Gezgin Robot Projesi

Umut YİĞİT
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi 2023
200201041

Betül BODUR
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi 2023
210201069

I. Özet

Bu dokümanda Programlama Laboratuvarı II dersimizin 1. projesi olan Gezgin Robot Projesi için oluşturulan algoritmalar açıklanmıştır . Dokümanda problemin tanımı verilip bu problemler hakkında araştırmalar ve yöntemler sunulmuştur.Araştırmalar ve yöntemlerde bahsedilen class ve metotların ayrıntılı açıklaması bulunmaktadır.

***Index Terms*—nesne,arraylist,en kısa yol,algoritma**

II. Problem Tanimi

Bu projede 2 problem ele alınmıştır.1. problem de bizimle paylaşılan URL üzerinden labirentte bulunan 3 çeşit engelin bulunduğu ve engelin bulunmadığı yolların bilgisini almamız beklenmektedir.Aldığımız bilgiler ile labirent oluşturmamız beklenmektedir.Robotun başlangıç ve hedef noktası rastgele bir şekilde oluşturulmalıdır.Oluşturulan labirent bir robot tarafından çözülmesi beklenmektedir.Robot labirent ile ilgili sadece bulunduğu konumun komşu kenarları hakkında bilgisi vardır.Robot labirenti çözdüğünde arayüz üzerinde en kısa yol gözükmesi istenmektedir.2.problemde kullanıcıdan alınacak ızgara boyutu ile rastgele bir labirent oluşturulacaktır.Başlangıç ve hedef noktası çapraz 2 köşesinde olmalıdır.1. problemde olduğu gibi robot labirent hakkında sadece bulunduğu konumun komşu kenarlarını bilmektedir.Robot labirenti çözdüğünde en kısa yolun arayüzde gözükmesi beklenmektedir.Bu proje java programlama dili ile yazılmıştır

III. Araştirmalar ve Yöntemler

Bu projede öncelikle olayı somut bir şekilde görebilmek için URL adresindeki veriler okunmuştur.Ve URL adresindeki bulunan bilgiler sayesinde oluşturulan labirent için çözüm olgoritmaları oluşturulmuştur.Oluşturulan bu algortimaların başarısından emin olunduktan sonra isterlerde belirtilen hızlandırma gibi bir çok arayüzü güzelleştirecek işlevlerin koda entegresi sağlanmıştır.Bu isterleri gerçekleştirmek için kullanıcıya kolaylık sağlamak adına buton eklenmiştir.Ancak buton arayüz üzerinde gözükmemiştir.Bu sorunu çözmek için **dispose() metodu**nu kaldırınca sorunumuz çözülmüştür.

IV. Geliştirme Ortami

- Bu proje JAVA programlama dili ile yazılmıştır.
- NetBeans 15 kullanıldı.
- JDK Sürümü: 19

V. Kod Bilgisi

A. Class'lar

Programın sınıfları şu şekilde özetlenmiştir:

Uygulama Sınıfı:

Ana class'tır.GameFrame sınıfından frame nesnesini oluşturmaktadır.

GameFrame Sınıfı:

- GameFrame metodu

Izgara sınıfından izgara adlı nesne oluşturulmuştur ve bu nesne Izgara sınıfında bulunan String tipinde bir parametresi bulunan constructer'ı çağırılmıştır.Parametresi ise URL'dir.Programın başladığına ve nasıl kullanılacağına dair kullanıcıyı bilgilendirmek için JOptionPane sınıfının **showMessageDialog metodu** kullanılmıştır.
Map Değiştir adlı mevcut labirenti değiştirmek için kullanılan butonumuz bu sınıfta tanımlanmıştır .Bu butona basıldığı an yapılması gerekenler Action Listener ile belirtilmiştir.Bu butona basıldığında mevcut ızgara, frame'den kaldırılır.Başlangıçta belirlenen integer tipindeki key değişkenimizin değeri 1'dir.Eğer key değerimiz 1 ise Izgara tipindeki nesnemizin parametresi URL1 olmaktadır ve butona her basıldığında değişmesi için key değerimiz 2 ile değiştirilip frame'e yeni labirent eklenmektedir..Eğer ki key değerimiz 2 ise Izgara tipindeki nesnemizin parametresi URL2 olmaktadır ve butona her basıldığında değişmesi için key değerimiz 1 ile değiştirilip frame'e yeni labirent eklenmektedir.Oluşabilecek herhangi **IOException hatası**

için try-catch yapısı kullanılmıştır.
Bir önceki labirent ile sonra oluşacak labirentin üst üste binmemesi için **revalidate** ve **repaint** metotları kullanılmıştır.

Izgara Sınıfı:

Bu sınıfta; URL dosyası okunması,robotun labirentte yol bulması, labirentin duvarlarının oluşması, kısa yol algoritması,oyun içinde bulanan ve akışın değişmesini sağlayacak butonların oluşturulduğu metotlar bulunur.
visited,emptyWays,emptyWays2,emptyWays3, shortestPath,wallList arraylistleri ve bir çok değişken aşağıdaki açıklanacak metotlarda kullanılmak için oluşturulmuştur.

- Izgara metodu

Izgara methodu bir parametre almaktadır.Bu parametre labirentin bulunduğu URL adresidir.Parametre olarak gelen URL adresi string tipindeki url adlı değişkene atanmıştır.Sonraki metotlarda kullanılmak üzere boolean değerinde problem1 ve problem2 değişkenlerine sırasıyla, true ve false değeri atanmıştır. Izgara sınıfında oluşturulan arraylistlerin üzerine veri girişi olmaması için **clear() metodu** ile arralistlerdeki nesneler temizlenmiştir.URL'den bilgi alınması için **dosyaOku() metodu** url adlı değişkeni parametre alarak çağırılmıştır.Ve sırasıyla **setWalls(), createWalls(),setStartFinish(),createPath() ve setButton() metotları** çağrılmıştır.

- paintComponent metodu

Bu metot ile arayüzdeki animasyonlar yapılmaktadır.Boolean tipinde olan paintKey değişkeni true ise **drawPath() metodu** çağrılmaktadır.Ve sornasında **drawMaze() metodu** çağrılmaktadır.

- showFinalScene metodu

Bu metot labirentin son haline vakit kaybetmeden gelmesi için tasarlanmıştır.Sayaç durdurulup bir while içinde sayaca bağlı olmadan **drawPath() metodu** çağrılmaktadır.

- setSideWinder metodu

2.problemi oluşturmak için kullanılır.Kullanıcdan alınacak olan boyut bilgisini kullanmaktadır.Boyut bilgisi alınan labirentteki tüm kareler 1 numaralı engel olarak tanımlanmıştır.İç içe for döngüsü kullanılarak satır sayısı 2 sutun sayısı 1 arttırılarak random bir şekilde boş yollar açılmıştır.karar adındaki random sayı üreten integer türündeki değişkenimiz 1 ise boş yol oluşturulur ve y değeri bir arttırılıp engel koyulmaktadır.0 ise engel koyulmaktadır.**sonKontrol() metodu** çağrılmaktadır.

- sonKontrol metodu

setSideWinder() metodunda oluşturlan labirentin, çözümü olmama durumu göz önüne alınarak bu durum engellenmiştir.

- shortestPathProblem2 metodu

Bu metotta en kısa yol algoritması oluşturulmuştur.Öncellikle matrisi basitleştirmek için matristeki 2(2 tipindeki engel),3(3 tipindeki engel) ve 5(dış duvarlar) değerlerine sahip değerleri 1 ile değiştiriyoruz.Eğer matriste bulunan değer 1 değilse onları ise 0 değeri ile değiştiriyoruz.Problem2 mi yoksa Problem1 mi olduğunu anlamak için if-else if yapısı kurulmuştur.Eğer Problem2 ise başlangıç x ve y değeri 1 olarak atanmıştır.Eğer ki Problem1 ise robotun gittiği yollar üzerinden en kısa yolu bulacağımız için robotun öğrenmediği yolların değerlerini 1 olarak değiştiriyoruz.
visited arraylistinde 7(bitiş noktası id) bulunana kadar tüm matris gezilmektedir.Gezilen her bir konumun her bir komşusu için değerinin 0 mı olduğu kontrol edilir.Eğer ki konumun değeri 0 ve 1 değilse ve değeri sira değişkenine eşit değilse o konumun komşu değerleri kontrol edilir.Eğer konumun komşusunun değeri 0 ise konumun komşusunun değeri sira değişkenini alır ve visited arraylistesine id numarası eklenir.Matris her gezildiğinde sira değeri 1 arttırılır.Bu sayede matris üzerinde kısa yolun hangi sırayla gidilebileceği gözükecektir.

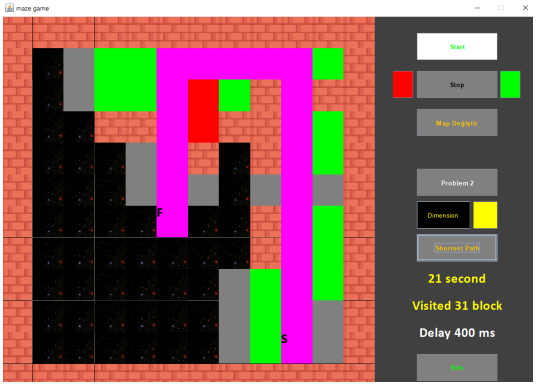

Fig. 1.

- fillShortList metodu

shortestPath arraylistini burada oluşturuyoruz.shortestPath arraylistinde herhangi bir veri olmasına karşı **clear() metoduyla** arraylist temizlenmiştir.Başlangıç konumu shortestPath arraylist'i içerisinde değilse ve Bitiş noktasının x ve y değerleri 0'dan büyük olduğu sürece eğer wallList'teki değerin sıra değeri 0 değilse ve başlangıç değeri 0 olarak atanan integer sira adlı değişkenden küçük ise kisaYol değişkenini 1 yapıp sira değişkenini wallList'teki değerin sıra değeriyle değiştirmektedir. Aynı şekilde engelin

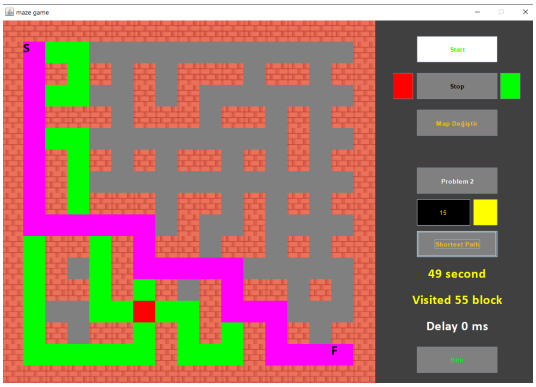

Fig. 2.

sağ,sol,yukarısı ve aşağısındaki değerleri sira değerleriyle kıyaslamaktadır.Her kenar için kısaYol değişkeni 1,2,3 ve 4 değeri almaktadır.Eğer kısaYol değişkeninde herhangi bir değişme olmazsa Yani bulunan konumun 4 komşu kenarını birbirleri arasında kıyaslamaktadır en az maliyeti olan komşu kenar shortestPath arraylistine eklenmektedir.

- drawMaze metodu

Arayüzdeki engelleri ve yolları belirtmek için tasarlanmıştır.for döngüsü içinde **switch-case** yapısı kullanılarak engellere,boş yollara, vb. atanan id numarasına göre renklendirme yapılmıştır.Arayüzde sadece robotun ilerlediği yolu görebilmek için engelin goster değeri 0 olmalıdır ve engelin id numarası 5(dış duvarlar),6(başlangıç noktası) ve 7(bitiş noktası) olmayan yerler siyah renk yapılmaktadır.Ve robotun gittiği yolların açıldığını gösteren bir animasyon olması için engelin robot değeri 1 ise turuncu renk olmaktadır.Labirentteki ızgaraların konumları bellidir ancak arayüzde gözükmemektedir.Engellerin x ve y değerlerini kullanarak 50-50 boyutunda kareler yerleştirilmiştir.Eğer ki engelin id değeri 6(başlangıç id) ise o x ve y değerlerine drawString() metoduyla isimlendirme yapılmıştır.Aynı şekilde id değeri 7(bitiş noktası id) olan x ve y değerlerine de isimlendirme yapılmıştır.

- drawPath metodu

Başlangıç değeri 1 olarak belirlenen tmp değişkeni visited.size()-1'den küçük ise visited arraylistinde bulunan tmp. indexte bulunan değeri çağırıyoruz ve wallList arrayListin'de, gelen değerinci indexteki değeri Engel tipindeki duvar değişkenine atanmıştır.Eğer duvar.id 0'a eşitse duvar.id 8'e eşitlenmektedir.duvar.id'si 8 olan kareler yeşil renge boyanmaktadır.Eğer ki duvar.id 0'a eşit değilse duvar.id 9'a eşitlenmektedir.Robotun bulunduğu konumun sağ,sol,üst ve alt kısımları wallList.goster değerlerine 1 değeri atanmıştır. Eğer ki visited.size() değeri, tmp'ye eşit ise sayaç durdurulmaktadır ve shortPath butonunun görünürlüğü aktif edilmektedir.Bu metot her çağrıldığında tmp değeri 1 arttırılmaktadır.

- actionPerformed metodu

Bu metotta start tuşuna basınca başlayan sayaçın değerleri burada arttırılmaktadır.

- createWalls metodu

Engel sınıfı tipinde duvar adlı nesne oluşturulmuştur.Engel sınıfının parametreli constructer'ı ile oluşturulmuştur.Değişken olarak **j * ((700/satir)), i * ((700/sutun))** almıştır. Bu sayede matristeki her bir değerin x ve y koordinatları belirlenmiştir.Her bir duvarın değeri duvar.id'ye atanmıştır ve her duvar nesnesi wallList arraylistine eklenmiştir.

- createPath metodu

Bitiş noktasına ulaşması için bitiş noktasının matristeki değeri 0 yapılmıştır.Başlangıç noktasını visited arraylistine eklenmiştir.Bitiş noktası visited arraylistine eklenene kadar aşağıdaki işlemler devam etmektedir.
emptyWays arraylisti clear() metodu ile temizlenmiştir.Son konumun değeri visited arraylistinde bulunana kadar, bulunan konumun komşu kenarlarında boş yol olup olmadığını kontrol eder eğer varsa emptyWays arraylistine atar.Eğer bulunan konumun komşu kenarlarında bdaha önce bir kez geçtiği yol olup olmadığını kontrol eder eğer varsa emptyWays2 arraylistine atar. Eğer bulunan konumun komşu kenarlarında daha önce çok kez geçtiği yol olup olmadığını kontrol eder eğer varsa emptyWays3 arraylistine atar.Eğer emptyWays,emptyWays2,emptyWays3 arraylistleri boş ise başlangıç değeri yeniden oluşturuluyor ,boş değilse hiç gidilmeyen komşu veya bir kere gidilen komşu yol tercih edilmektedir.Eğer konumun ziyaret sayısı 0 ise 1 arttırılıp visited arraylistine eklenir ve konumun değeri 8(yeşil yollar veya 1 kere ziyaret edilen yollar) yapmaktadır.Eğer ki konumun ziyaret sayısı 11 den küçük ise,bu şart ile bir onumu maximum 10 kere ziyaret etmesini ve çözümün yavaşlamamasını sağlanmıştır, ziyaret sayısı 1 arttırılıp visited arraylistine eklenmiştir ve konumun değeri -1(birden çok kez ziyaret edilen konum,kırmızı yollar) yapmaktadır.Eğer konumun komşu konumunda id'si 7 (bitiş noktası id'si) olan bir kenar varsa direkt olarak visited arraylistesine eklenir.

- setStartFinish metodu

Projede bizden random başlangıç ve bitiş noktası istenmiştir.Bunun için Random sınıfından bir nesne türetilmiştir.Random sayı aralığı 0 ile satir*sutun

arasındadır.İnteger tipinde oluşturulan bas ve son adlı değişkenlere random sayılar atanmıştır.Ancak herhangi bir boş yol olmaması için başlangıç ve bitiş noktası kontrol edilmektedir.Eğer id değerleri 0 veya 4 ise tekrardan random sayı oluşturulmaktadır.Başlangıç ve bitiş noktalarının x ve y koordinatları bulunmaktadır.Başlangıç ve bitiş noktalarını belirtmek için sırasıyla id numaraları 6 ve 7 olarak atanmıştır.

- setWalls metodu

Bu metotta engeller oluşturulmuştur.Random bir şekilde duvar oluşturmak için integer tipinde key3 ve key2 değişkenleri oluşturulup ilk değerleri 1 olarak atanmıştır.

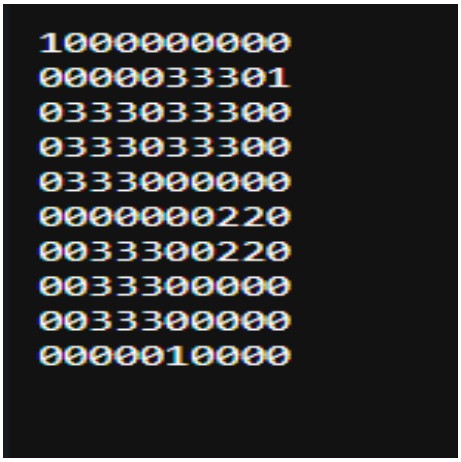

Fig. 3.

Matris değeri 3 ise ve komşu 4 kenarının değerleri toplamı 12 ise ve key3 değeri 1 ise bazı kenarları boş yol(boş yol değeri 0'dır.) yapmaktadır ve key3 değerini 0 yapmaktadır.Aynı şekilde key3 değeri 0 olduğunda da key3'ün 1 olması durumundan farklı yolları boş yol yapmaktadır.Bunlar sonucunda 3.Engel tipi oluşturulmuştur.Matris değeri 2 ise ve komşu 3 kenarının değerleri toplamı 6 ise ve key2 değeri 1 ise bazı kenarları boş yol yapmaktadır ve key2 değerini 0 yapmaktadır.Aynı şekilde key2 değeri 0 olduğunda da key2'ün 1 olması durumundan farklı yolları boş yol yapmaktadır.Bunlar sonucunda 2.Engel tipi oluşturulmuştur.

- dosyaOku metodu

dosyaOku() metodu URL'deki labirent bilgisini almamızı sağlamaktadır.Bu bir dosya işlemi olduğundan throws ile FileNotFoundException veya IOException hataları kontrol altına alınmıştır.Bu metot URL adresini içeren url adında bir parametre almaktadır.Java'da bulunan URL sınıfından txt_url adında url'yi parametre olarak alan bir nesne oluşturduk.Bu url adresini okunması için **BufferedReader** metodu kullanılmıştır.Url adresinin okunması için elimizde adresin sunucuyla bağlantı kurması gerekmektedir(TCP) bunun için **openStream()** metodu kullanılmıştır.Bu bağlantıdan gelecek olan veriyi okumamız için **InputStreamReader()** metodu kullanılmıştır.Gelen verilerin hepsi BufferedReader nesnesinden oluşturulmuş br adındaki değişkende tutulmaktadır.URL'den alınan veriyi bir matrise satır satır aktarılmıştır.Matrisin boyutu 2 arttırılarak dış duvarlar içinde yer açılmıştır ve dış duvarların değeri 5 olarak atanmıştır.

- smileyFace metodu

Arayüzde problemler arasında geçiş efekti oluşturulmuştur.

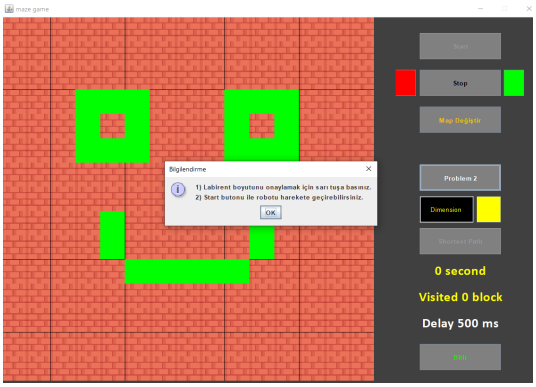

Fig. 4.

- setButton metodu

Bu metotta butonlar oluşturulmuştur ve her bi butona basıldığında hangi işlemlerin yapılacağı belirlenmiştir.JButton tipinde butonlar oluşturulmuştur.JTextField tipinde ise kullanıcıdan bilgi almak nesne oluşturulmuştur.Butonlara tıklandığı anda gerçekleşmesi istenilen işlemleri belirtmek için ActionListener oluşturulmuştur.
"Shortest Path" adlı butona tıklanıldığında sırasıyla **shortestPathProblem2(),fillShortList()** metodu çağrılmaktadır.**fillShortList()**metodunun çağrılmasıyla shortestPath arraylisti doldurulmaktadır. Doldurulan shortestPath arraylistindeki değerlerin id değeri -3 olarak atanmaktadır.
"Problem2" adlı buton oluşturulmuştur.Bu butona tıklandığında Start adlı butonun tıklanma özelliği devre dışı bırakılmıştır ve Timer tipinde oluşturulan timer nesnesi durdurulmaktadır.Ve wallList arraylistinde bulunan engellerin id değerleri 0'a eşitlenmiştir.**smileyFace()** metodu çağrılmaktadır.Problem 2'de istenildiği gibi kullanıcıdan boyut bilgisi alınması için bir TextField ve alınan bilginin onaylanması için JButton görünürlüğü ve tıklanabilirliği aktifleştirilmiştir.

”Problem2” adlı butona basıldığında çıkan TextField içine yazılan değeri onaylamak için sarı renk buton oluşturulmuştur.Bu onay butonuna basıldığında yazılan değer değişkenlere atanır.visited arraylisti temizlenir ve sırasıyla **setSideWinder(), createPath()** metotları çağrılmaktadır.
Arayüzde bulunan kırmızı buton robotun yavaşlamasını sağlamaktadır.Butona basıldığında robot 100 ms yavaşlamaktadır.
Arayüzde bulunan yeşil buton robotun hızlanmasını sağlamaktadır.Butona basıldığında robot 100 ms hızlandırmaktadır.Robotun hızı standart 500 ms olarak başlamaktadır. 5 kere bu butona basıldığında robot bitiş noktasına ulaşmaktadır.
”Start” adlı butona basıldığında timer başlamaktadır.”Stop” adlı butona basıldığında timer durdurulmaktadır.
”Bitir” adlı butonuna tıklandığında labirentin son aşamasına vakit kaybetmeden gelmesi sağlanmaktadır.Bunun için butona tıklandığında yukarıda bahsedilen **showFinalScene()** metodu çağrılmaktadır.

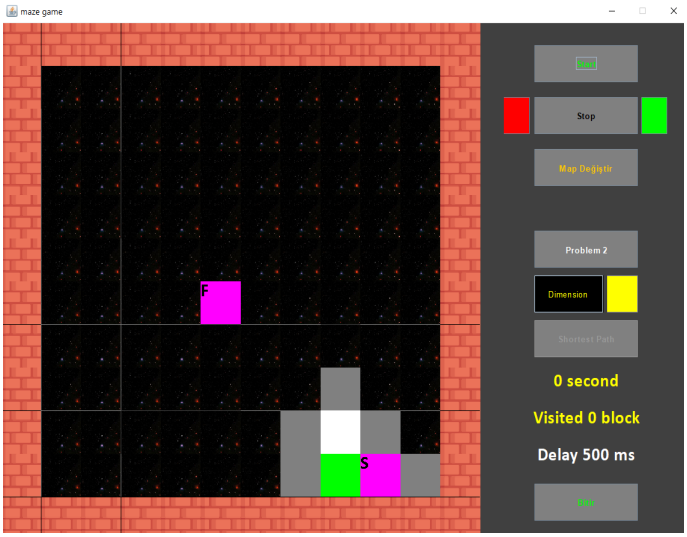

Fig. 5.

Engel Sınıfı:

Engellerin x ve y konumları ,id’leri ziyaret sayısı,en kısa yol algoritması için sira değişkeni, engellerin görünebilirliği için goster değişkeni bulunmaktadır.

- Engel metodu

Engelin x ve y değerlerini tutabilmek için parametreli constructer’dır.

Robot Sınıfı:

- Robot metodu

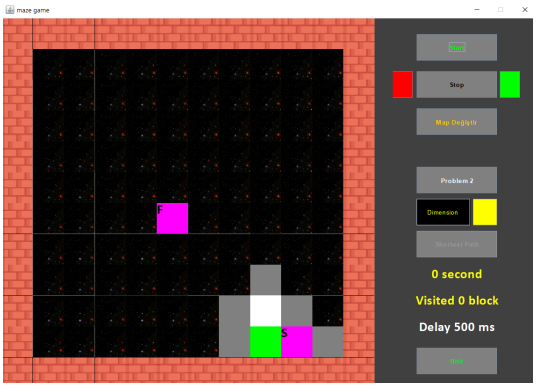

Fig. 6.

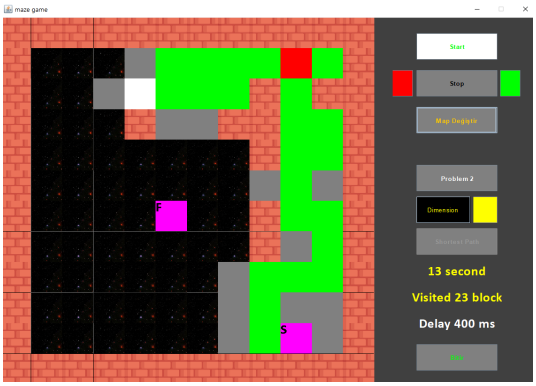

Fig. 7.

B. UML DİYAGRAMI

Kısım ektedir.[fig.9]

VI. İSTATISTIK

Kullanılan class’lar ve kullanım amaçları:

java.util
ArrayList oluşturmak,kullanıcıdan veri almak için kullanılmıştır.

java.awt
Yazı tipi, rengi gibi olayları düzenlemek için kullanılmıştır.

javax.swing
Buton, görüntü , çerçeve,.. gibi arayüzü oluşturmak için gerekli nesneleri oluşturmak için kullanılmıştır.

java.net
URL adresindeki veriye erişmek için kullanılmıştır.

java.io
URL’de bulan veriyi okumak için kullanılmıştır..

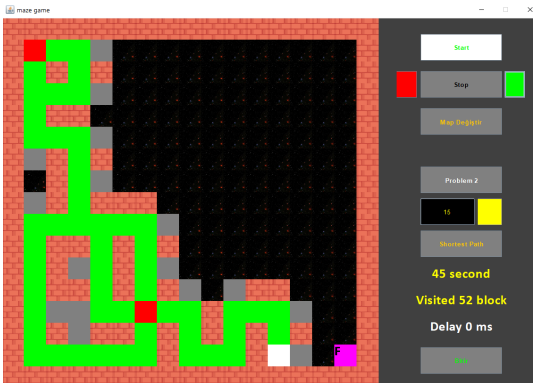

Fig. 8.

VII. Sonuçlar

Sonuç olarak, Problem1 çerçevesinde;Verilen URL adresleri okunur adreste bulunan bilgiler doğru bir şekilde oluşturulan matrislere aktarılır.Matriste bulunan bilgiler ile labirent 1,2 ve 3. tip engeller ile oluşturulur.Robot komşu kenarları öğrenerek başlagıç noktasından bitiş noktasına başarılı bir şekilde ulaşır.Ve en kısa yol robotun öğrenerek ulaştığı yol üzerinde gösterilir. Problem2 çerçevesinde ;kullanıcıdan alınan boyut bilgisi ile labirent 1 numaralı engeller ile oluştulur.Robot random bir şekilde belirlenen başlangıç noktasından başlayarak, hiçbir engelin konumunu bilmeden sadece komşu kenarlarının ne olduğunu öğrenerek random bir şekilde belirlenen bitiş noktasına ulaşır.En sonunda ise en kısa yolu gösterir.Bu işlemlerin hepsi arayüz üzerinde animasyonlu bir şekilde gözükür.Labirentin çözüm hızını arayüz üzerinde bulunan 2 buton sayesinde kontrol edilir.Hızlı bitirme işlemi,URL değiştirme işlemi,problem1 çözümü,problem2 çözümü olmak üzere kontrolu sağlayacak butonlar arayüzde bulunur.

References

[1] https://stackoverflow.com/questions/10136426/constructing-a-maze
[2] http://www.java2s.com/Code/Java/Swing-JFC/AligningtheTextinaJTextFieldComponent.htm
[3] https://www.youtube.com/watch?v=XIY9UTki9No
[4] https://en.wikipedia.org/wiki/Breadth-first_search/media/File:BFS-Algorithm_Search_Way.gif
[5] https://stackoverflow.com/questions/17865465/howdoidrawanimagetoa-jpanelorjframe
[6] https://math.hws.edu/eck/cs124/javanotes6/c6/s3.html
[7] https://en.wikipedia.org/wiki/Breadth-first_search
[8] https://weblog.jamisbuck.org/2011/1/10/maze-generation-prim-s-algorithm

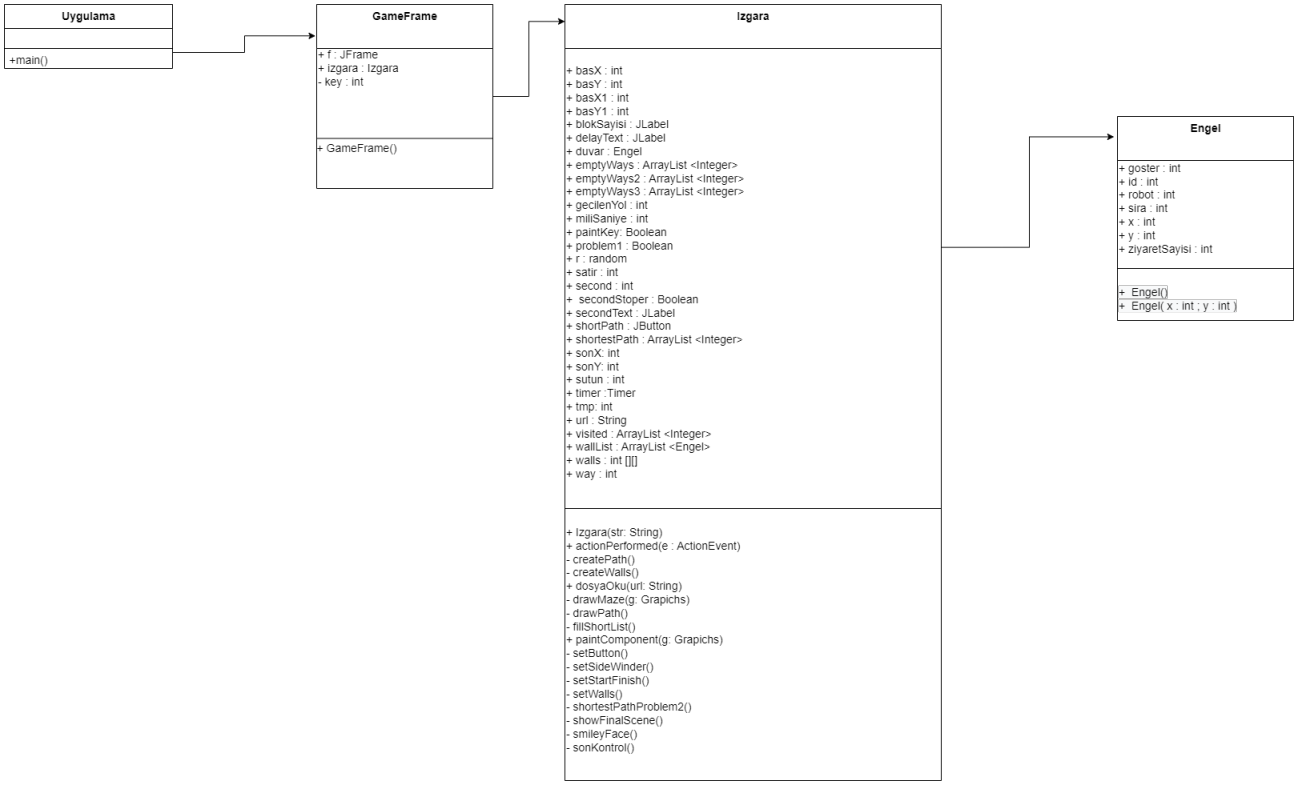

Fig. 9. UML DİYAGRAMI