen sonuçları istemciye iletir.

Skeleton: Ulaşan isteklerin ne olduğuna bakarak yönlendirme görevini sağlar. Parametre okur, uzak nesneyi çağırır, metotdan gelen sonucu almakla görevlidir.

Remote Reference: Uzaktaki nesnelerin ne yapması gerektiğinin anlaşıldığı yerdir. Skeleton'dan gelen istekleri Transport katmanının anlayacağı şekle çevirir. Nesne işlemlerinin yürütüldüğü yerdir.

Transport: Bağlantı ile ilgili işlemlerin yapıldığı katmandır. TCP'nin kullanılması işlemlerini yapmaktadır.

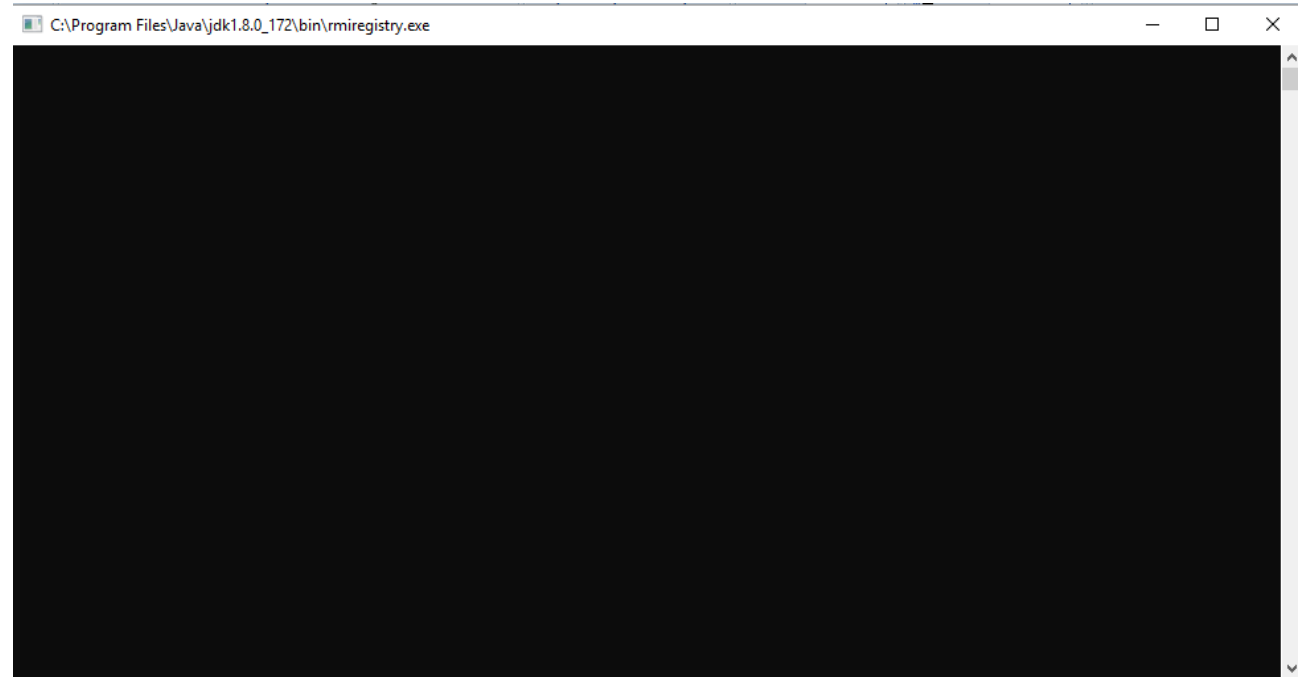
Class Diagrams



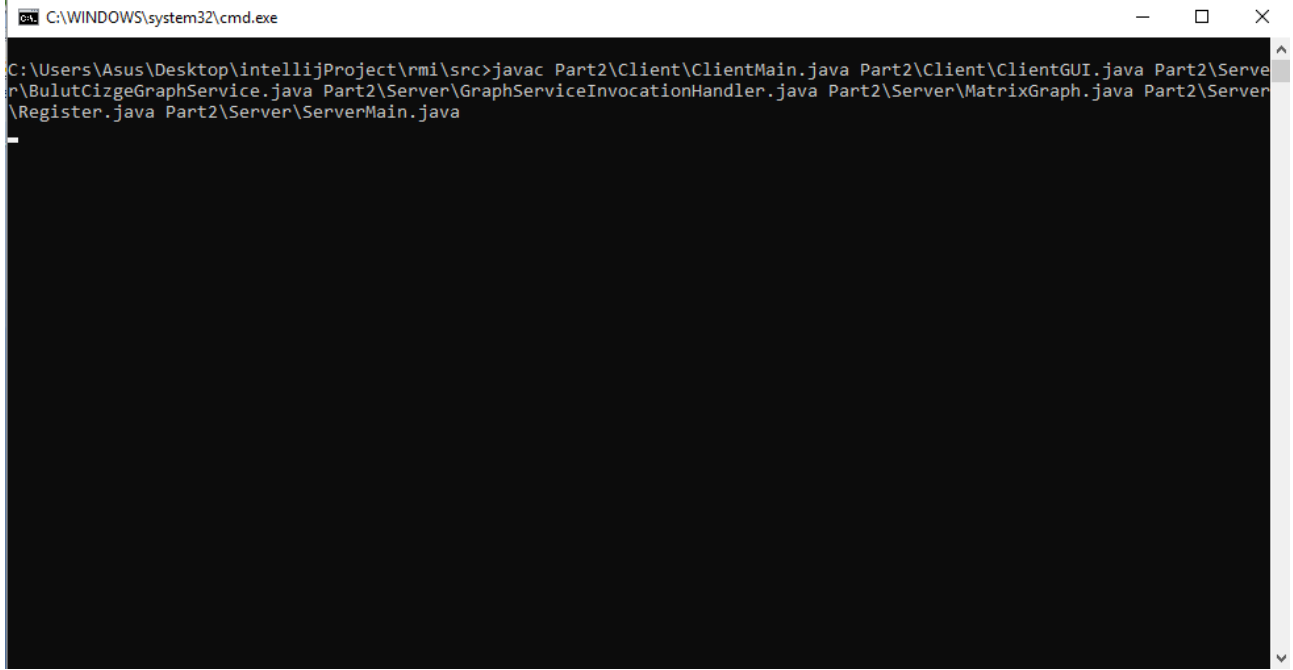
2 Result

Aşağı kısımda ilk önce startRMIRegistry.bat dosyası çalıştırılır

intellijProject > rmi > src		Ara: src	
Ad	Değişirme tarihi	Tür	Boyut
Part2	29.12.2018 21:46	Dosya klasörü	
compile.bat	30.12.2018 23:33	Windows Toplu İş ...	1 KB
startRMIRegistry.bat	30.12.2018 21:17	Windows Toplu İş ...	1 KB



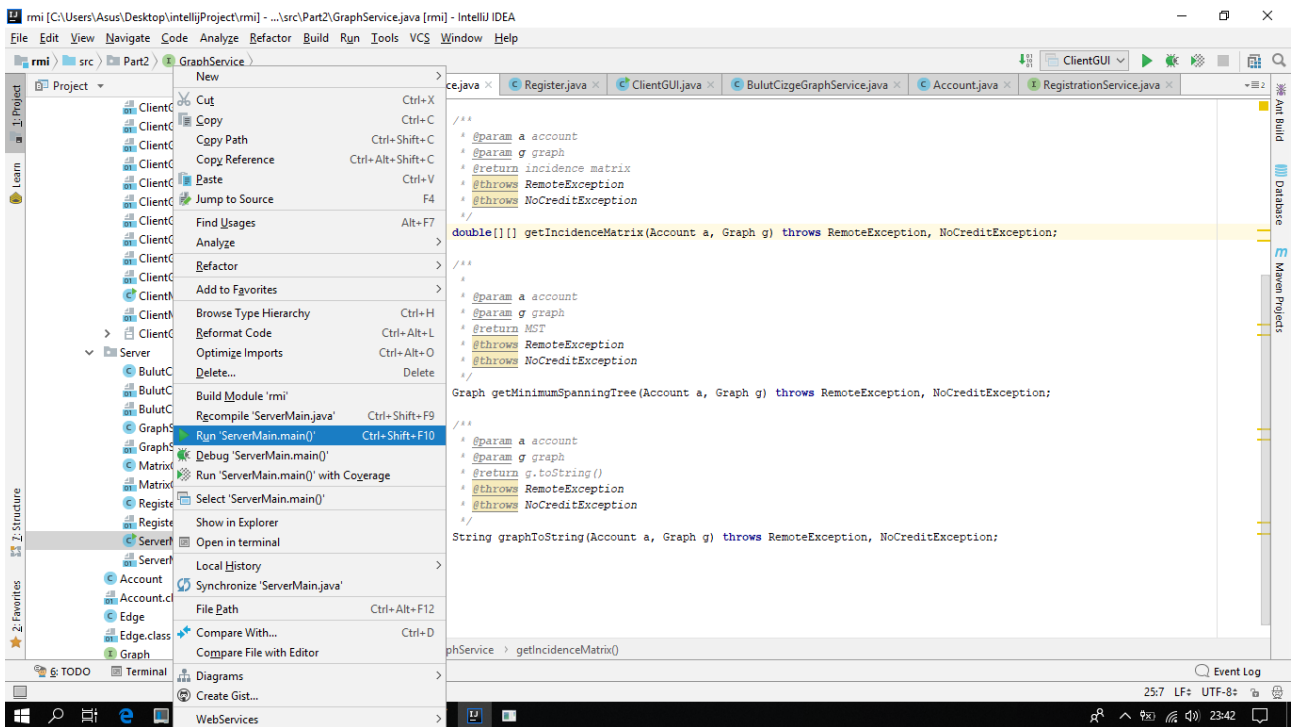
Daha sonra “compile.bat” dosyası çalıştırılır.



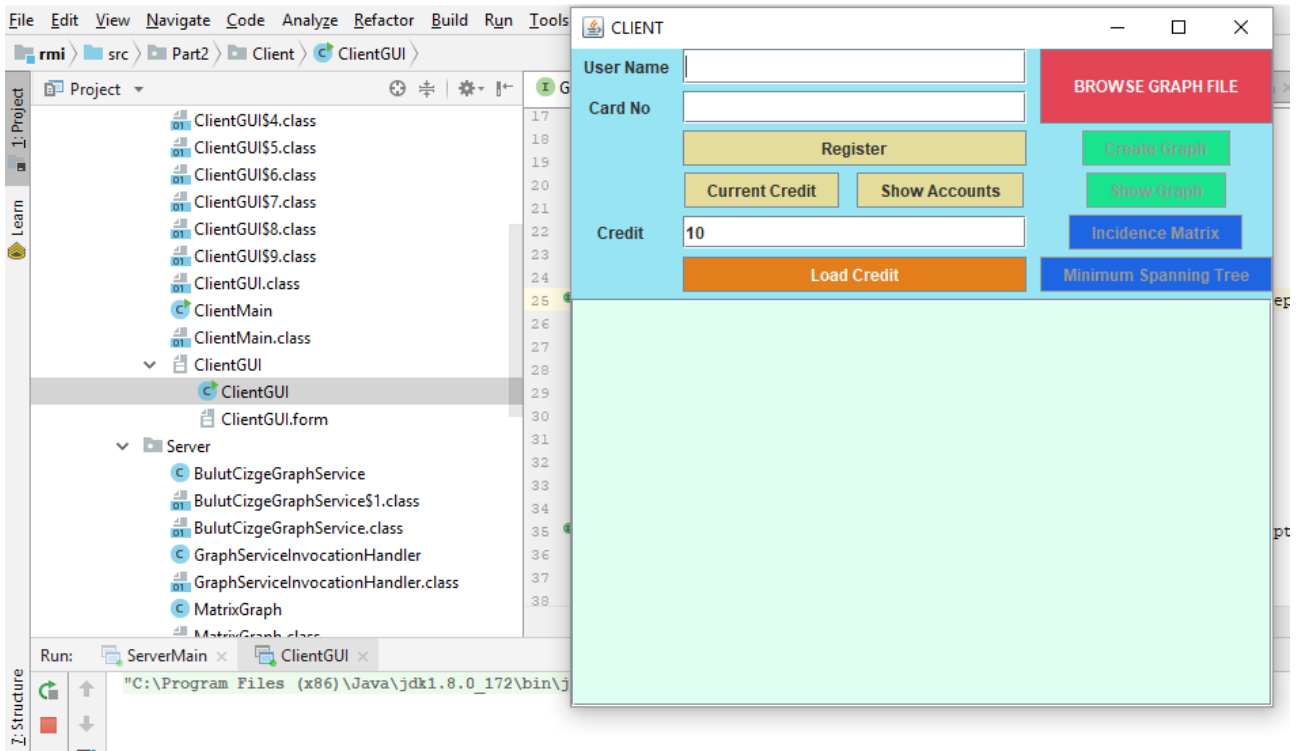
```
C:\WINDOWS\system32\cmd.exe

C:\Users\Asus\Desktop\intellijProject\rmi>javac Part2\Client\ClientMain.java Part2\Client\ClientGUI.java Part2\Server\BulutCizgeGraphService.java Part2\Server\GraphServiceInvocationHandler.java Part2\Server\MatrixGraph.java Part2\Server\Register.java Part2\Server\ServerMain.java
```

IntelliJ Projesini açtıktan sonra “ServerMain.java” dosyasına sağ tıklayarak “RUN” edilir.



Server aktif olarak çalışmaktadır. Daha sonra “ClientGUI.java” veya “ClientMain.java” ile Client çalıştırılmış olur. Aşağıda ClientGUI’ nin çalışması gösterilecektir.



Burada User Name kısmına kullanıcı ismini, Card No kısmını ise kart numarasını girer. “Register” butonuna basılarak hesap oluşturulur.

CLIENT

User Name

yunus

Card No

123

100

Register

Current Credit

Show Accounts

Credit

10

Load Credit

BROWSE GRAPH FILE

Create Graph

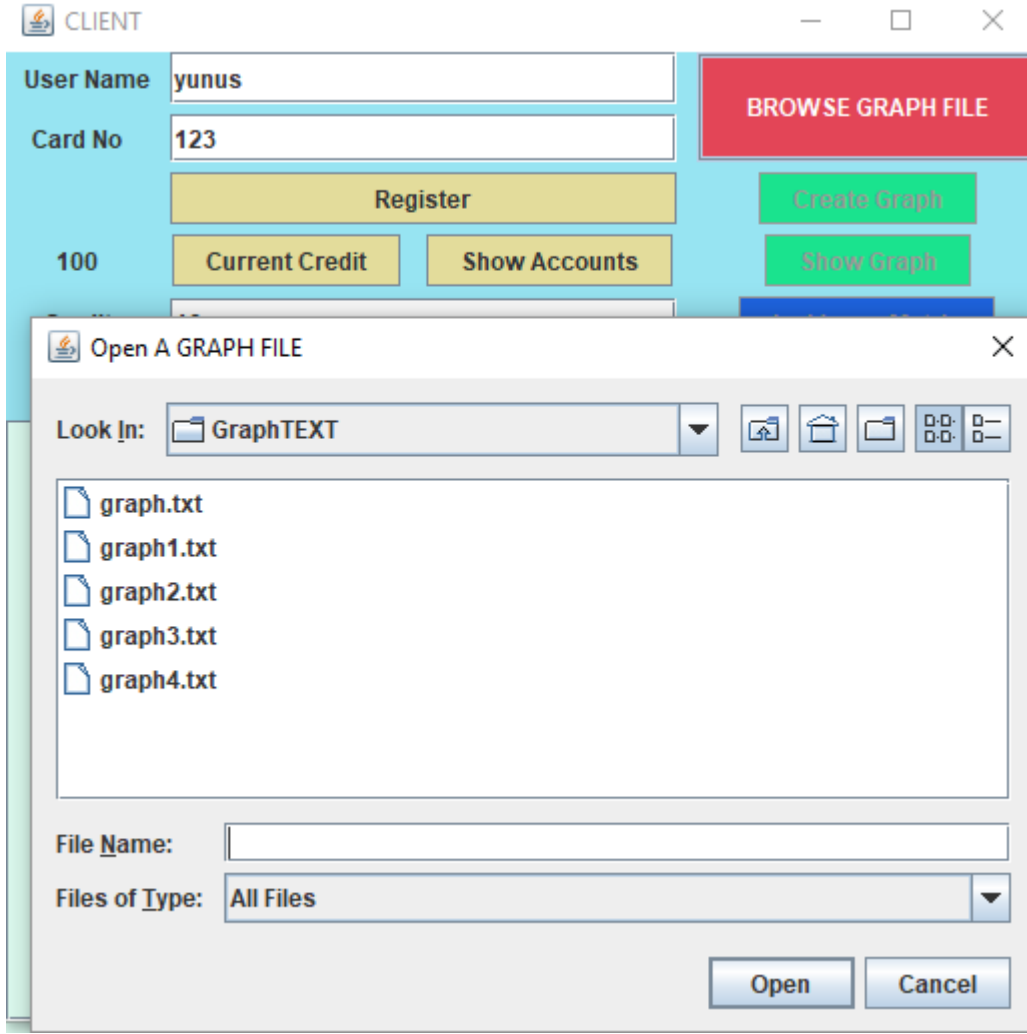
Show Graph

Incidence Matrix

Minimum Spanning Tree

Account registered

Burada “Load Credit” denildiği takdirde kullanıcıya ek kredi verilir. Sağ yan tarafta bulunan BROWSE GRAPH FILE ile Graph dosyası seçilir.



Seçilen graph dosyası sonucu “Create Graph” ile graph oluşturulur. “Show Graph” ile bu graph gösterilir. Incidence Matris ile de sonuçlar gösterilir.

CLIENT

User Name

yunus

Card No

123

97

Register

Current Credit

Show Accounts

Credit

10

Load Credit

BROWSE GRAPH FILE

Create Graph

Show Graph

Incidence Matrix

Minimum Spanning Tree

10,000 0,000 10,000 0,000 0,000 0,000

10,000 7,000 0,000 1,000 8,000 0,000

0,000 7,000 0,000 0,000 0,000 9,000

0,000 0,000 10,000 0,000 0,000 0,000

0,000 0,000 0,000 1,000 0,000 0,000

0,000 0,000 0,000 0,000 8,000 9,000

Aynı işlem “Minimum Spanning Tree” içinde geçerlidir.