Gebze Technical University
Computer Engineering

Object Oriented Analysis and Design

CSE443 – 2020

HW2-REPORT

Yusuf Abdullah ARSLANALP

151044046

Part1

1-) Aşağıda bir singleton sınıfı ve bir main fonksiyonu tanımlanmıştır.

```
public class Singleton {
    private static Singleton unique_instance;

    public static Singleton get_instance(){
        if( unique_instance == null )
        {
            unique_instance = new Singleton();
        }
        return unique_instance;
    }

    private Singleton(){

    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Singleton s1 = Singleton.get_instance();
        Singleton s2 = s1.clone();
```

Yukarıdaki main fonksiyonundan görüleceği üzere s1 Singleton nesnesinin clone metodu main'de çağrılmak istediğinde hata vermiştir. Bunun sebebi Object sınıfına ait clone metodunun protected olarak tanımlanmasıdır.

Bu yüzden Object sınıfına ait clone metodu kullanılarak singleton objesi **kopyalanamaz**.

2-) Bu örnekteki Singleton sınıfımızda yukarıda ki Singleton sınıfı ile aynı olsun. Bir nesneyi kopyalamanın iki yolu vardır. Biri clone metodunu kullanmak diğeri ise constructor kullanmak.

- ➔ Clone metodunun niçin kullanılamayacağı ilk soruda zaten açıklandı.
- ➔ Constructor kullanılarak Singelton objesi kopyalanamaz. Çünkü Singleton sınıfının constructor'ı private olarak tanımlanmıştır.

3-)

a-) Aşağıda bir super sınıf, bir Singleton sınıfı ve bir main fonksiyonunun program çıktısı verilmiştir. Singleton sınıfı Super sınıfının alt sınıfıdır. Ve Super sınıfı Clonebla arayüzünü implement etmektedir.

```
public class Super implements Cloneable {
    protected Object clone() throws CloneNotSupportedException
    {
        return super.clone();
    }
}
```

```
public class Singleton extends Super {
    private static Singleton unique_instance;

    public static Singleton get_instance(){
        if( unique_instance == null )
        {
            unique_instance = new Singleton();
        }
        return unique_instance;
    }

    private Singleton(){

    }
}
```

```
public class Main  {
    public static void main(String[] args) throws CloneNotSupportedException {
        Singleton s1 = Singleton.get_instance();
        Singleton s2 = (Singleton) s1.clone();

        if( s1 == s2 ) System.out.println( "s1 and s2 are equal" );
        else System.out.println( "s1 and s2 NOT equal" );
    }
}
```

```
s1 and s2 NOT equal
```

➔ Yukardaki programda clone metodu s1 Singleton nesnesi kopyalandı. Bu yüzden artık Singleton nesnesi Kullanılamaz. Çünkü artık Singleton değil.

b-) Aşağıda Singleton sınıfında clone metodunu override edilmiştir. Singleton objesi clone metodu ile kopyalanmak istenince Exeption fırlatılarak bu durum engellenir. Bu durumda clon metodu ile singleton objesi kopyalanamaz. Aşağıda programa ait örnek bir çıktı görülmektedir.

```
public class Singleton extends Super {
    private static Singleton unique_instance;

    public static Singleton get_instance(){
        if( unique_instance == null )
        {
            unique_instance = new Singleton();
        }
        return unique_instance;
    }
    protected Object clone() throws CloneNotSupportedException
    {
        throw new CloneNotSupportedException();
    }
    private Singleton(){}
}
```

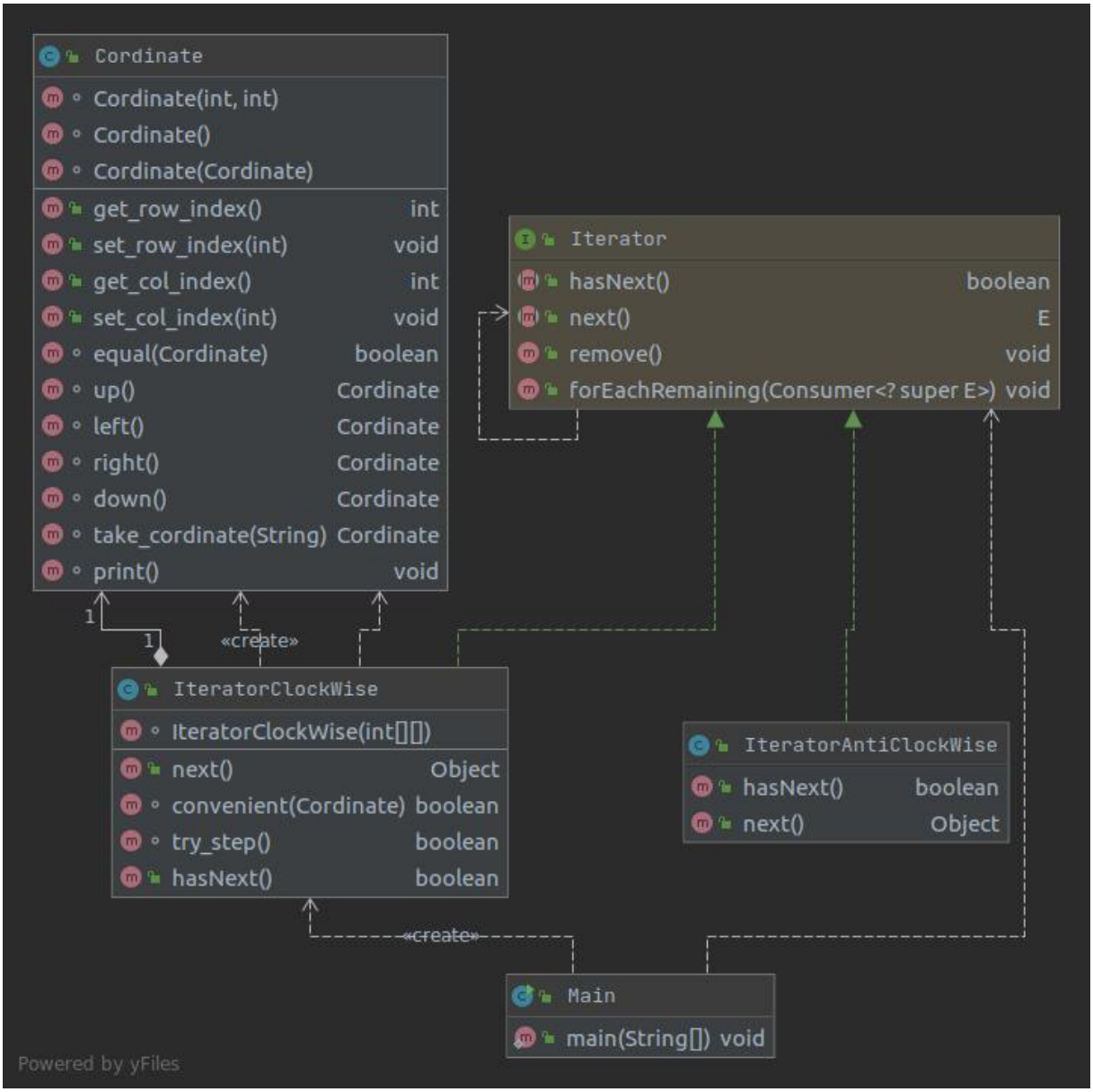
```
public class Main  {
    public static void main(String[] args) throws CloneNotSupportedException {
        Singleton s1 = Singleton.get_instance();
        Singleton s2 = (Singleton) s1.clone();

        if( s1 == s2 ) System.out.println( "s1 and s2 are equal" );
        else System.out.println( "s1 and s2 NOT equal" );
    }
}
```

```
Exception in thread "main" java.lang.CloneNotSupportedException
    at com.company.Singleton.clone(Singleton.java:15)
    at com.company.Main.main(Main.java:7) <5 internal calls>

Process finished with exit code 1
```

Part2



IteratorClockWise ve IteratorAntıClockWise sınıfları Iterator arayüzünü implement etmektedir. Ödevde sadece bir Iteratör implement etmemiz istendiği için sadece IteratorClockWise sınıfını implement ettim. Aşağıda iteratörün main fonksiyonunda çağırılma şekli ve ekarana basılan değerler gösterilmiştir..

```
while( itr.hasNext() )
{
    System.out.print( itr.next() + " " );
}
System.out.println( "" );
```

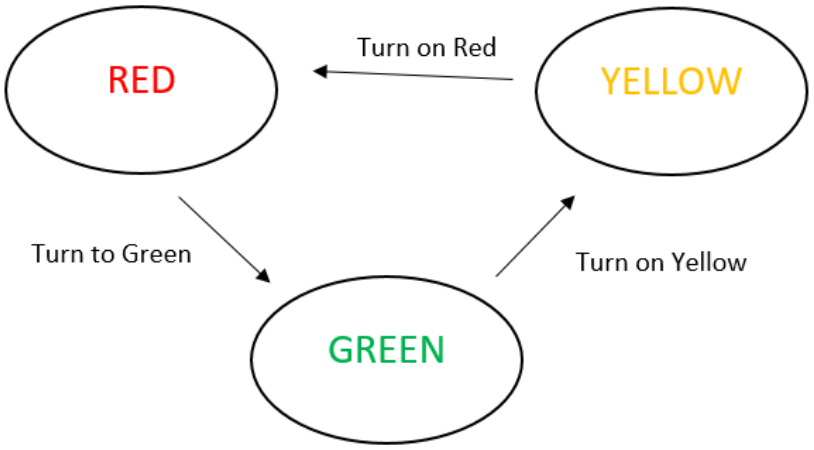
```
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
```

Part3

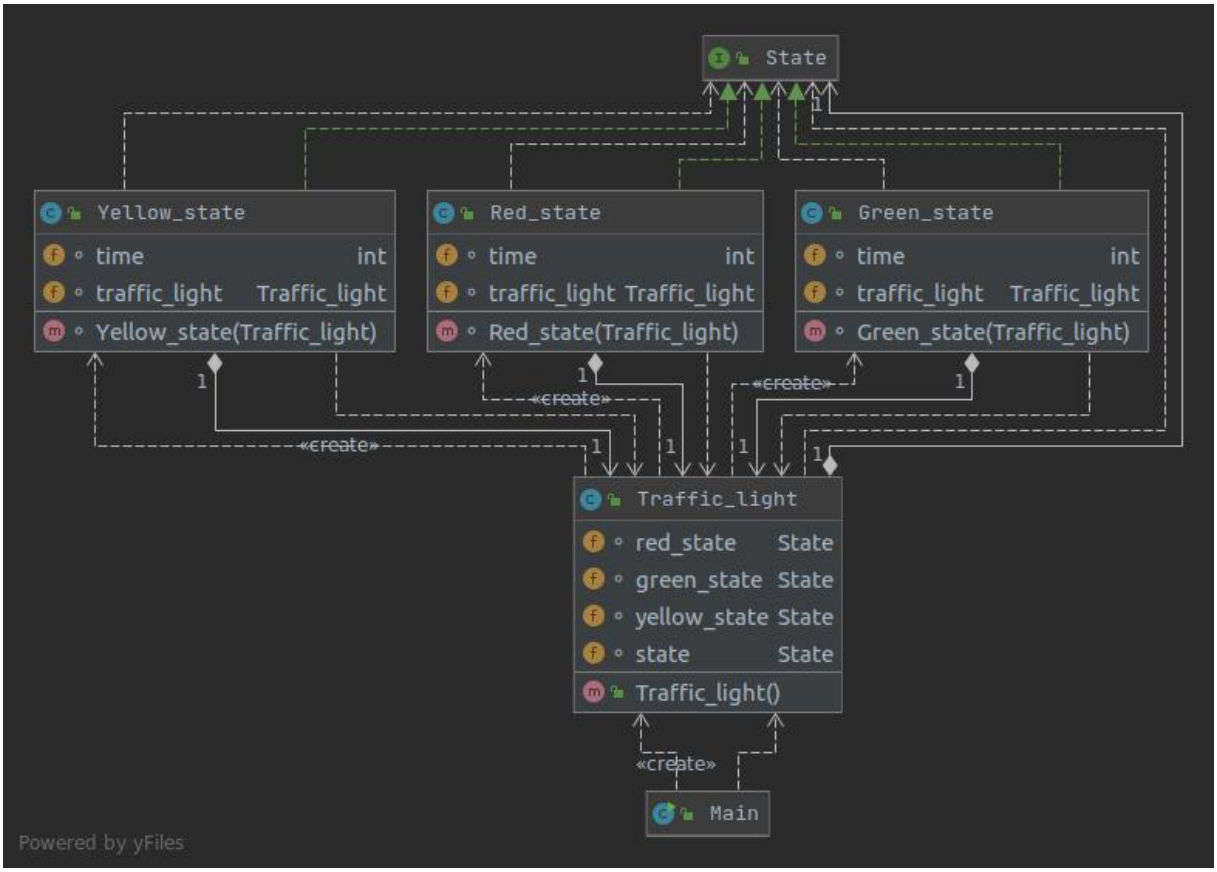
Part3 için state ve observer_and_state olmak üzere iki klasör mevcuttur. Her iki part içinde Javadoc oluşturulmuştur. İkinci kısım çalıştırıldığında kullanıcı sadece enter tuşlarına basarak programın çalışma şeklini gözlemleyebilir.

Sadece Durum Tasarım örüntüsü (İlk Kısım)

Trafik ışıkları için durum diagram aşağıda verilmiştir.



İlk kısım için Ssnıf diagram aşağıda verilmiştir



İlk kısım için main fonksiyon ve program çıktısı aşağıda verilmiştir.

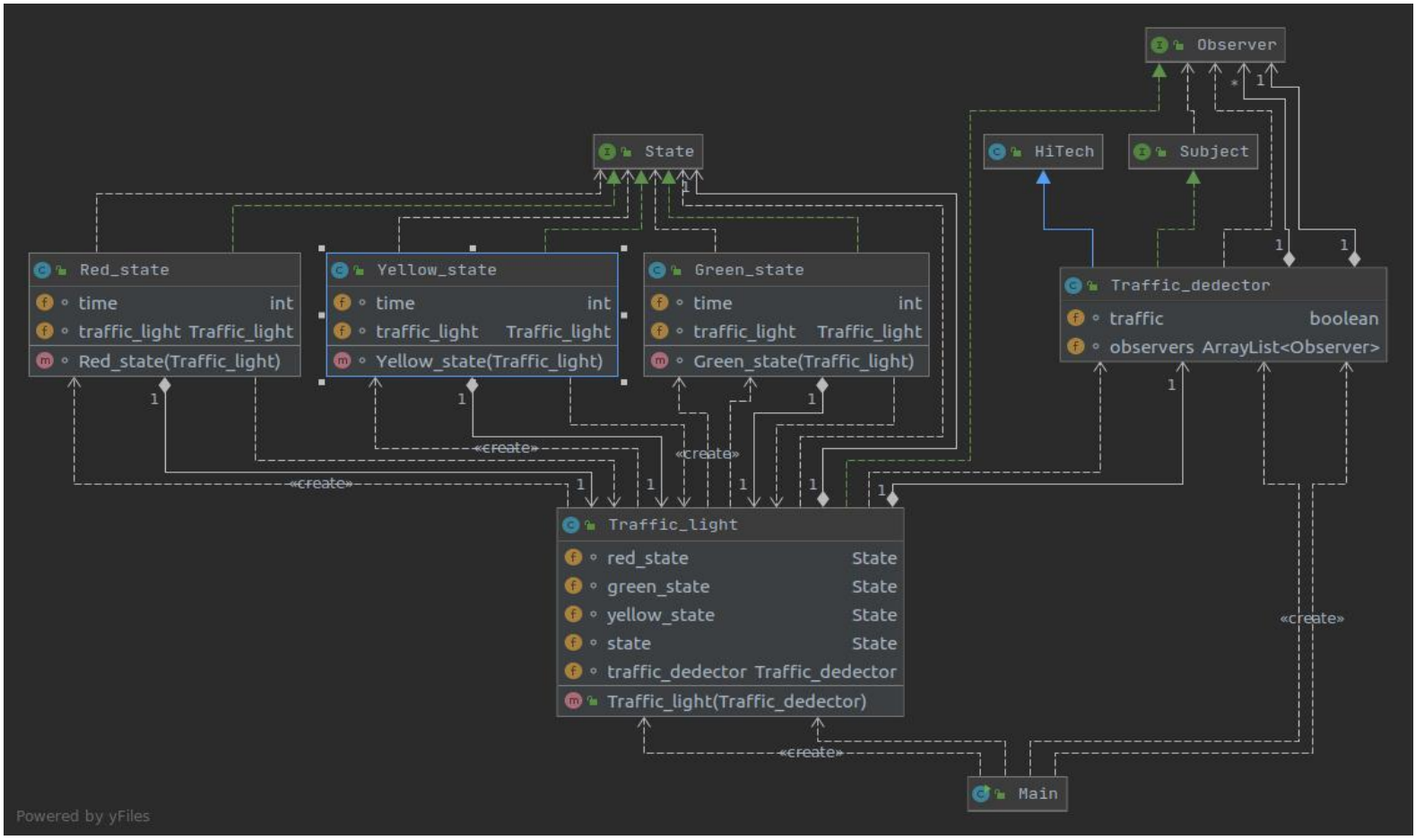
```
    public static void main(String[] args) {
        Traffic_light tl = new Traffic_light();

        tl.turn_on_red();
        tl.turn_on_green();
        tl.turn_on_yellow();
    }
```

```
RED (15sn)
GREEN (60sn)
YELLOW (3sn)
```

Durum ve Gözlemci Tasarım örüntüsü (İkinci Kısım)

Bu kısım için main fonksiyonu çalıştırılıldığında program kullanıcıdan input olarak enter tuşuna basmasın ister. Böylece kullnıcı programın çalışma şeklini kolayıkla takip edebilir.



Aşağıda main fonksiyonunun çıktısı verilmiştir

```
Now there is no traffic
Press enter to continue...

RED (15sn)
GREEN (60sn)
YELLOW (3sn)
You can see that green light turned on for 60 second When there are no traffic.

Press enter to continue...

Now the traffic increased. And now there are traffic
Press enter to continue...

RED (15sn)
GREEN (90sn)
YELLOW (3sn)
You can see that green light turned on for 90 second When there are traffic.

Press enter to continue...

Now the traffic turned normal. And now there are NO traffic
Press enter to continue...

RED (15sn)
GREEN (60sn)
YELLOW (3sn)
You can see that green light turned on for 60 second When there are no traffic.
```

Part4

Bu ödevde proxy tasarım örüntüsü kullanılarak çoklu thread özelliği olmayan bir sınıfa çoklu thread özelliği kazandırıldı.

Main fonksiyonunda ilk olarak çoklu thread özelliği olmayan DataBaseTable sınıfını test ettim. Ekran görüntüsünden de anlaşılacağı üzere tabloda 20 000 olması gereken yer 13 745 oldu.

Bu durumu düzeltmek için ProxyDataBaseTable sınıfını kullandım. Bu sınıf içerisinde bir adet DataBaseTable sınıfı bulunduruyor. Böylece DataBaseTable sınıfının birden fazla thread ile kullnılmasını sağlıyor. ProxyDataBaseTable sınıfının objesi birden fazla thread tarafından kullanılması ile tablonun merkezindeki değer olması gerektiği gibi 20 000 sayısına ulaşmıştır.

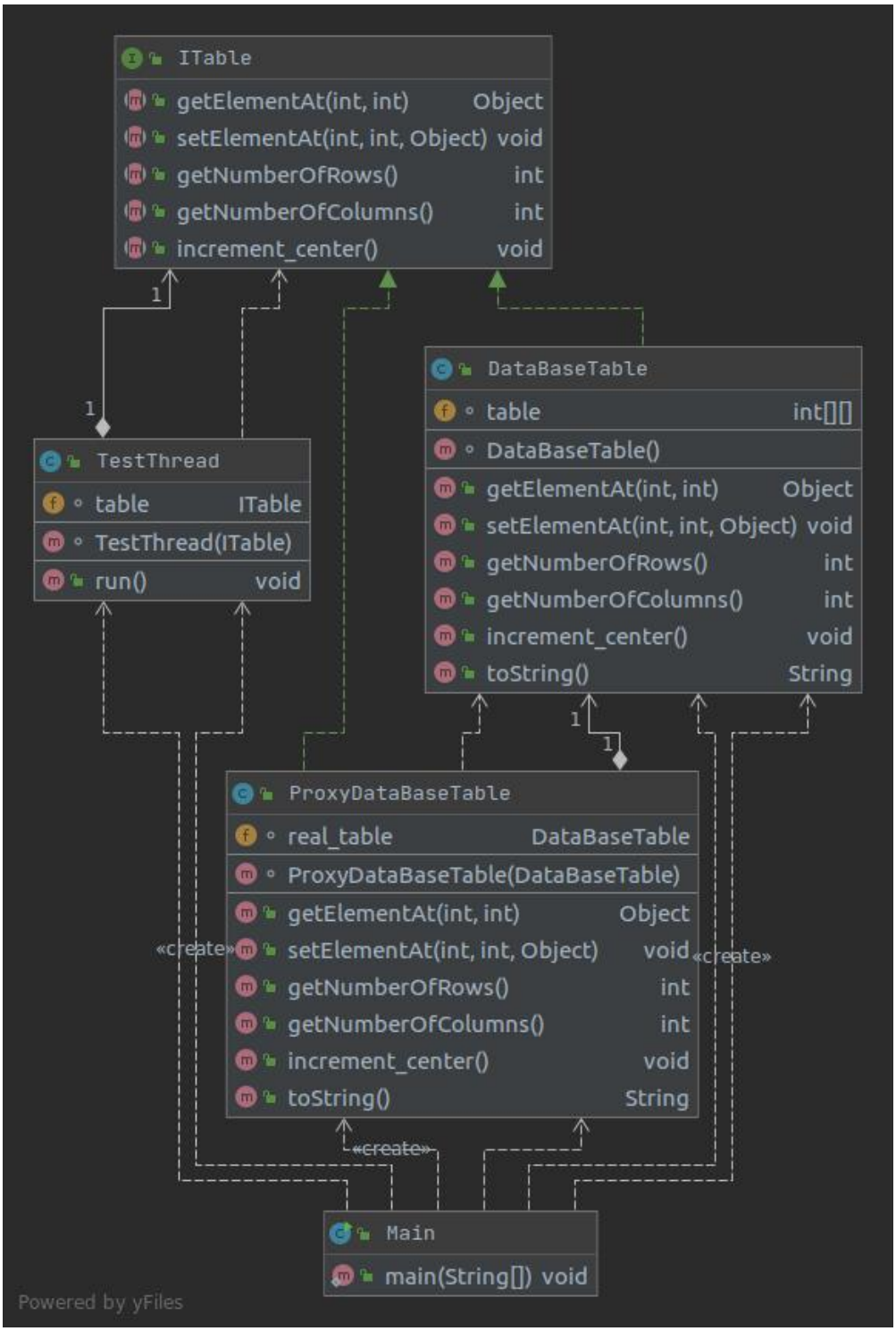
Main fonksiyonunun çıktısı aşağıdadır.

```
DataBaseTable has no multi thread property
Two thread increased center value of table 10 000 times.
The center value should be 20 000. But it is not.
With proxy design patern we can solve this issue
     0      0      0
     0  13745      0
     0      0      0

ProxyDataBaseTable class is a proxy for DataBaseTable class.
Due to ProxyDataBaseTable class center value of table is 20 000
     0      0      0
     0  20000      0
     0      0      0


Process finished with exit code 0
```

Sınıf diagram aşağıdadır.


ITable
getElementAt(int, int) Object
setElementAt(int, int, Object) void
getNumberOfRows() int
getNumberOfColumns() int
increment_center() void
1
1
TestThread
table ITable
TestThread(ITable)
run() void
DataBaseTable
table int[][]
DataBaseTable()
getElementAt(int, int) Object
setElementAt(int, int, Object) void
getNumberOfRows() int
getNumberOfColumns() int
increment_center() void
toString() String
1
1
ProxyDataBaseTable
real_table DataBaseTable
ProxyDataBaseTable(DataBaseTable)
getElementAt(int, int) Object
«create»
setElementAt(int, int, Object) void
«create»
getNumberOfRows() int
getNumberOfColumns() int
increment_center() void
toString() String
«create»
Main
main(String[]) void
Powered by yFiles