# Gezgin Robot Projesi Raporu

Tayyib Okur
*200202060*
*Kocaeli Universitesi Bilgisayar Mühendisliği İÖ*
ultratayyib@gmail.com

Fatma Nur Kurt
*210202003*
*Kocaeli Universitesi Bilgisayar Mühendisliği İÖ*
kurtfatmanur8@gmail.com

***Index Terms*—Java, Java Swing, GUI, Class, Method, Inheritence, Polymorphism, Abstraction,ArrayList.**

## I. Özet

Bu rapor belgesi 2022-2023 Bahar dönemi BLM210 Programlama Laboratuvarı II 1. projeyi açıklamaya yönelik bilgiler içermektedir. Projenin ismi "Gezgin Robot Projesi"dir. Proje kapsamında belirli kurallara göre hareket eden bir robotun önündeki engelleri aşarak istenen hedefe ulaşmasını sağlayan bir oyun tasarlanması beklenmektedir. Oyunda 2 adet problemin çözülmesi gerekmektedir. Problem 1 ve problem 2'de 4 adet adım bulunmaktadır. Projemizde Main, Izgara, Engel, Robot ve EnKısaYolBul olmak üzere 5 farklı sınıf vardır. Bize önceden verilen bir url içerisindeki text dosyasına Main sınıfımızdaki UrlOku metoduyla erişerek ve dosyadaki tasarıma göre ızgara, engel yapısını Izgara sınıfımızdaki IzgaraOluştur metoduyla oluşturarak Problem 1'deki adım 1 ve adım 2'yi gerçekleştiriyoruz. Adım 3'te istenilen robotun başlangıç ve hedef noktalarını ızgara üzerinde uygun karelere Main sınıfımızdaki yapıcımızda rastgele belirliyoruz. Aynı zamanda Main sınıfımızdaki Paint metoduyla tüm ızgarayı bulutlarla kaplayarak robotun tüm ızgara dünyasını değil sadece çevresini görmesini sağlıyoruz. Böylelikle Problem 1 adım 3'teki tüm isterleri gerçekleştirmiş bulunuyoruz. Adım 4'te robotun geçtiği yolları ve çevresini öğrenerek hedefe ulaşmasını ve hedefe ulaşırken daha önce geçtiği yollar belli olacak şekilde her adımda yol üzerinde iz bırakması istenmektedir. Hedefe ulaştığında ise robotun öğrendiği yolların içinde başlangıç noktasından hedef noktasına giden en kısa yol ızgara üzerinde ayrıca çizdirilmelidir. Robotun geçtiği yolları öğrenmesi için Robot sınıfındaki koordinat değişkenlerini Main sınıfında bulunan YolAra metoduna aktarıyoruz. Robotun hedef noktasına ulaşana kadar engel bulunmayan yollardan hareket etmesini ve öğrenilen yolları bir ArrayList'te tutarak hafızasına almasını sağlıyoruz. Oluşturduğumuz ArrayList'i EnKısaYolBul sınıfında kullanarak robotun öğrendiği yolların içinde başlangıç noktasından hedef noktasına giden en kısa yolu buluyoruz. Bulduğumuz yolu başka bir ArrayList'e aktarıyoruz. Topladığımız verileri kullanarak Main sınıfında bulunan Paint metoduyla ekrana robotun öğrendiği, üstünden geçtiği yolları ve en kısa yolu çizdiriyoruz. Son olarak kodumuza eklediğimiz bir sayaç ile toplam süre ve toplam geçilen kare sayılarının verilerini Paint metodunda kullanarak ekrana yazdırıyoruz. Böylelikle Problem 1 için istenilen tüm adımları yerine getiriyoruz. Problem 2 adım 1 ve adım 2'de istenilenleri yaparak kullanıcı tarafından girilen boyutlarda oluşturduğumuz ızgara üzerine engeller yerleştiriyoruz ve rastgele bir labirent elde ediyoruz. Kullanıcının girdiği boyutları Main sınıfından Engel sınıfının yapıcısına aktarıyoruz. Engel sınıfındaki Generate metodu ile labirentimizi, içerisinde çıkmaz sokaklar da olacak şekilde rastgele oluşturuyoruz. Adım 3'te bizden istenilen labirentin giriş ve çıkış noktalarını, labirentin çapraz köşelerinde olacak şekilde Engel sınıfımızdaki LabirentYol metodunda belirliyoruz. Robot sınıfından aldığımız robotun hareket koordinat verilerini LabirentYol metodunda kullanıyoruz. Bu metotta Stack yapısını kullanarak robot yanlış bir yola girdiğinde doğru olarak tespit ettiği en son konuma gitmesini ve o konumdan itibaren yol aramaya devam etmesini sağlıyoruz. Aynı zamanda Main sınıfımızdaki Paint metoduyla tüm ızgarayı bulutlarla kaplayarak robotun tüm labirent dünyasını bilmemesini de sağlıyoruz. Böylelikle Problem 2 adım 3'ü de yerine getiriyoruz. Adım 4'de LabirentYol metodunda elde ettiğimiz verileri kullanarak Paint metoduyla robotun çıkışa ulaşmak için izlediği yolu adım adım ızgara üzerinde gösteriyoruz. Robot hedefe ulaştığında ise giriş noktasından çıkış noktasına giden yolu ızgara üzerinde ayrıca çizdiriyoruz. Kodumuza eklediğimiz bir sayaç ile toplam süre ve toplam geçilen kare sayılarının verilerini Paint metodunda kullanarak ekrana yazdırıyoruz. Son olarak elimizdeki robotun geçtiği yolları ve bulduğu en kısa yolu bir text dosyası açarak bu dosyaya yazdırıyoruz. Böylelikle projedeki tüm adımları yerine getiriyoruz.

Proje kapsamında herhangi bir sorun veya anlamadığımız kısım olduğu taktirde uzemde açılan forum aracılığıyla sorunlarımızı giderebiliyoruz.

## II. Giriş

Projenin amacı, 2022-2023 güz döneminde aldığımız Veri Yapıları ve Algoritmalar ve Nesneye Yönelik Programlama dersindeki bilgileri pekiştirmek aynı zamanda problem çözme becerisini geliştirmektir. Belirli kurallara göre hareket eden bir robotun, önündeki engelleri aşarak istenen hedefe ulaşmasını sağlayan bir oyun tasarlanması beklenmektedir. Oyunda temelde 2 adet problem vardır. Bizden, bu problemlerin çözümünü yaparken nesneye yönelik programlama ve veri yapıları bilgilerimizin kullanılması beklenmektedir. Proje C++, C sharp, Java veya Phyton dili kullanılarak gerçekleştirilecektir. Her problemin içerisinde 4'er adet adım bulunmaktadır. Problemler ve Adımlar aşağıda belirtildiği gibidir.

Problem 1:

Robotun ızgara üzerinde verilen hedefe ulaşmasını sağlarken geçtiği ve gördüğü yolları öğrenmesi, hedefe ulaştıktan sonra ise ızgara üzerinde robotun öğrendiği yollar içerisinden en kısa yolu bularak ızgara üzerinde gösterilmesi gerekmektedir.

Adım 1: İstenilen 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 uygulama çalıştırıldığında otomatik olarak erişilerek dosyadaki tasarıma göre ızgara ve engel yapısı oluşturulacaktır. Adım 3: Robotun başlangıç ve hedef noktaları ızgara üzerindeki uygun 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) belirtilmelidir. Adım 4: 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 robotun öğrendiği yollar içindeki en kısa yol ızgara üzerinde ayrıca çizdirilmelidir. Geçen toplam süre (sn cinsinden) ve kaç kare üzerinden geçildiği bilgileri ekranda gösterilmelidir.

Problem 2:

Kullanıcıdan alınan boyutlara göre oluşturulan labirent içerisinde robotun başlangıç noktasından çıkış noktasına ulaşması istenmektedir.

Adım 1: Kullanıcı tarafından istenilen boyutlarda bir ızgara oluşturmanız gerekmektedir. Adım 2: Izgara üzerine 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: 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.

Robot, Izgara, Engel ve Uygulama sınıfı olmak üzere işlevsel bakımdan genel olarak 4 adet sınıf tanımlanmış olup her sınıf için kullanılacak özellik ve metotların tanımlanması bizden beklenmektedir. Ayrıca gerektiği durumlarda neden kullanıldığı açıklanması koşuluyla yukarıdaki sınıflardan farklı olarak sınıf tanımlamaları yapılabilecektir.

Arayüz ve Görsellik:

Problem 1 ve problem 2 için uygulama ilk çalıştırıldığında başta robotun nasıl hareket ettiği simüle edilmeli ancak sonrasında zaman kaybı olmaması "Sonuç Göster" butonu aracılığıyla sonuç kullanıcıya direk gösterilmelidir. Problem 1 için arayüzde 3 buton yer almalı bunlar "URL Değiştir" butonu, "Çalıştır" Butonu ve "Sonuç Göster" butonları olacaktır. Proje ilk çalıştırıldığında 1. url adresindeki text dosyasına göre ızgara ve engel tasarımı yapılarak ekrana getirilecektir. Url Değiştir butonuna tıklandığında 2. url adresindeki text dosyasındaki veri kullanılarak ızgara ve engel yapısı tekrar oluşturulmalıdır. Butona her tıklandığında url adresi 1. ve 2. url adresleri arasında değiştirilerek tasarım yenilenmelidir. Sonrasında Çalıştır butonuna tıklandığında tüm ızgara alanı başta bulutla kapatılarak engel ve yolların görünümü engellenmelidir. Robotun yavaş hareketi bu sırada gösterilmeli ve robot hareket ettikçe gördüğü yerlerdeki (bir kare sonrası) bulutlar kaldırılmalıdır. Sonuç Göster butonuna tıklandığında ise oyun sonuca ilerletilerek robotun sırasıyla gezdiği kareler ve bulduğu en kısa yol ekranda çizdirilmeli ayrıca bir text dosyasına yazdırılarak kaydedilmelidir. Kare sayıları ve ne kadar süre tuttuğu bilgileri ekranda gösterilmelidir. Problem 2 için de problem 1'de olduğu gibi 3 buton arayüzde yer almalıdır. Bunlardan ilki "Labirent Değiştir" butonu olmalıdır. Bu butona her tıklandığında labirent tasarımı değiştirilmelidir. Yine Çalıştır ve Sonuç Göster butonları problem 1 deki fonksiyonların aynısını yerine getirmelidir. Robotun sırayla gezdiği kareler ve bulduğu en kısa yol ayrıca bir text dosyasına yazdırılarak kaydedilmelidir. Proje son teslim tarihi 27/03/2023.

## III. YÖNTEM

Projemize Main sınıfı oluşturarak başladık. Arayüz kısımları hariç öncelikle problem1 in kodunu sonrasında ise problem2 nin kodunu tamamladık. En son arayüz kısımlarına geçtik. Öncelikle bize verilen url adreslerindeki bilgileri çekmek için Main sınıfımıza UrlOku metodu ekledik. Bu metotta 0,1,2,3 rakamlarından oluşan matris şeklindeki 2 adet Url verilerini url1Veri ve url2Veri adındaki String Arraylist'lere aktardık. Sonrasında Izgara adında yeni bir sınıf oluşturduk. Izgara sınıfında Main sınıfından aldığımız Arraylistleri matris1 ve matris2 isimli 2 adet integer iki boyutlu diziye aktardık. Oluşturacağımız Robotu ve Robotun ızgara üzerindeki x ve y koordinatlarını tutması için Robot adında yeni bir sınıf oluşturduk. Main sınıfımızda Robot sınıfından robot isimli bir nesne oluşturduk. Matris isimli iki boyutlu dizi tanımladık. Main yapıcısına parametre olarak matris1 isimli iki boyutlu bir dizi girdik. Oluşturduğumuz matris dizisine aldığımız parametreyi aktardık. Böylelikle yapıcının içerisine ızgaramızın matris değerlerini aktarmış olduk. Integer veri tipli başlangıçX, başlangıçY, bitişX ve bitişY isimli 4 adet değişken tanımladık. Bu değişkenlerin her birine random kütüphanesini kullanarak boyutları matris boyutlarını aşmayacak şekilde ramdom değerler atadık. Bu random değerleri, ızgara üzerindeki uygun olan yerlere yerleştirmek için başlangıç ve bitiş koordinatlarının herhangi bir engel veya duvar üzerine gelmemesini while döngüleri içerisinde

kontrolünü gerçekleştirerek sağladık. Robotun başlangıçtan başlayarak yol araya araya bitiş noktasına gelmesini sağlamak için Main sınıfında boolean YolAra metodunu oluşturduk. Bu metoda parametre olarak integer iki boyutlu dizi, integer başlangıçX, integer başlangıçY, integer bitişX, integer bitişY ve path adında ınteger bir arraylist kullandık. YolAra metotunda Robot sınıfından oluşturduğumuz robot isimli nesnenin x ve y koordinatlarına ilk olarak başlangıçX ve başlangıçY değerlerimizi atadık. Sonrasında if kullanıyoruz ve robot.x'i bitişX'e robot.y'si bitişY'ye eşit ise yani hedefe ulaşılmışsa path adındaki arrayliste sırasıyla robot.x ve robot.y değerini ekliyoruz. Değerimizi true döndürerek metodumuzdan çıkıyoruz. Altına bir if daha açıyoruz. Matrisimizde robot.x ve robot.y koordinatlarımız (geçerli konum) 0 değerine sahip ise onu ziyaret ederek değerini 5 atıyoruz. Sonrasında robot.x ve robot.y değerlerimizi +1 -1 ekleyerek (sol, sağ, aşağı, yukarı) YolAra metodumuzda diğer parametreler aynı kalacak şekilde metot true döndürene kadar (hedef noktasına ulaşılana kadar) parametrelerimizi gönderiyoruz ve path'e ekleme yapıyoruz.Böylelikle hedefe ulaşana kadar Recursif olarak boş yolları (0 değerlerini) ziyaret ediyoruz ve koordinatlarımızı path arraylistimize ekliyoruz. Main yapıcısında YolAra metoduna matris, başlangıçX, başlangıçY, bitişX, bitişY ve path2 parametrelerini gönderiyoruz. Sonrasında path ve path3 olmak üzere integer arraylist daha tanımlıyoruz. Path2 yi for içinde ikişerli olarak tersten path e aktarıyoruz. Matrisimizdeki ziyaret edilen yerlere 5 atamıştık tekrardan 0 atıyoruz. Main sınıfımızda HafızaTut metodu oluşturuyoruz. Oluşturduğumuz metotda parametre olarak bir matris bir hafızamatris ve bir path tanımlıyoruz. matrisimizle aynı boyutta olan hafızamatrisimizin her değerini 1 olarak atıyoruz. Sonrasında ise pathde bulunan robotun geçtiği yolları ve öğrendiği çevresinin değerlerini engel ya da boşluk olmasına göre değiştirerek hafızamatrisimize atıyoruz. Robotun öğrendiği yollardan en kısa olanını bulmak için EnKısaYolBul sınıfımızı oluşturuyoruz. Bu sınıfımızda Hücre adında ayrı bir sınıfımız daha var. Hücre sınıfında Hücre tipinde önceki adında bir değişken int tiplerinde mesafe, satır ve sutun değişkenleri var. Bu sınıfta bulunan EnKısaYol metodu çağrıldığında metoda bir matris, başlangıç ve bitiş noktaları bir de en kısa yolu tutmak için path adında ArrayList gönderiyoruz. Hücre sınıfından matrisin boyutları kadar 2 boyutlu bir dizi oluşturuyoruz ve bu dizinin satir ve sutun elemanlarına matrisin konumları atanıyor. Mesafe elemanına 200 değeri atanıyor. Daha sonra Hücre tipinde tanımladığımız LinkListe (kuyruk) Robotun başlangıç noktasını atayıp mesafeyi 0 olarak ayarlıyoruz. Daha sonra kuyruktan bir eleman çekiyoruz eğer çektiğimiz elemanın satir ve sutun değerleri bitiş değerleriyle eşitse döngüden çıkıyoruz. Eşit değilse kuyruk boş olana kadar döngü devam ediyor. Döngüde ZiyaretEt fonksiyonu çağrılıyor. Ziyaret et fonksiyonuna o anki Hücre ve o hücrenin sağının, solunun, yukarısının ve aşağısının koordinatları sırasıyla gönderiliyor. Eğer sağı, solu, yukarısı, aşağısı serbest alan değilse o bölge için bir şey yapılmıyor. Mesela sağı serbest ise o hücrenin mesafesi bulunan hücresin mesafesinin 1 fazlası olarak atanıyor. En sonunda mesafeler büyükten küçüğe olacak şekilde path oluşturuluyor. Böylece en kısa yol tersten bulunmuş olunuyor. Main sınıfımızda enKısaYol isimli bir integer arraylist tanımlıyoruz. EnKısaYol'a tersten Path3'ü ekliyoruz. Böylelikle 'te en kısa yolun koordinatlarını tutmuş oluyoruz. Problem 1 için Arayüz haricindeki kısımları hallettik. Problem 2 için bir Engel sınıfı oluşturduk. Bu sınıfta generate metodu ile boş bir matrise rastgele engeller atayarak bir labirent oluşturduk. Labirenti oluştururken Prim algoritmasından yararlandık. Engellere 1, yollara 0 vererek içerisinde çıkmaz sokakların da bulunduğu başlangıç ve bitiş koordinatlarının karşılıklı köşelerde olduğu ve her çalıştırıldığında random atamalar sayesinde birbirinden farklı labirent üreten bir algoritma yazdık. Labirenti oluşturduktan sonra robotun çıkışı bulması için bir algoritma daha yazdık. LabirentYol metodu oluşturduk. Bu metotta Stack yapısı kullanarak robotun çıkmaz bir yola girdiğinde en son kaldığı yere dönerek oradan itibaren yol aramasını sağladık. Robot tüm komşu yolları ziyaret ederek çıkış noktasına ulaşacaktı.

Arayüz kısmı için Main sınıfında ana Main metodunda ilk olarak frame isimli bir Frame oluşturduk. setSize() metodu ile boyutunu belirledik. Frame'e btn1 ve btn2 isimli oluşturduğumuz Buttonları ekledik. Btn1'e "Oyun 1", btn2'ye "Oyun 2" yazdık. Butonların konumlarını ve boyutlarını setBounds() metodu ile belirledik. İstediğimiz renklerin RGB değerlerini internetten bularak bu değerlerle yeni Color oluşturduk. Butonları ve içindeki yazı renklerini setBackground() metodu ile boyadık. Yeni Font oluşturarak istediğimiz yazı şekillerini kullandık. Yeni ImageIcon oluşturarak proje klasörümüze eklediğimiz resimleri yeni bir Label'a aktardık. setSize() metodu ile boyutlarını belirledik. Oluşturduğumuz Label'ı Frame'e ekledik. (Fig. 1.) btn1'e tıklandığında yeni bir frame1 oluşturduk. Boyutunu belirledik. frame1'e oluşturduğumuz btn1 ve btn2 isimli Buttonları ekledik. Btn1'e "IZGARA 1", btn2'ye "IZGARA 2" yazdık. Butonların konumlarını ve boyutlarını belirledik. Butonları ve içindeki yazı renklerini boyadık. setFont() metodu ile yazı şekillerini değiştirdik. Ekleyeceğimiz resimleri YeniBoyut metodu ile istediğimiz boyutlarda ayarladık. frame1'e 3 adet ImageIcon ekledik. (Fig. 2.) Izgara 1 'e tıklandığında Main yapıcısına parametre olarak matris1'i gönderiyoruz. Izgara 2'ye tıklandığında ise Main yapıcısına parametre olarak matris2'yi gönderiyoruz. Main yapıcısından bir nesne oluşturuyoruz. Yapıcımızda bir JFrame oluşturmuştuk. Frame'in boyutlarını belirledik. Sonrasında Main sınıfımızda oluşturduğumuz paint metodunu Graphics g parametresiyle override ettik. ilk olarak matrisimizin her yerini bulut rengi olan griye setColor, fillRect ve drawRect metotları ile boyadık. Başlangıç noktamızı path'den alarak pembeye boyadık ve drawImage metodu ile bir Image ekledik. For döngüsü içerisinde path'ten robotumuzun gittiği yolun koordinatlarını alarak matrisimizde boyama işlemimizi gerçekleştirdik. Her adımda robot hareket ederken geçtiği yerlere oklar ekleyerek ne yönde ilerlediğini ekranda gösterdik. Bunları yaparken her adımda robotumuzun çevresindeki gri bulutları kaldırarak ekranda robotun çevresinin gözükmesini sağladık.

Bitişe geldiğinde ise belirli bir Image ekleyerek bitiş noktası olduğunu belirttik. Robot bitişe ulaştığında öğrendiği yolları ızgara üzerinde tekrar çizdirerek içindeki en kısa yolu oklarla gösterdik. Öğrendiği yolları göstermek için hafızamatrisi kullandık. En kısa yolu göstermek içinse enkısayol arraylistini kullandık. Toplam süreyi ölçmek için currentTimeMillis() metodunu kullandık. Toplam kare sayısını ölçmek için boyamayı yaptığımız döngümüze bir sayaç ekledik. drawString() metodu ile ekrana yazdırdık.(Fig. 3.) btn2'e tıklandığında yeni bir frame1 oluşturduk. Boyutunu belirledik. Yeni bir panel oluşturarak boyutunu frame1 ile aynı yaptık. Yeni labellar oluşturarak setText() metodu ile ekrana "labirent boyutlarını giriniz:", "X", "max 34x34" yazılarını yazdırdık. Boyutlarını ve koordinatlarını ayarladık. Ekranda kullanıcıdan veri girişi alabilmemiz için iki tane TextField oluşturduk. Boyutlarını ve konumlarını ayarladık. Yeni bir Button oluşturarak içine "Labirent Oluştur" yazdık. Rengini, boyutunu ve koordinatlarını ayarladık. Her yazı için belirlediğimiz Fontu kullandık. Panelimize Buttonu, TextField'ları, oluşturduğumuz labelları ve arkaplan için bir Image ekledik. frame1'e panelimizi ekleyerek ekranımızı oluşturduk.(Fig. 4.) Kullanıcıdan aldığımız labirent boyutlarını Engel sınıfımıza gönderdik. Engel yapıcımızda oluşturduğumuz Frame'in boyutlarını tüm ekran yaptık. Sonrasında oluşturduğumuz paint metodunu Graphics g parametresiyle override ettik. ilk olarak oluşturduğumuz labirentin her yerini bulut rengi olan griye setColor, fillRect ve drawRect metotları ile boyadık. Başlangıç noktamızı path'den alarak mor renge boyadık. Sonrasında robotumuzun belli olması için daha koyu bir mor renk ile bir imleç oluşturduk. Bu imleci bir for döngüsünde tekrar tekrar boyayarak robotun konumunun belli olmasını sağladık. Aynı döngüde robot çıkışa ulaşana kadar robotun geçtiği yolları mor ile gösterdik. Robotun çevresini görebilmesi için her adımda çevresindeki bulutları kaldırdık. Engelleri siyah, boş yolları beyaz olarak gösterdik. Robot çıkışa ulaştığında labirentimizin tamamını görmeniz için tekrardan labirenti bulutsuz bir şekilde gösterdik ve robotun geri döndüğü yolları silerek geçtiği en kısa yolu mora boyadık. Toplam süreyi ölçmek için currentTimeMillis() metodunu kullandık. Toplam kare sayısını ölçmek için boyamayı yaptığımız döngümüze bir sayaç ekledik. drawString() metodu ile ekrana yazdırdık.(Fig. 5.) Projemizde Arayüz kısımlarının adımlarını görmeniz için her adımımıza bir Thread.sleep() metodu ekledik.

## IV. DENEYSEL SONUÇLAR VE SONUÇ

Projeyi geliştirmeye başladığımız günden itibaren araştırma kabiliyetlerimizin geliştiğini, nesneye yönelik programlama dersinin mantığını daha iyi kavradığımızı, veri yapılarında Stack kullanımı, LinkList kullanımı ve özellikle graph yapısı hakkında epeyce bilgi edinip deneyim sağladığımızı söyleyebiliriz. Graphda en kısa yol nasıl bulunur bunun hakkında bilgi sahibi olduk. Ayrıca Recursif fonksiyon kullanımı konusunda da gelişmiş olduk. En çok kendimizi geliştirdiğimiz nokta Arayüz kısmı oldu. Arayüzde Graphic nasıl kullanılır graphicde boyama yapma, çizgi çizme, resim ekleme gibi bir çok şey öğrendik. Aynı zamanda JButton kullanımı ve JLabel kullanımını tekrardan hatırladık.

Projemizi çalıştırdığımızda karşımıza ana ekranımız gelmektedir.

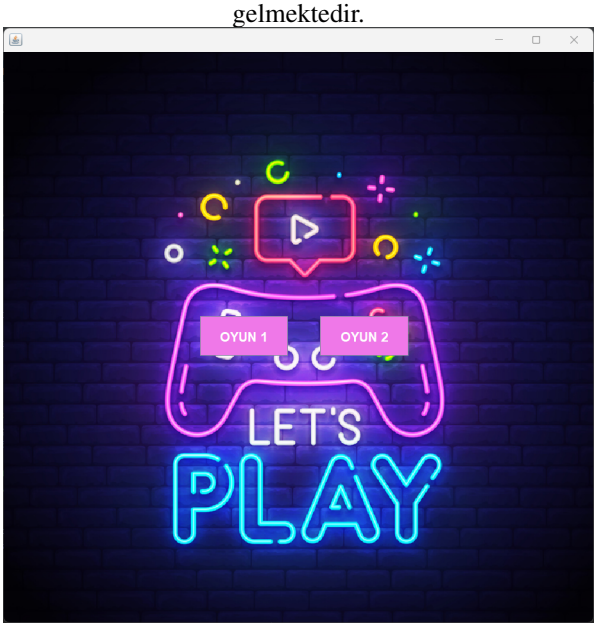


Fig. 1.
Problem 1'i görmek için OYUN 1 butonuna,
Problem 2'yi görmek için OYUN 2 butonuna tıklayınız.

Oyun 1 butonuna tıkladığınızda iki seçenek gelecektir.

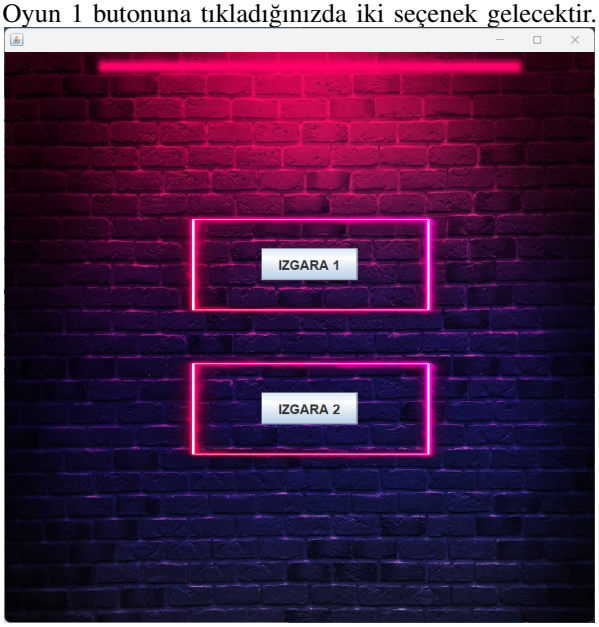


Fig. 2.
IZGARA 1 url1'i, IZGARA 2 url2'yi temsil etmektedir.
Seçmek istediğiniz ızgara butonuna tıklayınız.

Izgara 2 butonuna tıklandığında

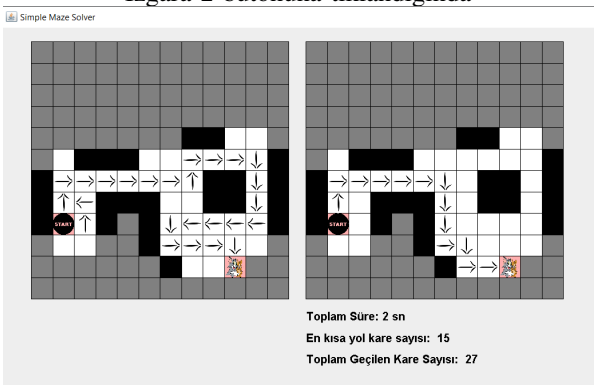


Fig. 3.

Robotumuz start noktasından başlayarak hedef noktasına ulaşmaktadır. Sol ızgarada robotun geçtiği yerler oklar ile gösterilmiştir. Sağ ızgarada robota göre en kısa yol oklarla çizdirilmiştir. Ayrıca süre ve kare bilgileri ekranda verilmiştir.

Oyun 2 butonuna tıkladığınızda labirentin boyutlarını girmeniz istenecektir.


Fig. 4.

Labirent Oluştur butonuna tıklandığında istediğiniz boyutlarda random bir labirent oluşacaktır.

## GELIŞTIRME ORTAMI

Kodumuzu java dilinde yazdık. Bunun için İntellij ide ve NetBeans ide kullandık. Arayüz için Java Swing kullandık. Kullandığımız kütüphaneler:

- java.awt.*;
- java.io.BufferedReader;
- java.io.BufferedWriter;
- java.io.File;
- java.io.FileWriter;
- java.io.IOException;


Fig. 5.

Robot giriş noktasından başlayarak çıkmaz sokağa girdiğinde geri dönerek çıkışa ulaşmaktadır. Çıkışa ulaştığında ise sol tarafta geçilen kısa yol gösterilmiştir. Ayrıca süre ve kare bilgileri ekranda verilmiştir.

- java.io.InputStreamReader;
- java.net.MalformedURLException;
- java.net.URL;
- java.util.*;
- java.util.ArrayList;
- java.util.LinkedList;
- java.util.Random;
- java.util.Stack;
- javax.imageio.ImageIO;
- javax.swing.*;
- javax.swing.JLabel;
- javax.swing.JPanel;

## KAYNAKÇA

[1] URl dosyasını açıp içindeki verileri okuyup kodumuza dahil edebilmek için ekteki internet sitesinden yararlanılmıştır. [2] .txt uzantılı dosya açıp içine veri yazabilmek için Youtube Kodlama vakti kanalından yararlanılmıştır. Ekte adres belirtilmiştir [3] Thread yapısı için kodlama vakti kanalından yararlanılmıştır.Ekte adres Belirtilmiştir. [4] Veri yapıları için ekte verilen kitap kullanılmıştır. [5] Gui Arayüzü için ekte bulunan Youtube kanalından yararlanılmıştır [6] Java için ekte verilen Youtube adresinden yararlanılmıştır. [7] Java için ekte verilen kitaptan yararlanılmıştır [**?**] Labirent oluşturma için prim algoritmasından yararlanılmıştır.

## REFERENCES

[1] https://tr.code-paper.com/q/11956-i-am-having-java-read-a-txt-file-from-a-url-live-by-line-and-print-to-a-txt-file-i-want-to-edit-the-txt-file-being-printed-to-remove-parts-of-the-txt?ysclid=leq1w7eagh985064740
[2] https://www.youtube.com/@KodlamaVakti
[3] https://www.youtube.com/@KodlamaVakti
[4] Toros Rifat Colkesen, VeriY pıları Ve Algoritmalar, PapatyaBilim, Istanbul, 2021
[5] https://www.youtube.com/@BroCodez
[6] https://www.youtube.com/@KodlamaVakti
[7] Mehmet Kirazlı - Sezer TANRIVERDİOĞLU, Her Yönüyle java, Kodlab,İstanbul,2022
[8] Bilgisayar kavramları Youtube kanalı