CSE 443 - Object Oriented Analysis and Design

Homework1 Rapor

Fürkan YILDIZ
141044031

Part1)

Bahsedilen problem tam olarak Strategy Design Pattern'e uyuyor çünkü,
Lineer denklemleri çözebilmek için Gaussian elimination, matrix inversion gibi yöntemler (stratejiler) var ve ileride bu yöntemlerin çeşitlenebileceğinden bahsediliyor. Aynı zamanda lineer denklem çözülürken, çözüm yönteminin dinamik olarak seçilebilmesi isteniyor. Ve tabiki bunları yaparken sınıfların birbirine bağımlığı minimum seviyede olmalı. İşte tüm bunlar problemi Strategy Pattern'e götürüyor.

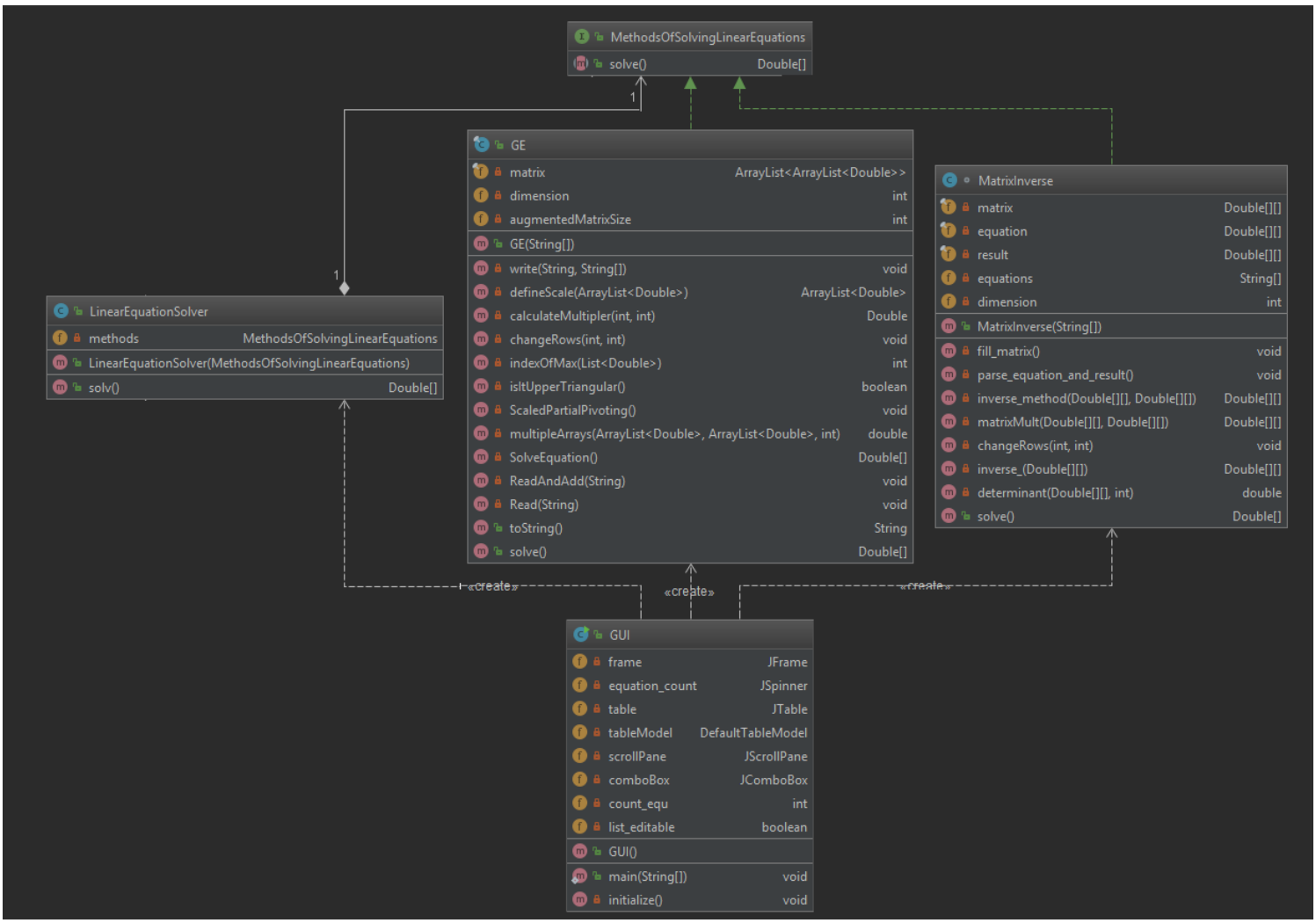
"MethodsOfSolvingLinearEquations" adında bir interface oluşturduk. Bu interface design pattern' in Strategy' si. Bize equation'un çözümü için "solve" methodu sağlıyor ve çözüm değerlerini double arrayi olarak return ediyor.

"GE" sınıfı Gaussian elimination yönteminin kodlanmış sınıfı ve lineer çözüm yöntemi olduğundan "methods_of_solving_linear_equations" 'dan implement ediliyor. "solve" methodunu Gaussian elimination yöntemine göre implement ediyor. Design patterne göre stratejilerden bir tanesi.

"MatrixInverse" sınıfı Matrix Inversion yönteminin kodlanmış sınıfı ve lineer çözüm yöntemi olduğundan "methods_of_solving_linear_equations" 'dan implement ediliyor. "solve" methodunu Matrix Inversion yöntemine göre implement ediyor. Design patterne göre stratejilerden bir tanesi.

"LinearEquationSolver" sınıfı ise bu design pattern'in content'i. Lineer çözüm yöntemi (MethodsOfSolvingLinearEquations) alarak, linear denklemleri çözmek için bir fonksiyon("solve") sağlıyor.

GUI' den seçilen lineer method a göre Gaussian elimination ya da matrix inversion sınıflarının interfaceleri tipinde objeleri oluşturuluyor ve bu objelerine, GUI'den girilen equationlar veriliyor (String arrayi: her array elemanı bir equation) böylece linear equationlar matrix olarak objelerin içine kaydediliyor. Daha sonra interface' nin solve methodunu çağırarak, dinamik olarak üretilen objenin solve methodu çağırılıyor. Bu method da equation çözülüyor ve bilinmeyenler (x0,x1,x2,…,xn) Double array olarak solve methodundan return ediliyor.
Bu işlemler yapılırken, solution yoksa ya da birden fazla çözüm varsa exception fırlatılır. GUI' de yakalanılan exception ekrana Error mesajı olarak basılır. Eğer çözüm varsa, bilinmeyenler (x0,x1,x2,…,xn) değerleriyle ekrana basılır.

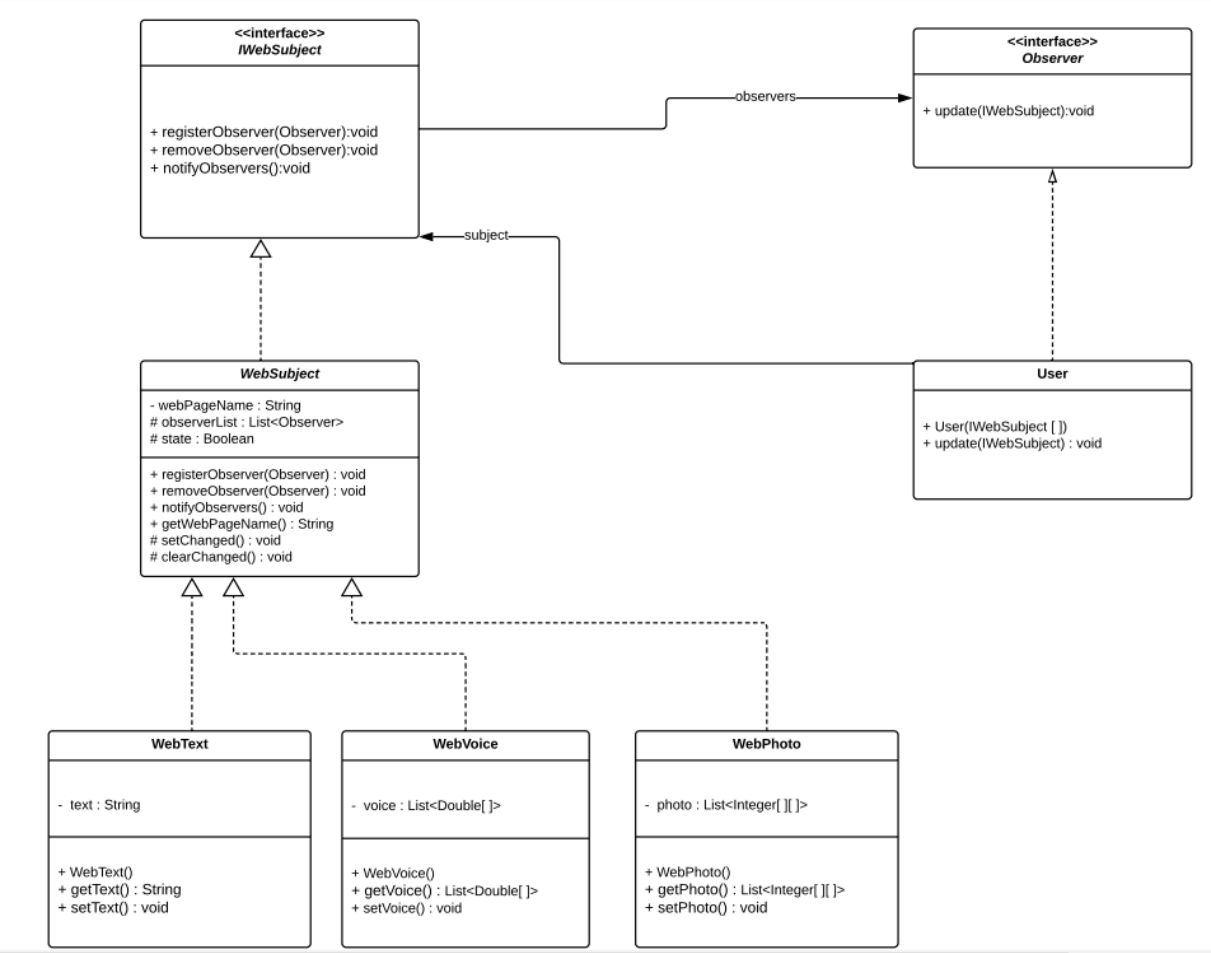


Lineer denklemlerin birden fazla çözümü olabileceğinden, hiyerarşinin başına lineer denklemler için "MethodsOfSolvingLinearEquations" adında bir interface eklendi. Bu interfaceden ise Gaussian elimination ve İnverse Method'u implement eden sınıflar türetildi. (diagramda yeşil ok burada impiment'i temsil ediyor.) Bu iki çözüm yönteminde de interfacede implement edilmemiş "solve" methodunu implement ediyor. İki çözüm yönteminin construction ları da, GUI' den girilmiş matrix 'in değerlerini String arrayi olarak alarak (Her string bir equation) kendi data field'inde saklıyor ve çözüm yaparken bu matrix' i referans alıyor.

GUI sınıfında LinearEquationSolver sınıfı yardımıyla, seçilen çözüm yöntemi kullanılıyor. Bu aşamada GUI 'den seçilen çözüm yöntemine göre run time sırasında MethodsOfSolvingLinearEquations objesi oluşturularak, LinearEquationSolver'a veriliyor ve "solv" methodu çağırılarak çözümleri (x1,x2,…,xn) geri alıyor. LinearEquationSolver, GE,MatrixInverse sınıflarının objeleri kullanılabileceğinden GUI sınıfından , bu sınıflara kesik çizgili ok var ve bu okta "<<create>>" yazısı mevcut.

"LinearEquationSolver" sınıfının constructure sinde "MethodsOfSolvingLinearEquations" sınıfının objesi alınıyor ve solv methodunda bu alınan objenin solve methodu çağırılıyor. Bu sınıf sayesinde run time sırasında kendisine gönderilen sınıfın(solving methodlardaki, Gaussian ve Matrix Inverse) tipi önemli olmaksızın objesini referans alarak onun solve methodunu çağırıyor ve linear equation'un eğer sonucu varsa sonucunu gönderiyor. Bu sınıf ile "MethodsOfSolvingLinearEquations" sınıfı arasında composition ilişkisi yer alıyor. Çünkü eğer LinearEquationSolver sınıfı yok olursa "MethodsOfSolvingLinearEquations" sınıfı kullanılamayacak. Yani lineer çözüm methodları onların solver'i olmadan bir işe yaramayacaktır.

Part2



Interface'ye göre kodlama yapmak için Subject ve Observer için interface oluşturuldu. Öncelikle Subject hiyerarşisinden bahsetmek gerekirse, interfaceyi bir abstract sınıf takip ediyor. Bu abstract sınıf(WebSubject) içerisinde web sitesinin adı, bu web sitesine abone olan obsever' ların tutulduğu bir liste ve registerObserver,removeObserver,notifyObserver methodları implement ediliyor. Böylece bu sınıftan türetilecek herhangibir subject' in bu methodları implement etmesine gerek kalmadan kendi abonelerini kaydedebiliyor,silebiliyor ve onların update'lerini çağırabiliyor.

Web sayfasında bir değişiklik olduğunda ise notify() methodu çağırılıyor. Bu methodun çalışabilmesi için abstract class' daki state değişkeninin true olması (yani bir değişikli olması anlamına geliyor) gerekiyor. Herhangibir değişiklik olduğunda alt classlar(WebText,WebVoice,WebPhoto) abstract class' da implement edilmiş setChanged methodunu çağırıyorlar böylece state değişkeni true oluyor ve notify methodu ile abonelerin update metotlarını çağırabiliyorlar. Update methodları çağırıldıktan sonra ise yine abstract classda implement edilmiş "clearChanged" methodu yardımıyla state tekrar False yapılıyor ki tekrar notify çağırıldığında herhangibir değişiklik olmadan abonelerin update methodları çağırılmasın diye.

WebText sınıfında text ‘ler String olarak tutuldu. Erişmek ve değiştirebilmek için getter,setter leri yazıldı.

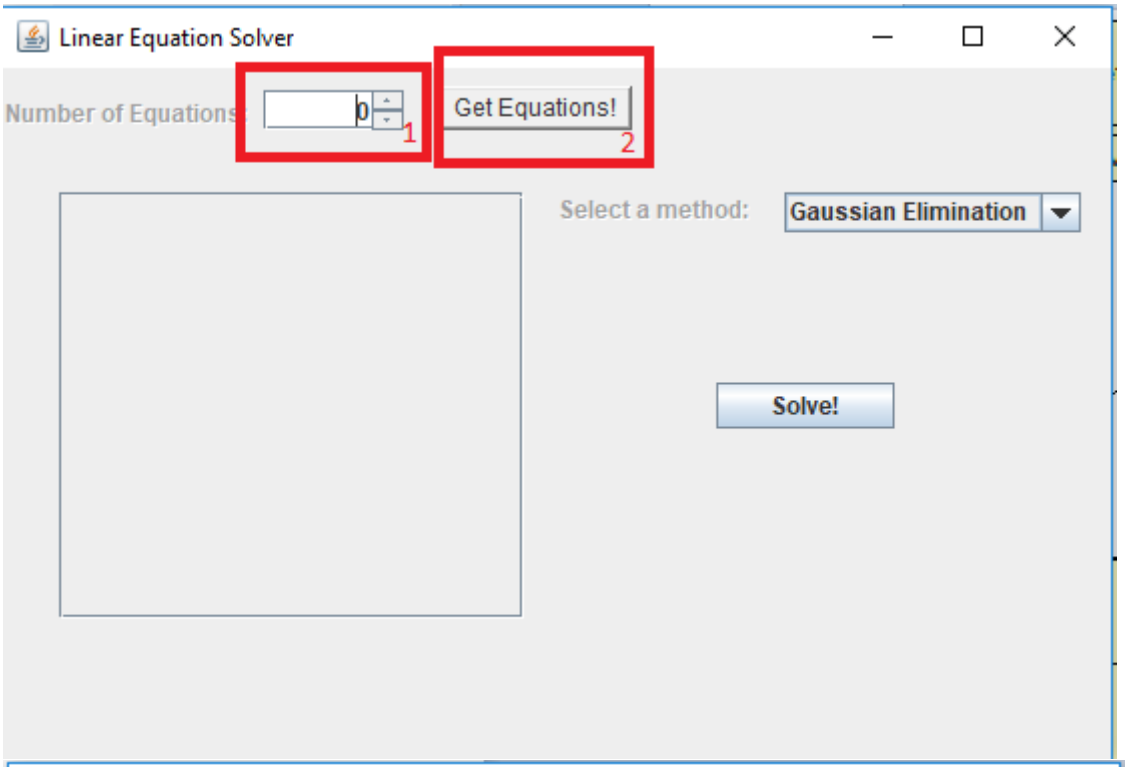
WebVoice sınıfında voice tipi olarak Double[] seçildi, bir sitede birden fazla voice olabileceği için ise sınıfın içerisinde bunların listesi tutuldu. Erişmek ve değiştirebilmek için getter,setter leri yazıldı.

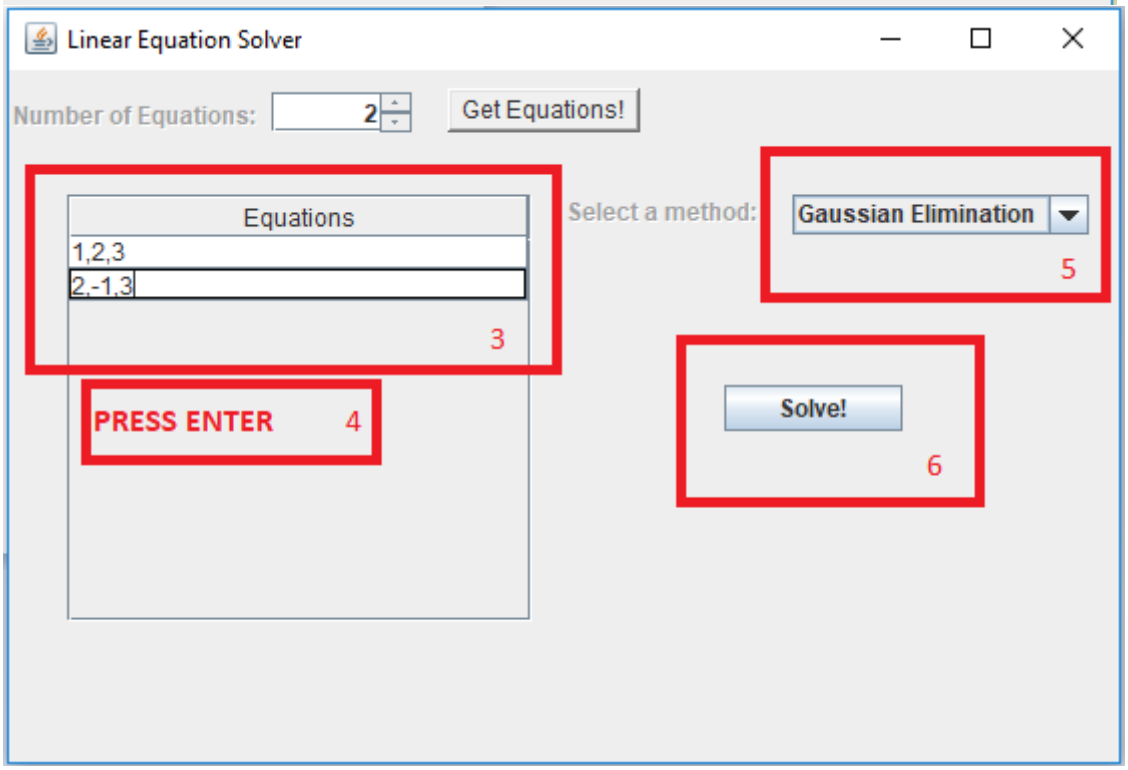
WebPhoto sınıfında bir fotoğraf ‘ı temsil etmek için tip olarak Integer[][] seçildi ve sitede birden fazla fotoğraf yer alabileceği için bunların listesi yapıldı. Erişmek ve değiştirebilmek için getter,setter leri yazıldı.

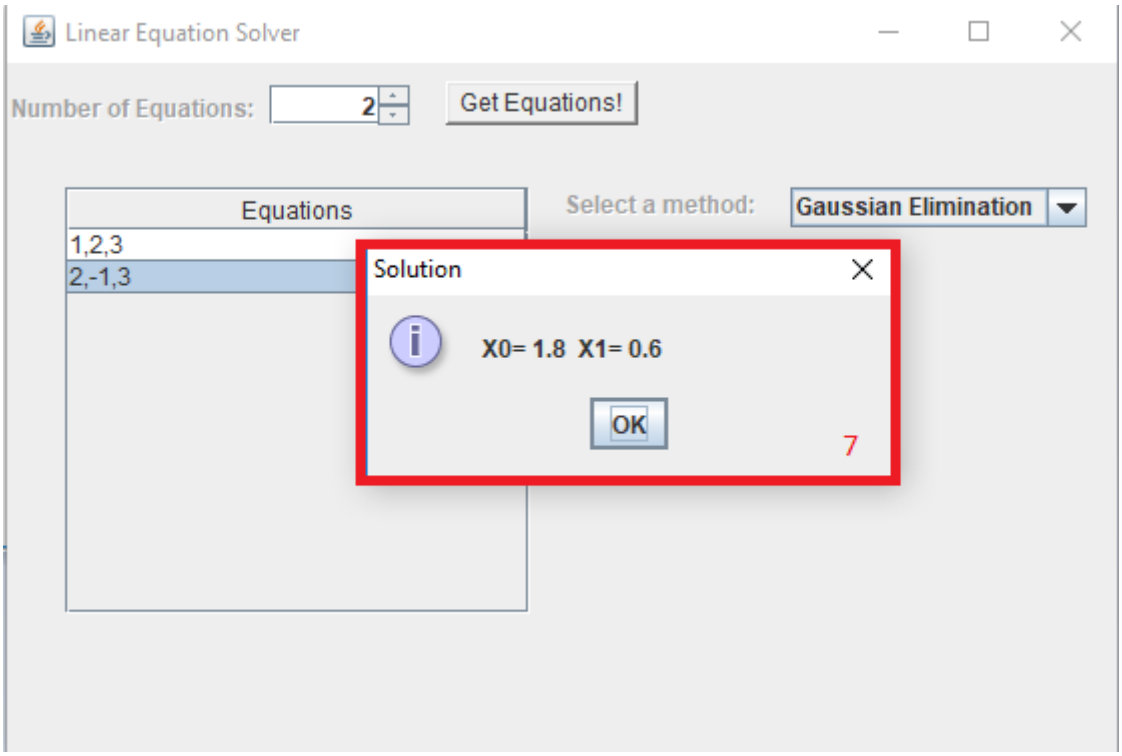
Bu sub classların her biri (WebText,WebVoice,WebPhoto) abstract sınıf(WebSubject) da ki abone listesini, state değişkenini ve abone ekleme çıkarma,uyarma ları kullanıyor. Böylece her seferinde yeniden implement edilmiyor. OOP ‘in code reuse methodolojisinden faydalanıyor. Ayrıca text,voice,photo dışında yeni bir yapı eklendiğinde ise yapılması gereken tek şey tutulacak yapının tipini belirmek ve constructor (abone listeleri için yer alacak), getter, setter lerini yazmak, diğer operasyonları abstract sınıfı extend ederek miras alacak.

Observer tarafından bahsetmek gerekirse, her observer’ ın subject te bir değişiklik olduğunda çalışması gereken bir methodunun olması lazım. Bu methodu bir interface oluşturarak içerisine “update” adıyla koyduk. Bu update methodunu, Observer web sayfasında bir değişiklik olduğu zaman parametresi ile bilgiyi push layacak ve böylece kullanıcı değişimden haberi olacak. User sınıfının constructure si Observer listesi alıyor, bunun sebebi ise bir user aynı anda birden fazla content e abone olabilir.

Nasıl Çalıştırılır?







1) Girilecek linear equation sayısı seçilir.
2) "Get equations!" butonu ile equationları girmek için bir alan getirtilir.
3) Bu alanın her satırına, virgüller ile ayırarak equation sonucu ile birlikte yazılır. Örneğin
   $X_0 + 2x_1 = 3$
   $2x_0 - x_1 = 3$
   Denklemlerinin çözümü için yukarıdaki resimdeki gibi
   1,2,3
   2,-1,3
   Şeklinde input girişi yapılmıştır.

4) Lineer denklemler yazıldıktan sonra işlemin bitmesi için denklemlerin yazıldığı alan editable olmaktan çıkmalıdır. Bunun için enter tuşuna basmak gereklidir.
5) Lineer denklemin çözüm methodu seçilir.
6) "Solve!" butonuna basılır.
7) Eğer çözüm varsa, ekrana çözüm, yoksa bir uyarı gelecektir.

Yeni bir denklem girmek içinse tekrar 1. adımdan başlanmalıdır.

Fürkan YILDIZ
141044031