Gebze Technical University
Computer Engineering

Object Oriented Analysis and Design
CSE443 – 2018 Autumn

HOMEWORK 1 REPORT

Yunus ÇEVİK
141044080

Course Teacher: Erchan Aptoula

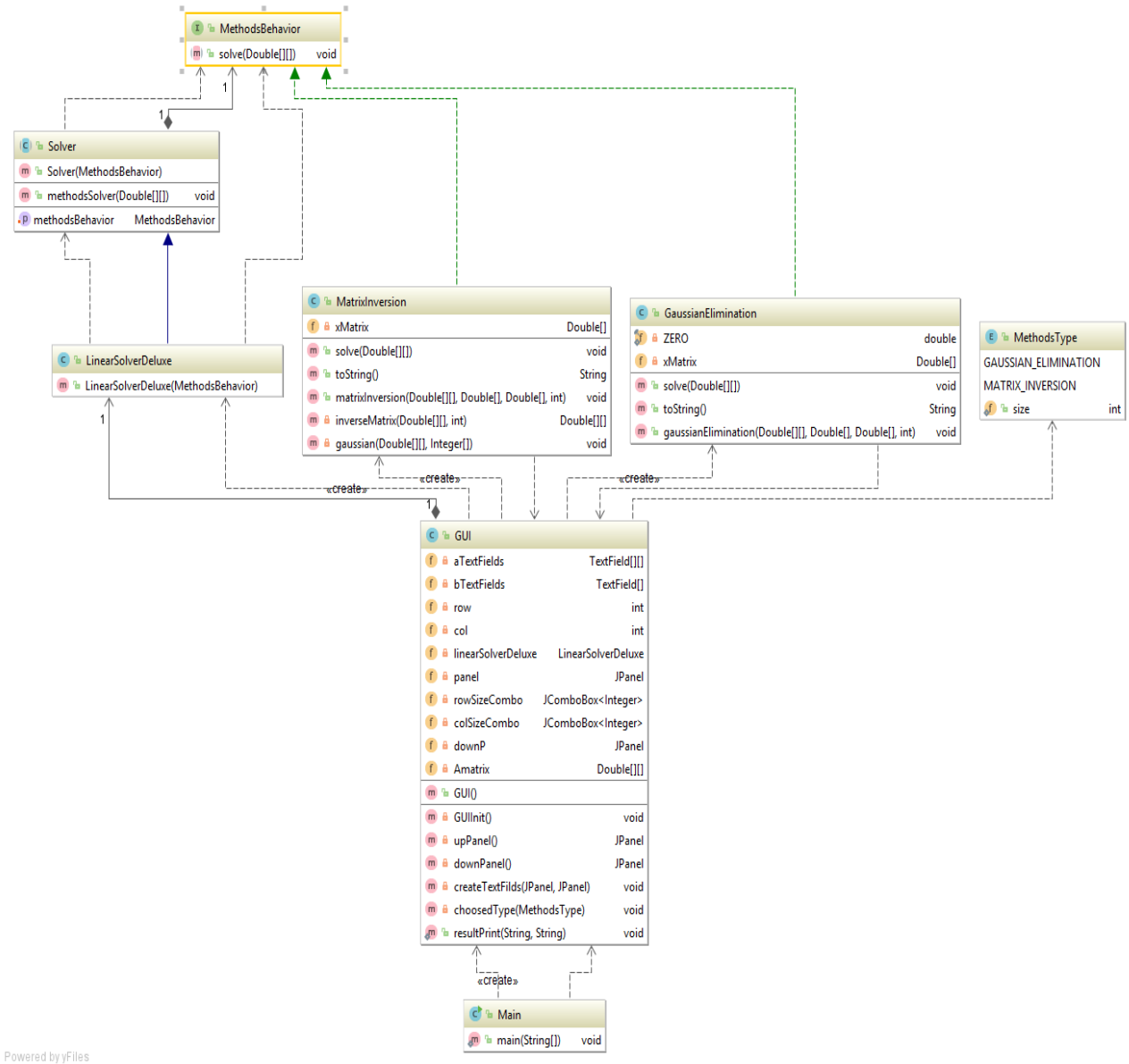
Course Assistant: Muhammet Ali Dede

Answer-1

1 METHOD

Linear Solver Deluxe projesini tasarlarken Strateji (Strategy) Tasarım Deseni kullanarak tasarladım. Bu tasarım deseni, bir işlem için farklı yöntemlerin uygulanabilir olduğu durumlarda, bu yöntemi kullanacak olan nesne, hangi yöntemin uygulanacağını seçer. Çünkü bu içerik nesnesi, yöntemleri belirleyen üst sınıfı içerir. Farklı yöntem veya strateji alt sınıfları da, bu üst sınıftan türerler. Bu tasarım deseniyle, yöntemin nasıl uygulanması gerektiği ile ilgili detaylar, bu yöntemi kullanacak nesneden ayrılmış olur.

1.1 Class Diagrams



2 Add New Metod

“Gaussian Elimination” ve “Matrix Inversion” metotlarının yanına yeni bir metot eklemek istediğimizde, Örneğin “Determinant Calculation” metodu bunun için yapmamız gereken işlemler aşağıdaki gibi olmalıdır.

1. **Adım**

```
public enum MethodsType {
    GAUSSIAN_ELIMINATION, MATRIX_INVERSION, DETERMINANT_CALCULATION ;
    public static int size = MethodsType.values().length;
}
```

2. **Adım**

```
private void choosedType(MethodsType choosed) throws Exception{
    if(row == col){
        switch(choosed){
            case GAUSSIAN_ELIMINATION:
                linearSolverDeluxe = new LinearSolverDeluxe(new
GaussianElimination());
                break;
            case MATRIX_INVERSION:
                linearSolverDeluxe = new LinearSolverDeluxe(new
MatrixInversion());
                break;
            case DETERMINANT_CALCULATION:
                linearSolverDeluxe = new LinearSolverDeluxe(new
DeterminantCalculation());
                break;
        }
    }
    else
        throw new Exception("No Square Matrix for this method")
}
```

Not: Yukarıdaki işlemler sadece kare matris işlemleri için yazılmış metotların oluşturulmasıdır. Kare olmayan matrisler üzerinde işlemler yapılacağı bir metot oluşturulduğunda ufak bir veya birkaç değişiklik yapılarak düzenleme yapılmalıdır.

3. **Adım**

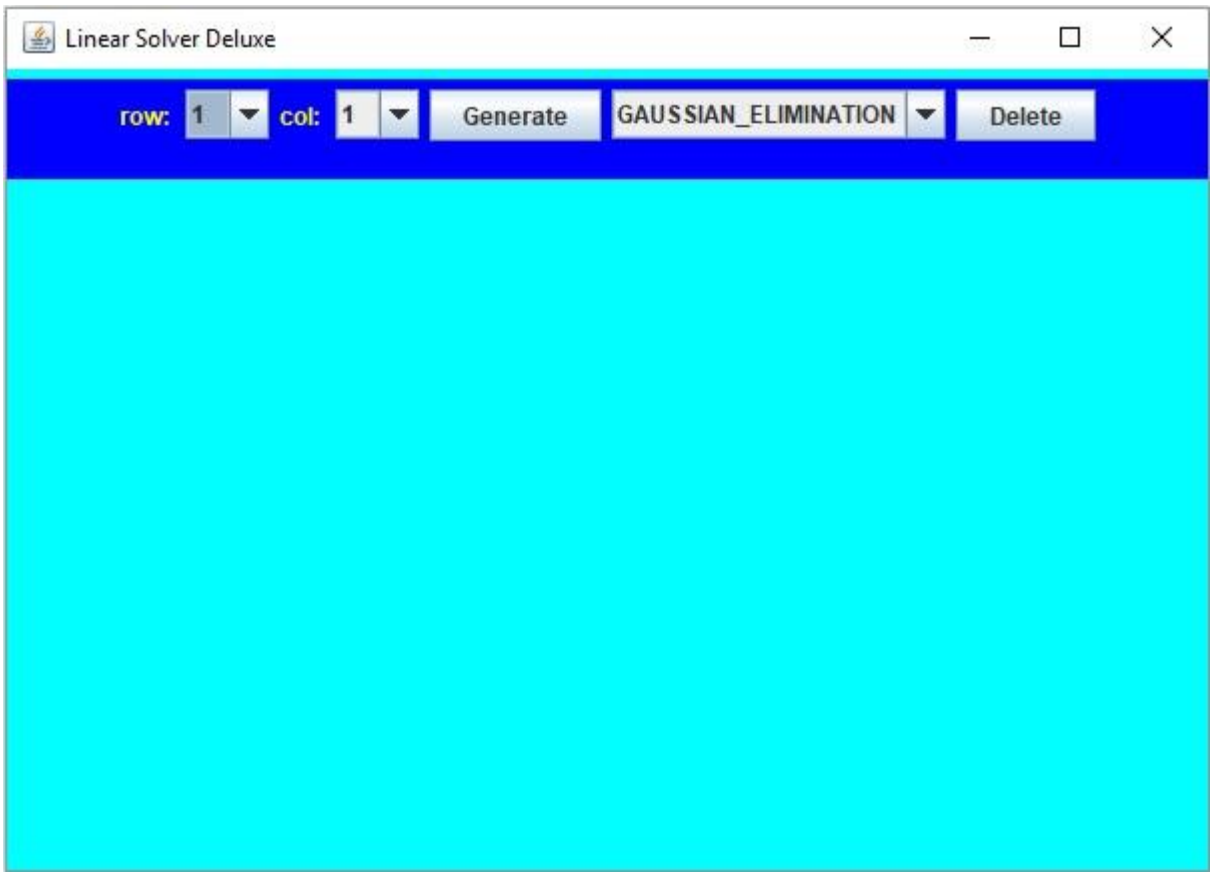
```
public class DeterminantCalculation implements MethodsBehavior {
    @Override
    public void solve(Double[][] matrix) {
        // Yeni metot buraya yazılır veya bu blok içinde çağırılır.
    }
    @Override
    public String toString() {
        // Bu kısım sonuç matrisinin "String" bir değişken içerisine be-
lirli bir format kapsamında eklenerek return etmesi ile yapılmalıdır.
    }
}
```

4. **Adım**

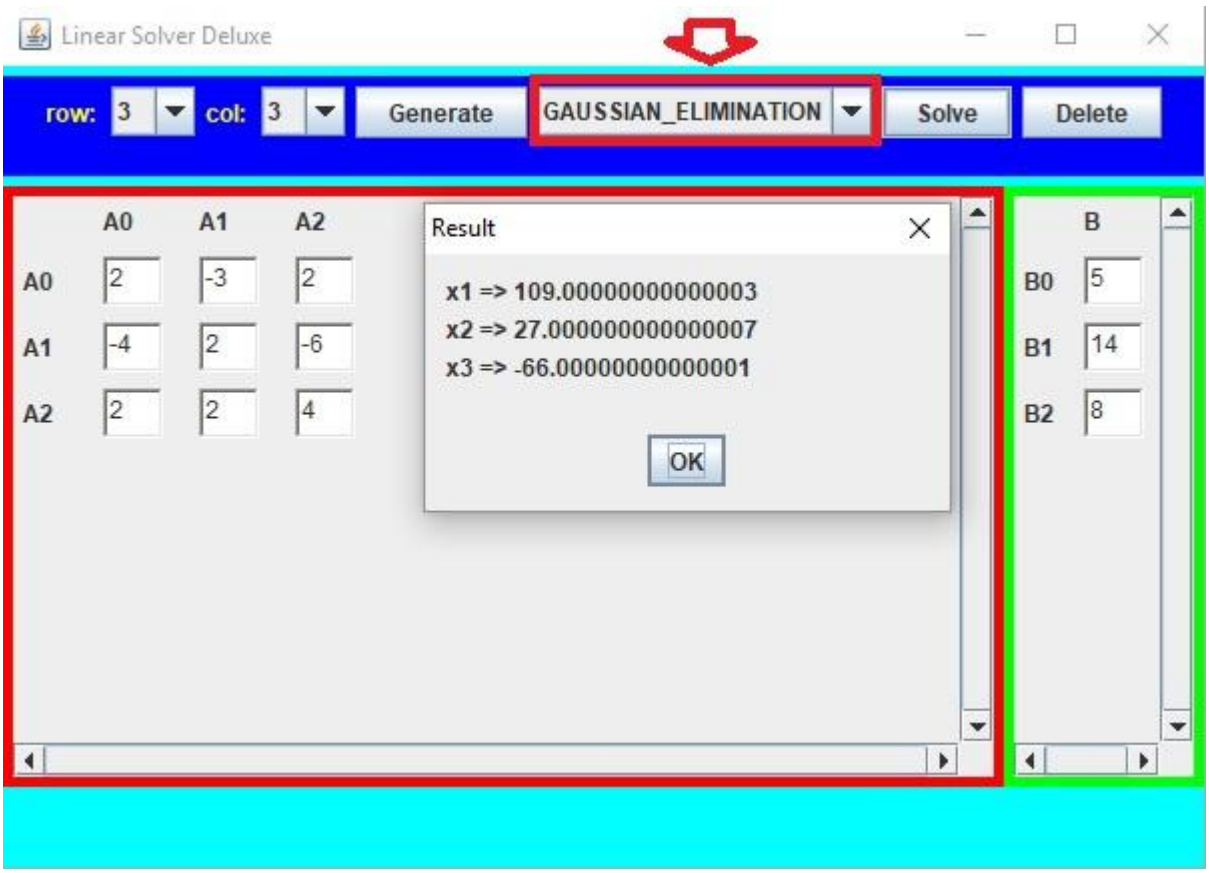
```
try {
    determinantCalculation (matrix, xMatrix, size);
    GUI.resultPrint("Result",this.toString());
} catch (Exception ex) {
    GUI.resultPrint("Warning", ex.getMessage());
}
```

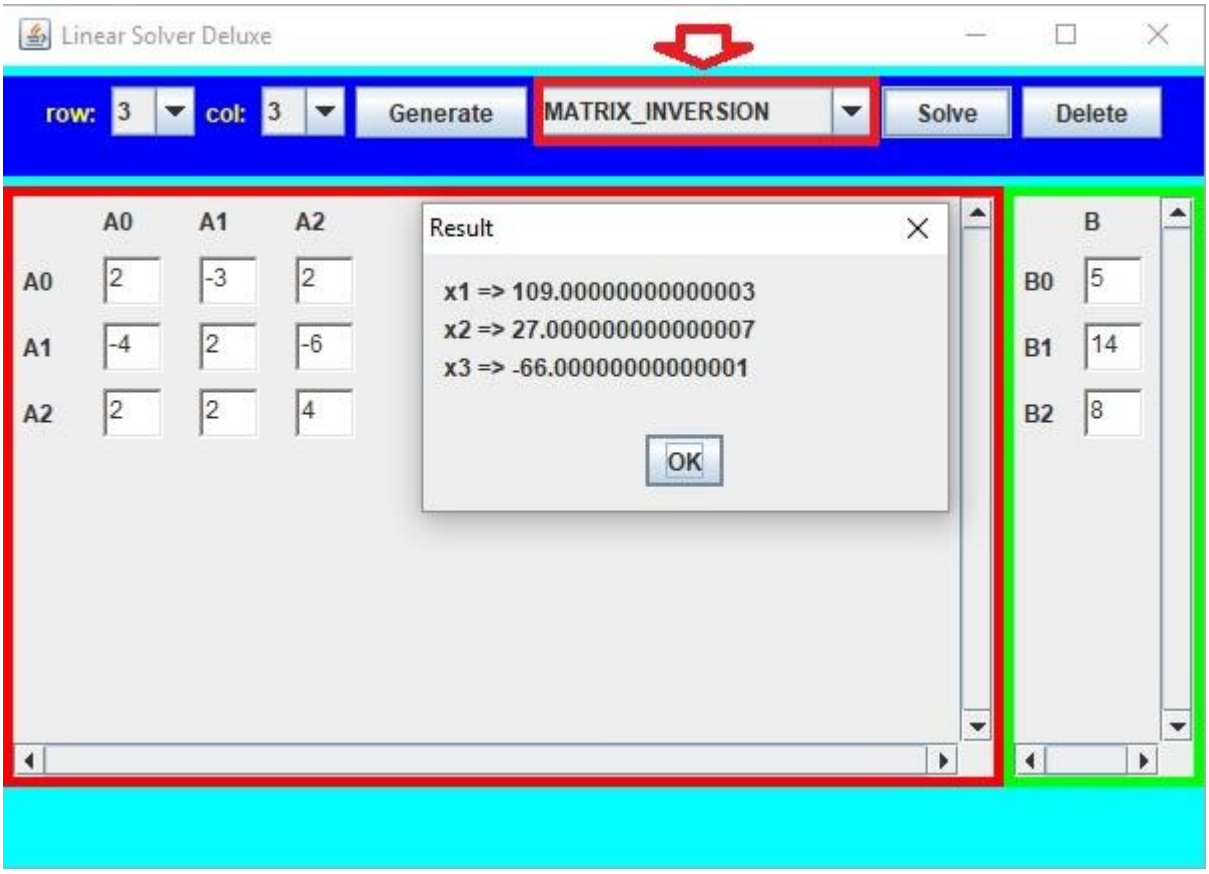
3 RESULT

Kullanıcıdan lineer denklem için gerekecek matris boyutları, “row” ve “col” ComboBox nesnelerinden alınır ve “Generate” butonu ile matris değerlerinin yazılacağı TextFields nesneleri oluşturulur.



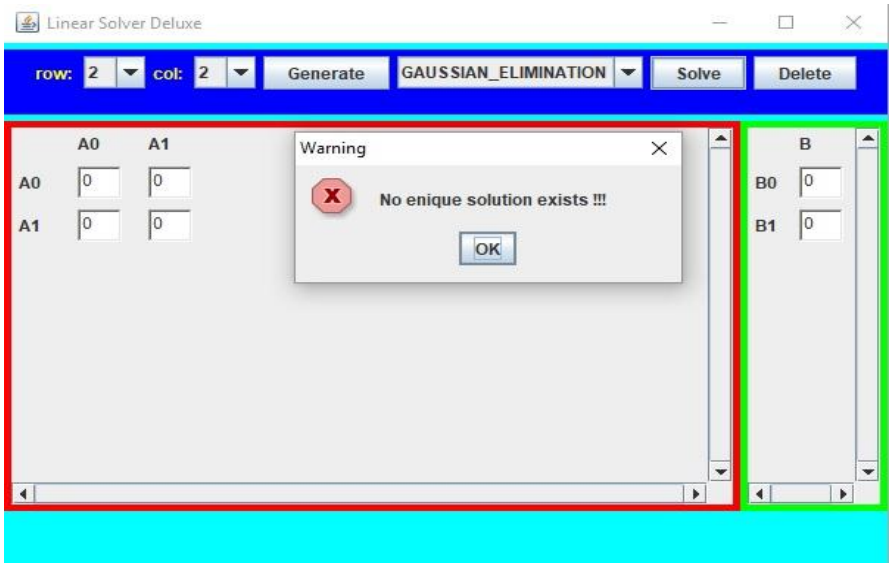
Matris değerleri yazılarak metot seçimi “GAUSSIAN_ELIMINATION” ComboBox ile yapılır ve “Solve” butonuna tıklandığında ekrana açılır bir pencerede sonuçlar gösterilir. Aşağıda “GAUSSIAN_ELIMINATION” ve “MATRIX_INVERSION” metotlarının sonuçları gösterilmektedir. Sonuçlar görüldüğü gibi aynı çıkmaktadır. Ancak bazı durumlarda “double” / “double” yapıldığı için küçük sapmalar olmaktadır. Bunun çözümü ise sonuçları **Math.round()** metodunu kullanarak ekrana çıktı olarak sunmaktır.

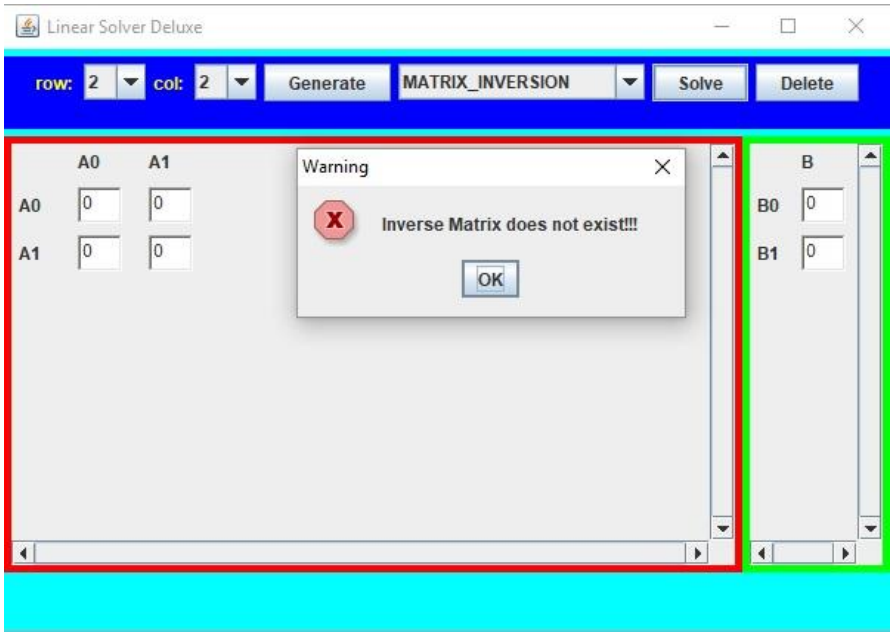

Linear Solver Deluxe
row: 3
col: 3
Generate
GAUSSIAN_ELIMINATION
Solve
Delete
A0
A1
A2
A0 2 -3 2
A1 -4 2 -6
A2 2 2 4
Result
x1 => 109.00000000000003
x2 => 27.000000000000007
x3 => -66.00000000000001
OK
B
B0 5
B1 14
B2 8


Linear Solver Deluxe
row: 3
col: 3
Generate
MATRIX_INVERSION
Solve
Delete
A0
A1
A2
A0 2 -3 2
A1 -4 2 -6
A2 2 2 4
Result
x1 => 109.00000000000003
x2 => 27.000000000000007
x3 => -66.00000000000001
OK
B
B0 5
B1 14
B2 8

Program belirlenen metotlar çerçevesinde çözüm gerçekleştiremediğinde oluşacak hatalar şu şekilde olacaktır.

“ERROR MESSAGE”



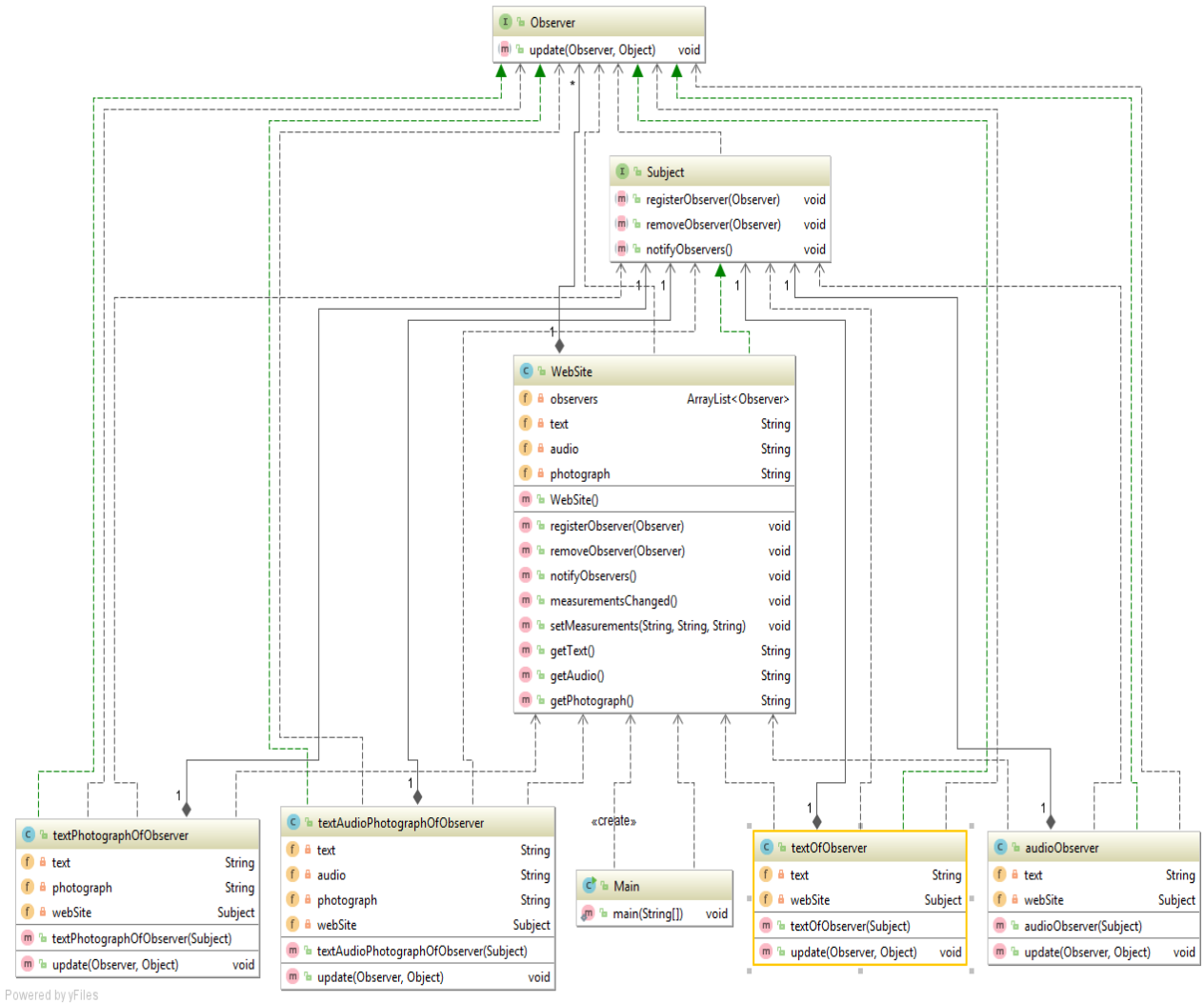


Answer-2

1 METHOD

Bir WebSite uygulamasında gözlemcilerin favori WebSite belirlemeleri ve bunları metin (text), ses (audio) veya fotoğraf(photograph) gibi özelliklere yaptıklarını varsayarsak. Bu WebSite' nin birden fazla abonesi ve bu abonelerin eklenip – çıkarılması ve tüm abonelere yayın yapması gerekir. Bu yüzden bu WebSite tasarımı için tasarım kalıplarından "Gözlemci Tasarım Deseni (Observer Design Pattern)" kullanılır. Gözlemci(Observer) tasarım deseni, behavioral tasarım desenlerinden biridir. Nesneler arasında one-to-many ilişki sağlar. Bir nesne durumunu değiştirdiğinde, ona bağlı diğer tüm nesneler uyarılır ve otomatik olarak güncellenir.

1.1 Class Diagrams



Öncelikle iki tane interface sınıfına ihtiyacımız vardır: Subject ve Observer. Subject interface sınıfı durumu değişecek nesneyi temsil ederken, Observer türünden olan nesneler ise Subject türündeki nesneyi gözlemleyecek ve bir değişiklik olduğu zaman uyarılacaktır. Yani burada Subject etkileyen nesneyi, Observer ise etkilenen nesneleri temsil eder.
Observer tasarım deseni one-to-many prensibini uyguladığı için tek bir tane Subject olmalı, birden çok ise Observer sınıfı olmalıdır.

Observer sınıfı. Görüldüğü gibi constructor parametre olarak somut Subject türünden (örnek textOfObserver) bir nesne alıyor ve bu nesnenin registerObserver() metodunu kullanarak kendisini kaydediyor veya removeObserver() metodunu kullanarak silme yapıyor. update() metodunda ise Subject sınıfının yaptığı değişikliği alıyor ve işliyor.
textOfObserver, audioOfObserver, textPhotographOfObserver ve textAudioPhotograph sınıfları Observer tasarım deseninde somut Subject sınıfını temsil eder.

textOfObserver ➔ sadece metin (text) ile ilgili olanları favori webSite olarak seçer.
audioOfObser ➔ sadece ses (audio) ile ilgili olanları favori webSite olarak seçer.
textPhotographOfObserver ➔ metin ve fotoğraf (photograf) ile ilgili olanları favori webSite olarak seçer.
textAudioPhotograph ➔ metin, ses ve fotoğraf ile ilgili olanları favori webSite olarak seçer.

Yeni bir favori webSite eklenmek istendiğinde örnegin sadece fotoğraf ile ilgili favori webSite eklenmek istendiğinde şu şekilde yapılmalıdır.

```
public class photographOfObserver implements Observer {
    private String  photograph;
    private Subject webSite;

    public photographOfObserver (Subject webSite) {
        this.webSite = webSite;
        webSite.registerObserver(this);
    }

    public void update(Observer observer, Object object) {
        if (observer instanceof WebSite) {
            WebSite webSite = (WebSite)observer;
            photograph = webSite.getPhotograph();
        }
    }
}
```

Yapılması gereken olay yeni bir class oluşturup bu class içirisinde private değişkenini belirlemek ve bu değişkene göre website nesnesinden uygun olan getter metodu çağrılmış olur.