GEZGİN ROBOT PROJESİ

1st Tuna GÜLCAN
Bilgisayar Mühendisliği
Prolab-2 1. Proje
İzmit, KOCAELİ
tuna.gulcan@outlook.com

1st Fahrettin DOĞAN
Bilgisayar Mühendisliği
Prolab-2 1. Proje
İzmit, KOCAELİ
fahrettinndogann7@gmail.com

I. ÖZET

Bu projede nesneye yönelik programlama ile urlden içerik okumayı ve bu içeriklerle birlikte bir cismin bir harita üzerinde başlangıç noktasından hedef noktasına giden en kısa yolu bulmasını sağladık. Swing kütüphanesi ile bu işlemi görselleştirdik.

II. GİRİŞ

Bu rapor gezgin robot projesinin özet, giriş, yöntem deneysel sonuçlar, sonuç ve kaynakça bölümlerinden oluşmaktadır. Ek olarak akış diyagramı ve UML diyagramlarını da göstermektedir.

III. YÖNTEM

A. *Main Sınıf*

main metodu : Arayüzü main metodunda başlatıyoruz.

B. *Problem 1 Sınıfı*

Bu sınıfta verilen url dosyalarındaki txt dosyalarını okuyup labirenti çizdiriyoruz.

Problem1 Constructor : Frame oluşturuyoruz. Url dosyalarını bu metodda açıyoruz. Labirenti çizmek için updateMatris metodunu kullanıyoruz. Başla butonuyla başlangıç noktamızı DepthFirst sınıfında traverse metoduna yolluyoruz. Labirentte gezme ve arama işlemi bu metodda gerçekleşiyor.Gezilen yerler ekranda gösteriliyor. Bitir butonuyla arama işleminin sonuna geçiliyor ve geçilen yollar ekranda gösteriliyor. Değiştir butonuyla url dosyaları arasında geçiş yapıyoruz, Ekrana yeni labirent bastırılıyor.

updateMatris Metodu : Url dosyalarını okuyup buradaki değerleri bir matrise aktarıyoruz. Matrisi gezip engelleri rastgele biçimde tekrardan oluşturuyoruz.

C. *DepthFirst Sınıfı*

isValidSpot Metodu : Matriste aranması için gelen noktayı matris boyutları içinde mi ve o nokta yol mu diye kontrol sağlar.

traverse Metodu : Recursive metottur. Gelen noktayı isValidSpot metoduna yollar eğer bu nokta yol ise sırasıyla sağ ,sol ,alt , yukarı noktalarını kontrol etmeye başlar. Recursive olmasından dolayı gidebildiği noktaya kadar gider.Eğer aranan nokta bulunursa true bulunamazsa false değeri döndürür.

D. *Problem 2 Sınıfı*

Bu sınıfta istenilen boyutlarda rastgele bir labirent üretiyoruz.

Problem 2 Constructor : Frame oluşturuyoruz. Kullanıcıdan satır ve sütun boyutunu alıyoruz. Oluştur butonuyla labirenti ekranda bastırıyoruz.

createMaze Metodu : bu metodda istenilen boyutlardaki labirenti oluşturuyoruz.

showMaze Metodu : Ekranda görülen labirenti silip yeni bir labirent oluşturur.

resizeMaze Metodu : Yeni labirentin boyutlarını alır.

E. *DepthFirst 2 Sınıfı*

isValidSpot Metodu : Matriste aranması için gelen noktayı matris boyutları içinde mi ve o nokta yol mu diye kontrol sağlar.

traverse Metodu : Recursive metottur. Gelen noktayı isValidSpot metoduna yollar eğer bu nokta yol ise sırasıyla sağ ,sol ,alt , yukarı noktalarını kontrol etmeye başlar. Recursive olmasından dolayı gidebildiği noktaya kadar gider.Eğer aranan nokta bulunursa true bulunamazsa false değeri döndürür.

F. *Engel 2 Sınıfı*

2x2lik engelleri yeniden oluşturduğumuz sınıftır.

engelsec Metodu : 0 ile 5 arasında rastgele sayı üreten metottur.

engelturu Metodları : üretilen rastgele sayıya göre o fonskiyona girip engeli yeniden düzenler.

G. *Engel 3 Sınıfı*

3x3lük engelleri yeniden oluşturduğumuz sınıftır.

engelsec Metodu : 0 ile 5 arasında rastgele sayı üreten metottur.

engelturu Metodları : üretilen rastgele sayıya göre o fonskiyona girip engeli yeniden düzenler.

IV. DENEYSEL SONUÇLAR

```
public Problem1() throws MalformedURLException {
    setTitle("Labirent Oyunu");
    setSize( width: 900,  height: 900);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    panel = new JPanel();
    panel.setLayout(new FlowLayout());
    degistirButton = new JButton( text: "Değiştir");
    degistirButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if(URLPath1.equals("http://bilgisayar.kocaeli.edu.tr/prolab2/url1.txt")) {
                URLPath1 = ("http://bilgisayar.kocaeli.edu.tr/prolab2/url2.txt");
                try {
                    url= new URL(URLPath2);
                } catch (MalformedURLException ex) {
                    throw new RuntimeException(ex);
                }
            }
            else {
                URLPath1 = ("http://bilgisayar.kocaeli.edu.tr/prolab2/url1.txt");
                try {
                    url= new URL(URLPath1);
                } catch (MalformedURLException ex) {
                    throw new RuntimeException(ex);
                }
            }
            updateMatris();
            repaint();
        }
    });
```

Fig. 1. *
URL Okuma Fonksiyonu

Verilen urlleri okumaya yarayan fonksiyon.

```
public void paint(Graphics g) {
    super.paint(g);

    g.translate( x: 50,  y: 50);
    for (int i = 0; i < Problem1.col; i++) {
        for (int j = 0; j < Problem1.col; j++) {
            Color color = null;
            switch (Problem1.matris[i][j]) {
                case 0:
                    color = Color.WHITE;
                    break;
                case 1:
                    color = Color.BLACK;
                    break;
                case 7:
                    color = Color.BLUE;
                    break;
                case 8:
                    color = Color.RED;
                    break;
                case 4:
                    color = Color.ORANGE;
                    break;
                case 5:
                    color = Color.WHITE;
                    break;
                case 9:
                    color = Color.GREEN;
            }
            g.setColor(color);
            g.fillRect( x: 30 * j,  y: 30 * i,  width: 30,  height: 30);
            g.setColor(Color.BLACK);
            g.drawRect( x: 30 * j,  y: 30 * i,  width: 30,  height: 30);
        }
    }

   for (int p = 0; p < path.size(); p += 2) {
        int pathX = path.get(p);
        int pathY = path.get(p + 1);
        g.setColor(Color.GREEN);
        g.fillRect( x: 30 * pathY,  y: 30 * pathX,  width: 30,  height: 30);
    }
}
```

Fig. 2. *
Problem 1 Tablosunu Ekrana Çizdirme

Bu fonksiyon 1. problemin matrisini ekrana çizdirmeye yarıyor.

```
baslaButton = new JButton( text: "Başla");
baslaButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Thread t = new Thread(new Runnable() {
            public void run() {
                sleeptime=300;
                if (DepthFirst.traverse(matris,  x: 0,  y: 0, col)) {
                    System.out.println("Yol bulundu!");
                    int x = 0, y = 0;
                    path.add(x);
                    path.add(y);
                    while (x != col - 2 || y != col - 1) {
                        if (y + 1 < col && matris[x][y + 1] == 5) {
                            y++;
                        } else if (x + 1 < col && matris[x + 1][y] == 5) {
                            x++;
                        }
                        path.add(x);
                        path.add(y);
                        repaint();
                        try {
                            Thread.sleep(sleeptime); // yavaşlama için
                        } catch (InterruptedException ex) {
                            ex.printStackTrace();
                        }
                    }
                } else {
                    System.out.println("Yol bulunamadı!");
                }
            }
        });
        t.start();
    }

});
```

Fig. 3. *
Problem 1 Başlama Butonu Fonksiyonu

Bu fonksiyon 1. problemde cismin harekete başlayıp en kısa yolu bulmasını sağlıyor.

```
bitirButton = new JButton( text: "Bitir");
bitirButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        sleeptime=0;
        if (!path.isEmpty()) {
            int p = 0;
            while (p < path.size() - 1) {
                int x = path.get(p);
                int y = path.get(p + 1);
                matris[x][y] = 9;
                p += 2;
            }
            path.clear();
            try {
                Thread.sleep(sleeptime); // yavaşlama için
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
            repaint();

        }

    }
});
panel.add(baslaButton);
panel.add(bitirButton);
panel.add(degistirButton);
add(panel, BorderLayout.SOUTH);
updateMatris();
```

Fig. 4. *
Problem 1 Bitirme Butonu Fonksiyonu

Bitirme butonu fonksiyonunun başlangıç butonundan farkı cismin en kısa yolu direkt bularak ekrana direkt bastırması.

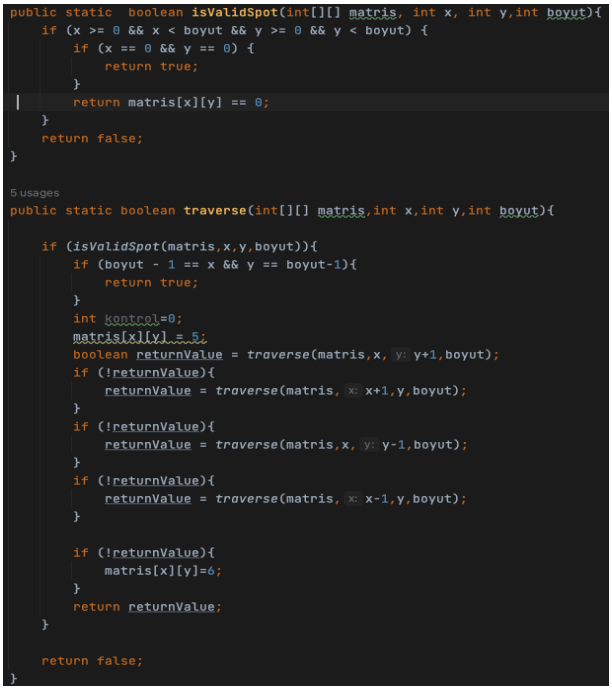

Fig. 5. *

En Kısa Yol Algoritması

Bu fonksiyon cismin haritada dolaşarak en kısa yolu bulmasını sağlıyor.

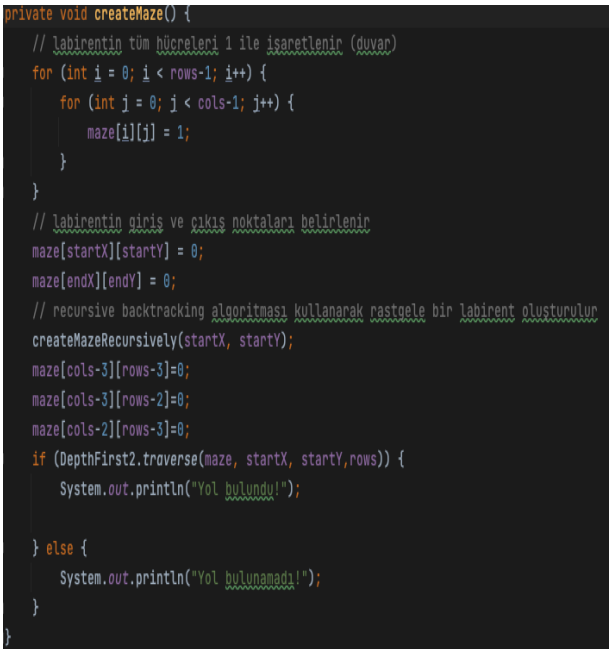

Fig. 6. *

Problem 2 Labirent Oluşturma Fonksiyonu

Problem 2'nin başlangıç ve bitiş noktalarının belirlendiği ve labirentin oluşturulma fonksiyonunun ilk kısmı.

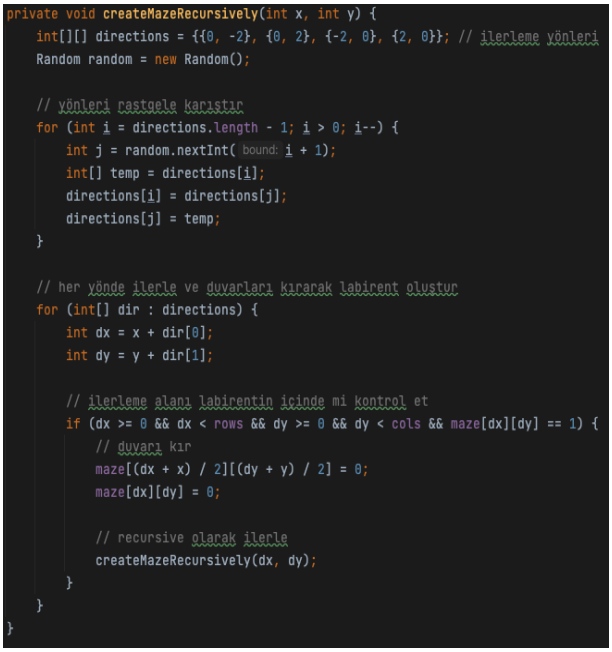

Fig. 7. *

Problem 2 Labirent Oluşturma Fonksiyonu 2. Kısım

Labirent oluşturma fonksiyonunun ikinci kısmı.

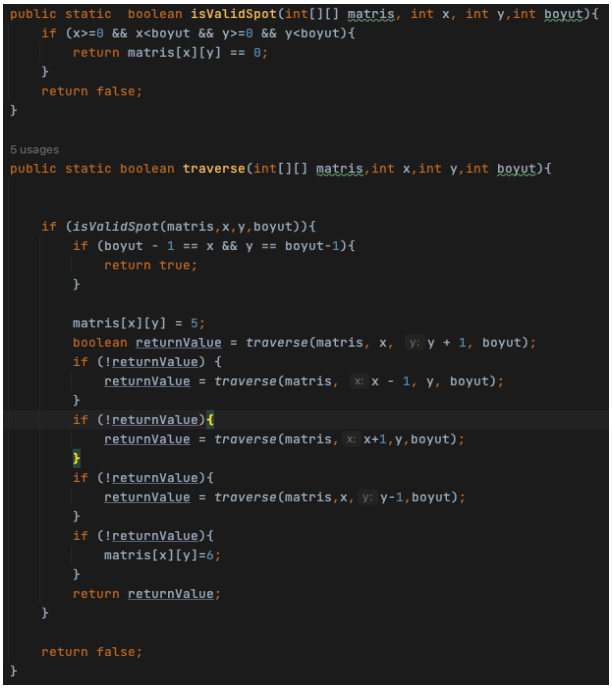

Fig. 8. *

Problem 2 Labirenti Dolaşma Fonksiyonu

Problem 2 cismin labirenti dolaşmasını sağlayan fonksiyon.

V. SONUÇ

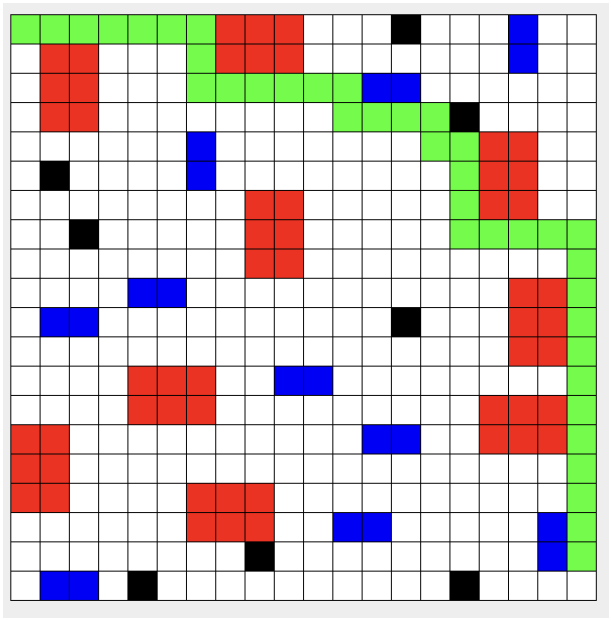

Fig. 9. *
URL1 Haritasında En Kısa Yol

1. URL'nin haritasında en kısa yolun bulunup ekrana bastırılmış hali.

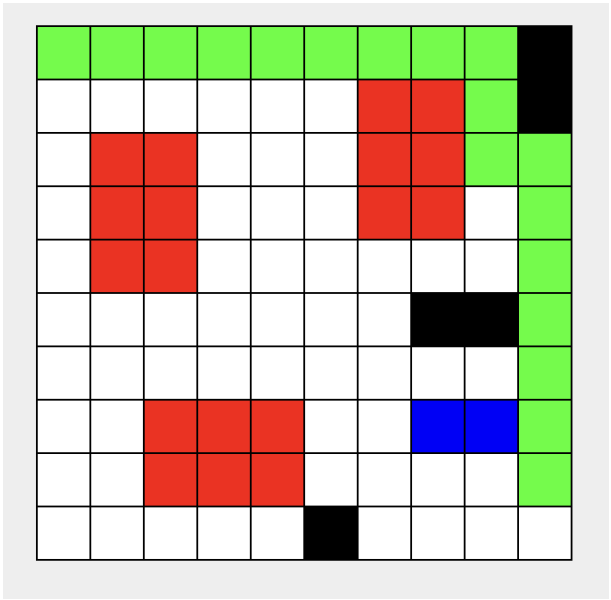

Fig. 10. *
URL2 Haritasında En Kısa Yol

2. URL'nin haritasında en kısa yolun bulunup ekrana bastırılmış hali.

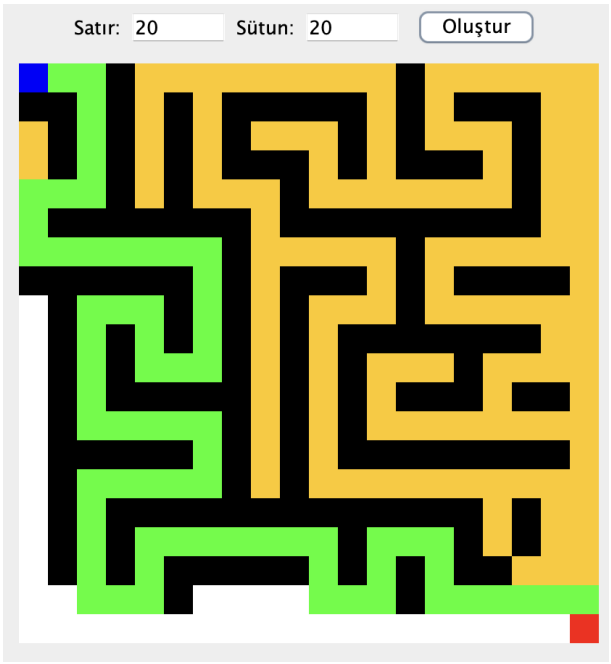

Fig. 11. *
Problem 2 Labirentin Çözülmüş Gösterimi

Problem 2 deki labirentin çözülüp ekrana bastırılmış hali.

VI. AKIŞ DİYAGRAMI

1-Problem 1 ve Problem 2 seçimini yapıyoruz.

Problem 1 seçilirse:
1-Url dosyaları okunup matrise aktarılır.
2-Başla butonu ile Başlangıç noktasından itibaren arama işlemi başlatılır.
3-Aranan nokta bulunana kadar gezilen noktalar işaretlenir.
4-Bitir butonuna basılırsa gezme işlemi sonuna getirilir. Direkt yol gösterilir.
5-Aranan nokta bulunduktan sonra işlem biter.

Problem 2 seçilirse:
1-Kullanıcıdan labirent boyutu alınır.
2-Labirent oluşturulur.
3-Başlangıç noktasından itibaren arama işlemi başlatılır.
4-Aranan nokta bulunana kadar gezilen noktalar işaretlenir.
5-Bulunduktan sonra işlem biter.

VII. KAYNAKÇA

REFERENCES

[1] https://stackoverflow.com
[2] https://www.javatpoint.com
[3] https://www.tutorialspoint.com/index.htm
[4] https://www.geeksforgeeks.org
[5] https://www.java-examples.com
[6] https://www.delftstack.com
[7] https://www.thejavaprogrammer.com
[8] https://github.com

VIII. UML DİYAGRAMI

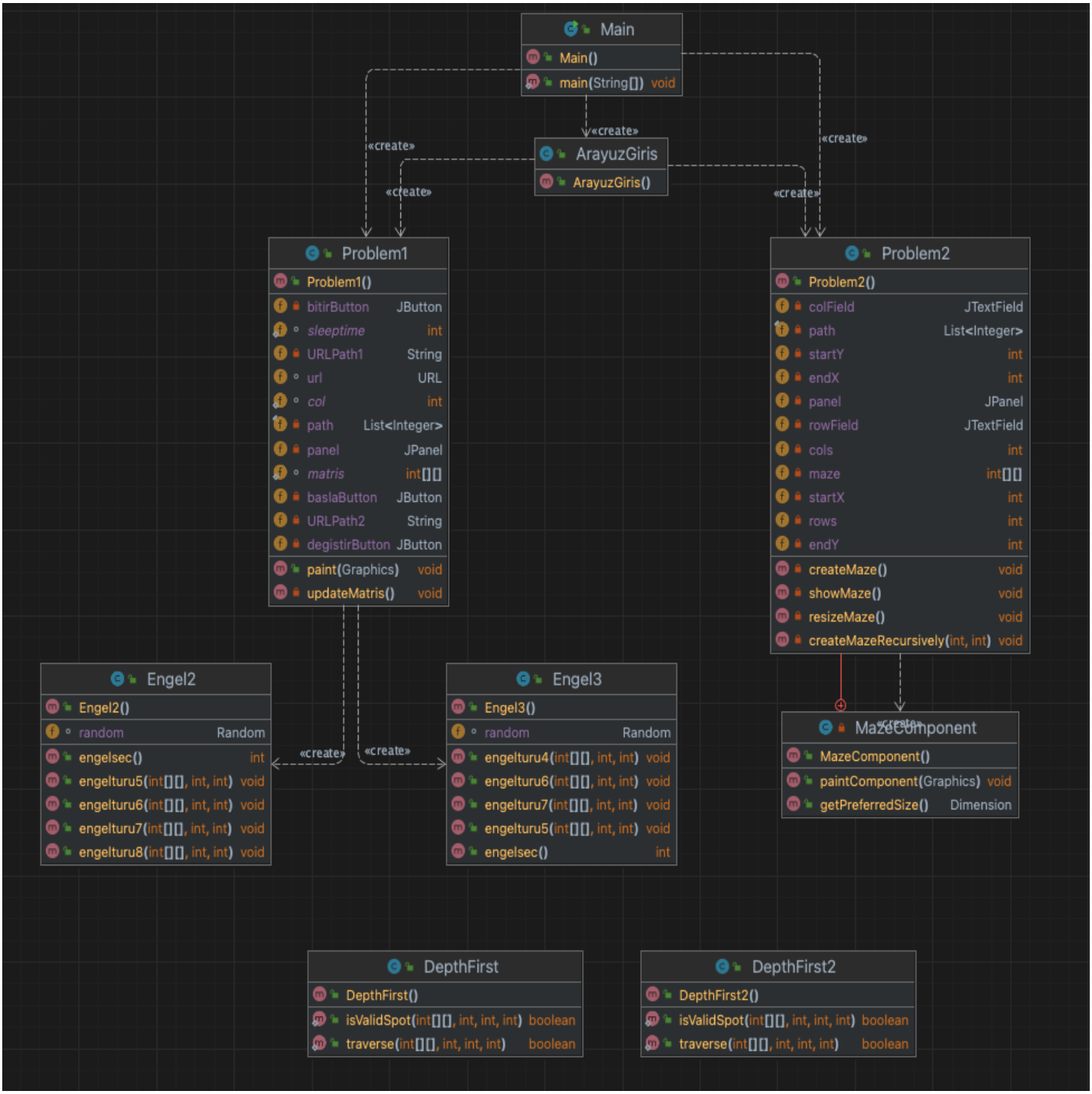

Fig. 12. *
UML Diyagramı