PROJE 1

Egemen Çakır

Kocaeli Bilgisayar Mühendisliği
Email: cakiregemen0@gmail.com

I. GIRIS

Bu projende, belirli kurallara göre hareket eden bir robotun önündeki engelleri aşarak istenen hedefe ulaşmasını sağlayan bir oyun tasarlanması beklenmektedir. Oyunda iki adet problemin çözülmesi gerekmektedir. Problemlerin çözümü için nesneye yönelik programlama ve veri yapıları bilgilerinin kullanılması beklenmektedir.Bu projenin amaçı proje gerçekleştirimi ile öğrencilerin nesneye yönelik programlama ve veri yapıları bilgisinin pekiştirilmesi ve problem çözme becerisinin gelişimi amaçlamaktadır.Ayrıca verilen iki problem:

- Problem 1:Bu problemde sizden robotu ızgara (grid) üzerinde verilen hedefe engellere takılmadan en kısa sürede ve en kısa yoldan ulaştırmanız beklenmektedir. Robotu tüm ızgarayı değil, yalnızca gerekli yolları gezerek hedefe ulaşmasını sağlamalısınız. Adım 1: Gerekli boyutlarda karesel bir ızgara alanı oluşturmanız gerekmektedir. Adım 2: Izgara üzerine engeller ve duvarlar yerleştirilmelidir. Izgara boyutu, engel sayısı ve engellerin konum bilgileri içeriği matris biçimindeki bir text dosyasından alınacaktır. Bu text dosyasına önceden verilecek bir url adresinden (e-destek üzerinden paylaşılacaktır) uygulama çalıştırıldığında otomatik olarak erişilerek dosyadaki tasarıma göre ızgara ve engel yapısı oluşturulacaktır. Engeller birbirinden farklı tipteki nesnelerden oluşabilir. (Verilecek text dosyasındaki 0 değeri engelsiz yollara; 1, 2, 3 değerleri ise üç farklı tipteki nesne için engelleri temsil edecektir. Birbirinden farklı sayıda karesel alan işgal eden bu üç engel nesnesinden 1 değerli nesne yalnızca 1 karelik alan 2 değerine sahip nesneler yanyana 2 kare içeren maksimum 2x2 lik; 3 değerine sahip nesneler ise yan yana 3 kare içeren maksimum 3x3 lük kare alana Şekil 1' deki gibi yerleştirilecektir.) Adım 3: Robotun başlangıç ve hedef noktaları ızgara üzerindeki uygun (engel veya duvar içermeyen) karelere rastgele belirlenmelidir. Robot başlangıçta tüm ızgara dünyasını bilmemelidir, sadece bir adım sonraki kareleri görebilmelidir. Her adımda robotun öğrenmediği kareler bulutlu (kapalı) olarak gösterilmeli, öğrenilen kareler ise açılarak ilgili karelerde bulunan nesneye göre (engel, duvar, yol, vs.) belirtilmelidir. Adım 4: Tüm bu bilgiler doğrultusunda, robotun hedefe en kısa sürede ulaşabileceği en kısa yol, adım adım ızgara üzerinde gösterilmelidir. Robotun daha önce geçtiği yerler belli olacak şekilde her adımda yol üzerinde iz bırakması gerekmektedir. Hedefe ulaşıldığında ise başlangıç noktasından hedef konuma giden robota göre en kısa yol ızgara üzerinde ayrıca çizdirilmelidir. Geçen toplam süre (sn cinsinden) ve kaç kare üzerinden geçildiği bilgileri ekranda gösterilmelidir.
- Problem 2:Bu problemde sizden robotu labirentteki çıkış noktasına ulaştırmanız beklenmektedir. Adım 1: Kullanıcı tarafından istenilen boyutlarda bir ızgara oluşturmanız gerekmektedir. Adım 2: Izgara üzerine 1 nolu tipte engeller yerleştirilerek labirent oluşturulmalıdır. Labirent içerisinde mutlaka çıkışa ulaşamayan yollar bulunmalıdır. Adım 3: Labirentin giriş ve çıkış noktaları dörtgen ızgaranın herhangi çapraz 2 köşesi olarak belirlenmelidir. Robot başlangıçta labirenti bilmemelidir. Labirentte yanlış girilen bir yol algılandığında robotun doğru olarak tespit ettiği en son konuma giderek buradan itibaren yol aramaya devam etmesi gerekmektedir. Adım 4: Tüm bu bilgiler doğrultusunda, robotun çıkışa ulaşmak için izlediği yol adım adım ızgara üzerinde gösterilmelidir. Her adımda robotun daha önce geçtiği yollar üzerinde iz bırakması gerekmektedir. Robot hedefe ulaştığında giriş noktasından çıkış noktasına giden yol ızgara üzerinde çizilmelidir. Geçen toplam süre (sn cinsinden), kaç kare üzerinden geçildiği bilgileri ekranda gösterilmelidir.

Bu proje javadan dosyadan veri alma ve işleme işlemlerini geliştirme açısından önemli ayrıca bu proje veri yapılarını daha fazla aşina olmamızı ve daha iyi kavramamızı saglayıp yukarıdaki problemleri ve adımlarını gerçekleştirebilmeyi saglaması acısından önemlidir. Bu projeyi java dilinde yazdım. Birçok sınıf kullandım , sınıflarda kapsülleme , miras alma gibi birçok yöntem kullandım.Ayrıca soy agacını görsel bir şekilde gösterebilmek için swing'i ve birçok yararlı kütüphane kullandım.

II. ADIMLAR

Adım 1

İlk adım olarak bir main classı oluşturdum ve bu sınıfa JFrame i extends ettim.Daha sonra başlık için bir JLabel ekledim ve başlıgı verdi daha sonra iki adet buton koydum.Bu butonlar iki problem için ayrı ayrı pencereler açmak için kullandım.Bu butonlar için iki class açtım biri problem 1 için P1 digeri ise P2.Her iki butona action verip bastıgımda problem 1 veya problem 2 penceresine gidip main pencereyi gizledim.

Adım 2

Bu adımda gerekli olan Uygulama , Robot, Izgara ,Engel sınıflarını oluşturdum fakat ileriki adımlarda bunları dolduracagım.Bundan sonra P1 sınıfını dizayn etmeye başaldım.Pencerenin sagına adım sayısını ,süreyi ve minimüm adım sayısını gösterecegim JLabellerimi alt alta koydum ayrıca bunların altına beş buton koydum . Bu butonlardan biri main pencereye dönmek için kullanacagım , birini genel ızgara şemasını göstermek için ,birini simüle için birini ızgara degiştirmek için ve en sonda robotu adım adım hareket ettirmek için kullanacagım.Ana pencereye dönmek için kullanacagım butonun içinde main classının nesnesini oluşturdum ve P1'in setVisibil'ını false olarak ayarladım.

Adım 3

Labirent yapısını ve bu ızgaraların arasında geçiş yapmak için "yapı" butonunun action kısmını yazmaya başladım.Öncelikle url den yapının bilgilerini cekmek ve inşa etmek için iki metot yazdım.Bunlar:

- read metotu:İki parametre alıyor bunlar String ve Izgara adında degerler.Bu fonksiyon Url den verileri satır satır alıp Izgara sınıfında bulunan bir ArrayListe ekliyor.
- build metodu:Bir parametre alıyor bu deger Izgara degeridir.Bu metot iç içe for döngüleri ile read metodundan elde ettigim ArrayList degerlerini char dizisine cevirip yol ya da engel olmasına göre Izagra sınıfında bulunan engel , seem ,visited gibi özellikleri ayarlar.

Bu butona tıklanma durumuna göre bir if yapısı oluşturdum.Bu if içinde ilk önçe daha önçe P1 içinde tanımladıgım Izgara nesnesini kullanarak read ve build metotlarını kullanarak bir Izgara tipinde iki boyutlu dizi elde ettim.Daha sonra ızgaraya engelleri eklemek için Engel sınıfında 3 metot ve ayrıca EngelT1iEngelT2 ve EngelT3 adında sınıflar oluşturdum.Bunlar:

- engelAlan1 metodu:Bu metot parametre olarak bir Izgara dizisi alıyor ve bu ızgara içinde engelDegeri 1 olanları Engel sınıfında yer alan bir ArrayListe ekler.
- engelAlan2 metodu:Bu metot parametre olarak bir Izgara dizisi alıyor ve bu ızgara içinde engelDegeri 2 olanları Engel sınıfında yer alan bir ArrayListe ekler.
- engelAlan3 metodu:Bu metot parametre olarak bir Izgara dizisi alıyor ve bu ızgara içinde engelDegeri 3 olanları Engel sınıfında yer alan bir ArrayListe ekler.
- EngelT1 classı : Bu sınıfta model adındaki metot daha önçe engel sınıfından depoladıgımız ArrayListi parametre olarak alır ve bir foreach ile engelDegeri 1 olan ızgaraların engel özelligine EngelT1 nesnesi ekler ve engelOn özelligini true olarak ayarlar.
- EngelT2 classı:Bu sınıfta model adındaki metot daha önçe engel sınıfından depoladıgımız ArrayListi parametre olarak alır ve bir foreach ile engelDegeri 2 olan ızgaraların engel özelligine EngelT2 nesnesi ekler fakat burada ekleme işlemi farklı modeller için 4 farklı if yapısı ile farklı konumlarda yerleştirir.
- EngelT3 classı:Bu sınıfta model adındaki metot daha önçe engel sınıfından depoladıgımız ArrayListi parametre olarak alır ve bir foreach ile engelDegeri 3 olan ızgaraların engel özelligine EngelT3 nesnesi ekler fakat burada ekleme işlemi farklı modeller için 5 farklı if yapısı ile farklı konumlarda yerleştirir.

, bu sınıfları ve metotları kullanarak Izgara dizisindeki her bir ızgaranın genel özellikleri tanımlandı ve bir iç içe for ile ekrandaki panele bu Izgara elemanlarını ekledim ve genel bir yapı oluşturmuş oldum.Aynısını else durumu için yaptım.

Adım 4

Bu aşamada programın herhangi bir anında genel yapıyı görmek için koydugum butonun action kısmını yazmaya başladım.Burada bir if yapsı var if eger gerçekleşirse genel yapı gözükür.Bu if yapısı içinde iç içe for döngüsü var burada en iç for da eger başlangıc ve bitiş noktaları ise rengini pembe yapar degilse engelDegerine göre farklı icon ve renk verdim.Eger genel görünümü kapatmak için rengi ve ıconu null degerleri vererek gizledim.

Adım 5

Izgara yapsını oluşturudugum için artık robotun adım adım gitmesi için çalıştır adlı butonun actionı yazdım burada P1 de daha önçe tanımladıgım Timer ile her 1 sn de P1 in içindeki actionını çalıştırdım.Bu çalıştır butonunda timerı başlatmak ve durdurmak için bir koşul yapısında timer.start() else durumda timer.stop() kodları yer adldı.Bu timerın cagırdıgı action içinde ilk önçe eger daha önçe simule çalışmışsa bir iç içe for ile ızgaranın renk,visible,seem ve icon gibi özelliklerini false ya da null yaptım.Diger if koşulu ise cıkış ızgarasının visited degeri true oldugunda robotun içindeki finis özelligine ekledim daha sonra JOptionPane ile kullanıcıya cıkısa ulaştıgını ve adım sayısını gösterdim.Aynı şekilde JOptionPane ile kısa yolu isteyip istemedigini sordum eger kullanıcı YES derse ekranda kısa yol farklı bir renkle gösterilir.Bu kısa yolu ise Robot sınıfında yer alan shortestPath metodu ile buldum ve renklendirdim.Bu metod:

- shotestPath metodu:Bu metot parametre olarak bir Izgara dizisi alıyor ve içinde Queue yapısını kullandım.Bu metodun macı Breath First Search ve Dijkstra karışımı bir algoritma ile başlangıç noktasından taki bitiş noktasına ulaşana kadar bütün ızgara karelerinin başlangıç noktasına uzaklıgını hesaplıyıp en sonda bitiş noktasına varınca bir do while yapısı ile Robotun shortest adındaki harita degişinceye kadar sıfırlanmayacak özelligi olan LinkedListe ekledim.

Kısa yoldan sonra yine bir JOptionPane ile kullanıcıya baştan başlamak isteyip istemedigini sordum eger YES e basarsa Izgara yapısının bütün özlleiklerini kapattım.Bu actionın ana kısmına gelince Robot sınıfında robotun hareketini saglayacak bir metot olan move adında bir metot yazdım.Bu metot:

- move metodu:Bu metot bir Izgara dizisi ve iki tane de int deger alıyor.İçinde belli koşul ve durumlara göre yöneliyor ve her adımda o adımın cevresindeki bulutlanmayı acıyor.Öncelikle en kısa yol bulunduysa onu takip ediyor ve her her adımda bulutlanmayı acıyor.Eger en

kısa yol bulunmadıysa daha önçe belli yerleri gectiyse robotun hafızasındaki gecilenYerler adındaki ArrayListin eleman sayısına ve belli kosullara göre ilk önçe daha önçe geçilen yerler geçiliyor.Eger bu koşulda saglanmazsa 0 ile 4 arasında bir sayı üretilir ve bu sayı mevcut adımdaki bir yönü temsil eder fakat üretilen sayı yönünde engel varsa yeniden daha önçeden üretilen sayıdan farklı bir sayı üretilir ve bir yere gidilmeye çalışışır eger 4 rakam üretilmişse ve bir yere gidilmemişse cıkmaz sokak olacagından bu fonksiyonda kullandıgım stack yapısı ile bir önçeki yere geri gitter.robot her hareketinde özelligi olan geçilen yerler adlı Arrayliste ekler.

Robot her hareket ettiginde adım sayısını ve adım sayısını gösterecek JLabelı güncellemek için Uygulama sınıfını şekillendirdim.Bu sınıfta:

- Classın özellikleri:İki adet static biri Double digeri Iteger tipinde degişkenler koydum.
- Update metotu:Bu metotu overloading yaptım biri parametre olarak JLabel alırken digeri JLabel ve Integer bir deger alıyor.Fonksiyonun amacı JLabelin setText özelligini kullanarak adım sayısını degiştirmek.
- gecenSure metodu:Bu metot iki adet long deger ve bir adet JLabel alıyor.Temel olarak simule butonun action kısmında robotun hareket başladıgında alınan long bir deger ve cıkışa ulaştıgında alınan long degeri ile arada geçen süreyi hesaplayıp JLabel da göstermeye yarıyor.
- res metotu: Bu metotu overloading yaptım temel olarak static degişkenleri sıfırlamak ve JLabelleri sıfırlamak için kullandım.

Uygulama sınıfınıdaki metotları kullanarak adım adım çalışma problemini tamamladım.

Adım 6

Bu adımda simule butonunun action bölümünü yazmaya başladım.Öncelikle daha önçe çalıştır ya da simule butonlarının çalışma olasılıgına göre bütün ızgara yapısının özelliklerini default hale getirdim ve bazı düzenlemeler yaptım.Bu koşullaradan sonra bir while içinde Robot sınıfındaki move fonkisyonunu çalıştırdım ta ki cıkışa ulaşana kadar.Daha sonra kısa yolu bulup renklendirme işlemlerini yapıp kısa süre minimum adım ve toplam adım sayılarını günçelledim.En son olarakta bir text dosyasına robotun geçtigi adımlar, kısa yol adımları ,geçen süre ,toplam adım sayısı gibi degerleri yazdırdım.Böylelikle Problem 1'i çözmüş oldum.

Adım 7

Bu adımda Problem 2 için P2 sınıfını tasarlamaya başaldım.Bu sınıfın genel görüntüsü P1 sınıfınki gibi yaptım.Daha sonra main pencereye dönmek için koydugum butona action verdim.Burada main sınıfından bir nesne oluşturudm ve P2 sınıfının setVisibil ını false yaptım.Daha sonra kullanıcının istegi dogrultusunda oluşacak ızagara görünümünü için map butonuna action verdim.Öncelikle kullanıcıdan iki adet deger aldım ve bu degerlerle bir Izgara tipinde iki boyutlu bir dizi oluşturdum.Sonra burada kullanmak üzere Izgara sınıfında bazı metotlar yazdım.Bunlar:

- creat metodu:Bu metot parametre olarak bir Panel ve Izgara dizisi alıyor temel olarak Panele ızgara dizisindei elemanları engel olarak yerleştiriyor.
- Maze metodu:Bu metot ızgara dizisi ve iki adet int deger alıyor.Bu metodun yaptıgı iş ise Depth First Search ile girişten başlayıp bir labirent oluşturmak.Bunun yanında gecilen yerlerin engel özelligini kapatıyor.
- fillMaze metodu:Bu metod parametre olarak Izgara tipinde bir dizi alarak dizi içindeki elemanların engel özelliklerine atama gerçekleştiriyor.

Bu metotlarıda yazdıktan sonra bir while içinde Maze ve fillMaze metotlarını koydum bu while ın koşulu ise cıkışın yanında yol varsa bu labirent olmuş diyerek döngü bitecek.Daha sonra Engel sınıfındaki engelAlan1 ve EngelT1 sınıfındaki model metotlarını kullanıp labirenti oluşturmuş oldum.Daha sonra programın herhangi bir anında genel yapıyı görmek için ekledigim butona action verdim.Burada genel yapı açıksa kapatması kapalıysa açması için koşullar yazdım.Genel yapıyı açmak için iç içe for döngüsü kullandım ve engelOn==true ve seem==false olan yerler engel tipine göre icon ve background renkleri verdim.Genel yapıyı kapatmak için ise bu özelliklere null degeri koydum.

Adım 8

Daha sonra adım adım yürütme ve simule işlemlerini yapmak için Robot classında iki metot yazdım.Bunlar:

- mod metodu :Bu metot 4 adet int deger alır ve bunlar giriş cıkış noktalarının kordinatlarıdır.Bu kordinatların birbirine göre 4 farklı moddan birini Robotun mod özelligini degiştirir.
- move2 metodu:Öncelikle kısa yol bulunduysa kısa yol bilgisine göre gider eger bu bilgi yoksa daha önçe geçtigi yerlere öncelik verir ve buraları gezer fakat bu iki koşul gerçekleştirilmezse oluşan moda göre switch yapısına girer ve öncelik verilen yönde yol var mı ona bakar eger varsa gider ve cevresindeki bulutları açar.Fakat cevresi hep engellerle kaplıysa stack yapısını kullanarak en son konumuna gider.

Bu metotları yazdıktan sonra simule butonunun actionını yazmaya başladım.Öneçelikle daha önçe calıştır butonuna basılmış olma durumu ve ya simule edilmiş durumuna karşı iç içe for ile ızgara elemanlarının özelliklerini default hale getirdim ve bazı işlemler yaptım.Daha sonra while içine cıkışa ulaşana kadar robot.move2() metodunu çalıştırdım.Bu whilden önçe o anki zamanı aldım daha sonra kısa yolu bulup renklendirdikten sonra tekrar zamanı alıp bunları Uygulama sınıfındaki metot ile ekranda gösterdim.En son işlem olarak bir txt dosyasına robotum gezdigi tüm yolları,kısa yolu,minimum adım sayısını,toplam adım sayısını ve süreyi yazdırıdm.

Adım 9

Bu adımda robotun adım adım gitmesi için çalıştır adlı butonun actionı yazdım burada P2 de daha önçe tanımladıgım Timer ile her 1 sn de P1 in içindeki actionını çalıştırdım.Bu çalıştır butonunda timerı başlatmak ve durdurmak için bir koşul yapısında timer.start() else durumda timer.stop() kodları

yer adldı.Bu timerın cagırdıgı action içinde ilk önçe eger daha önçe simule çalışmışsa bir iç içe for ile ızgaranın renk,visible,seem ve icon gibi özelliklerini false ya da null yaptım.Diger if koşulu ise cıkış ızgarasının visited degeri true oldugunda robotun içindeki finis özelligine ekledim daha sonra JOptionPane ile kullanıcıya cıkısa ulaştıgını ve adım sayısını gösterdim.Aynı şekilde JOptionPane ile kısa yolu isteyip istemedigini sordum eger kullanıcı YES derse ekranda kısa yol farklı bir renkle gösterilir.Bu kısa yolu ise Robot sınıfında yer alan shortestPath metodu ile buldum ve renklendirdim.Kısa yoldan sonra yine bir JOptionPane ile kullanıcıya baştan başlamak isteyip istemedigini sordum eger YES e basarsa Izgara yapısının bütün özlleiklerini kapattım.Bu actionın ana kısmına gelince Robot sınıfında yazdıgım move2() metodunu kullandım ve en sonunda robotun adımlarını adım adım göstermek için Uygulama sınıfındaki metotları kullanıp hem bu adımı hemde genel olarak Projeyi bitirmiş oldum.

III. Özet

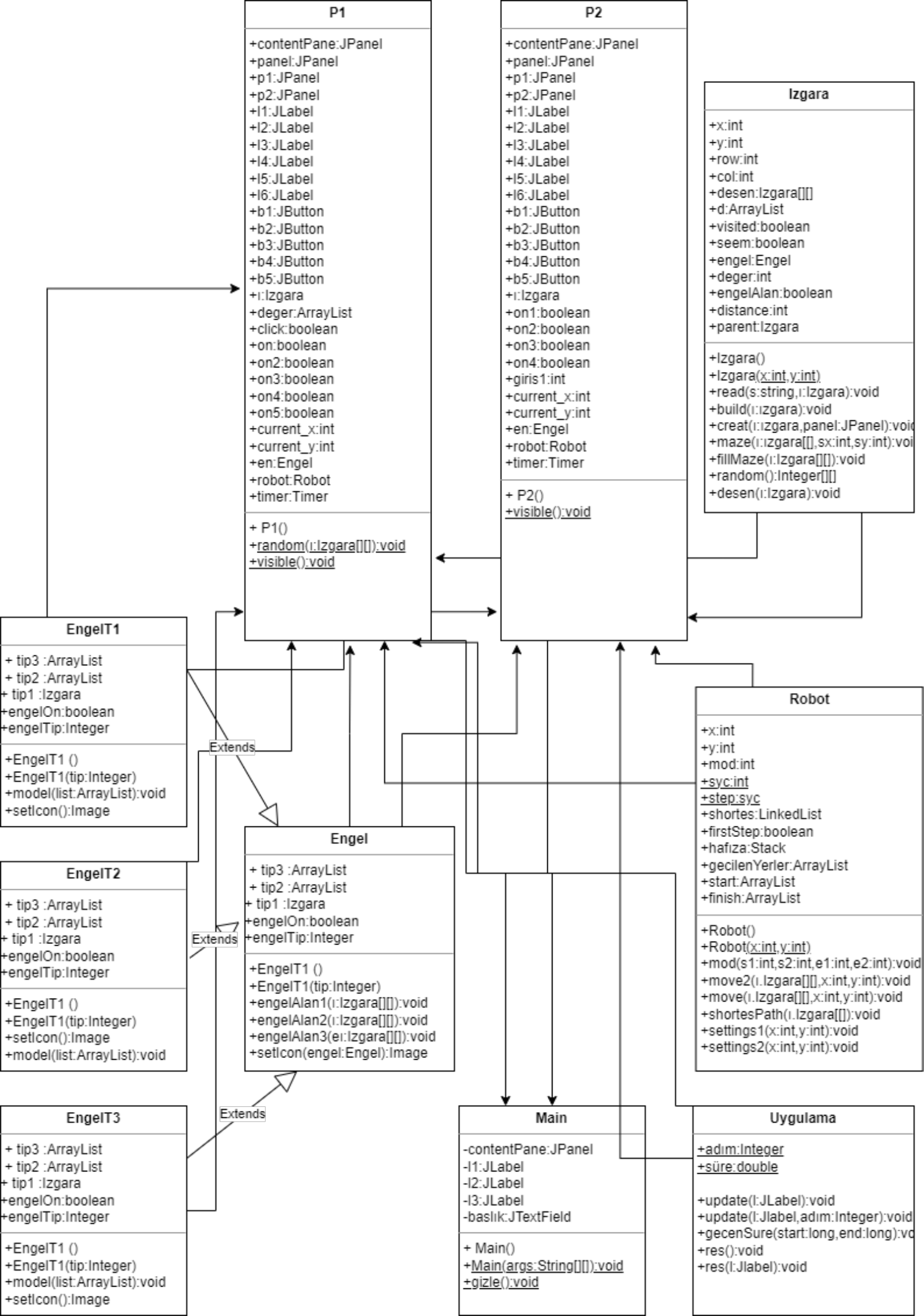
Bu projenin amacı iki adet labirent problemini çözmeye çalıştım.Bunları yaparken veri yapılarını daha iyi benimsedim ve OOP yapısına daha da aşina oldum.Ayrıca verileri dosyaya kaydetmek ve bir URL den veri çekerken Java için dosya işlemlerimi geliştirdim.

IV. Sonuç

Tüm kontroller yapıldıktan sonra artık problem 1 için url'lerden verileri çekip labirent yapısını oluşturabiliyor ve robotu adım adım yürütüp simule ile de sonucu gösterebiliyorum.Problem 2 içinde kullanıcıdan aldıgım bilgiler ile bir labirent yapısı oluşturabiliyor aynı şekilde robotu yürütüp simule ile sonucu gösterebiliyorum ve her iki problemde de her labirentin sonucunu bir txt dosyasına yazabiliyorum.

V. Kaynakca

- https://www.javatpoint.com/java-get-data-from-url
- https://www.javatpoint.com/java-swing
- https://www.w3schools.com/java/default.asp
- https://www.javatpoint.com/java-oops-concepts


P1
+contentPane:JPanel
+panel:JPanel
+p1:JPanel
+p2:JPanel
+l1:JLabel
+l2:JLabel
+l3:JLabel
+l4:JLabel
+l5:JLabel
+l6:JLabel
+b1:JButton
+b2:JButton
+b3:JButton
+b4:JButton
+b5:JButton
+ı:Izgara
+deger:ArrayList
+click:boolean
+on:boolean
+on2:boolean
+on3:boolean
+on4:boolean
+on5:boolean
+current_x:int
+current_y:int
+en:Engel
+robot:Robot
+timer:Timer
+ P1()
+random(ı:Izgara[][]):void
+visible():void
P2
+contentPane:JPanel
+panel:JPanel
+p1:JPanel
+p2:JPanel
+l1:JLabel
+l2:JLabel
+l3:JLabel
+l4:JLabel
+l5:JLabel
+l6:JLabel
+b1:JButton
+b2:JButton
+b3:JButton
+b4:JButton
+b5:JButton
+ı:Izgara
+on1:boolean
+on2:boolean
+on3:boolean
+on4:boolean
+giris1:int
+current_x:int
+current_y:int
+en:Engel
+robot:Robot
+timer:Timer
+ P2()
+visible():void
Izgara
+x:int
+y:int
+row:int
+col:int
+desen:Izgara[][]
+d:ArrayList
+visited:boolean
+seem:boolean
+engel:Engel
+deger:int
+engelAlan:boolean
+distance:int
+parent:Izgara
+Izgara()
+Izgara(x:int,y:int)
+read(s:string,ı:Izgara):void
+build(ı:Izgara):void
+creat(ı:Izgara,panel:JPanel):voi
+maze(ı:Izgara[[],sx:int,sy:int):voi
+fillMaze(ı:Izgara[][]):void
+random():Integer[][]
+desen(ı:Izgara):void
EngelT1
+ tip3 :ArrayList
+ tip2 :ArrayList
+ tip1 :Izgara
+engelOn:boolean
+engelTip:Integer
+EngelT1 ()
+EngelT1(tip:Integer)
+model(list:ArrayList):void
+setIcon():Image
Extends
Engel
+ tip3 :ArrayList
+ tip2 :ArrayList
+ tip1 :Izgara
+engelOn:boolean
+engelTip:Integer
+EngelT1 ()
+EngelT1(tip:Integer)
+engelAlan1(ı:Izgara[][]):void
+engelAlan2(ı:Izgara[][]):void
+engelAlan3(eı:Izgara[][]):void
+setIcon(engel:Engel):Image
Robot
+x:int
+y:int
+mod:int
+syc:int
+step:syc
+shortes:LinkedList
+firstStep:boolean
+hafıza:Stack
+gecilenYerler:ArrayList
+start:ArrayList
+finish:ArrayList
+Robot()
+Robot(x:int,y:int)
+mod(s1:int,s2:int,e1:int,e2:int):void
+move2(ı.Izgara[][],x:int,y:int):void
+move(ı.Izgara[][],x:int,y:int):void
+shortesPath(ı.Izgara[[]):void
+settings1(x:int,y:int):void
+settings2(x:int,y:int):void
EngelT2
+ tip3 :ArrayList
+ tip2 :ArrayList
+ tip1 :Izgara
+engelOn:boolean
+engelTip:Integer
+EngelT1 ()
+EngelT1(tip:Integer)
+setIcon():Image
+model(list:ArrayList):void
Extends
EngelT3
+ tip3 :ArrayList
+ tip2 :ArrayList
+ tip1 :Izgara
+engelOn:boolean
+engelTip:Integer
+EngelT1 ()
+EngelT1(tip:Integer)
+model(list:ArrayList):void
+setIcon():Image
Extends
Main
-contentPane:JPanel
-l1:JLabel
-l2:JLabel
-l3:JLabel
-baslık:JTextField
+ Main()
+Main(args:String[][]):void
+gizle():void
Uygulama
+adım:Integer
+süre:double
+update(l:JLabel):void
+update(l:Jlabel,adım:Integer):void
+gecenSure(start:long,end:long):vo
+res():void
+res(l:Jlabel):void