Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Programlama Laboratuvarı II

ŞİRİNLER

Taha Uz/Burak Emir Ayrancı

maskeden_2000@hotmail.com[Taha]/190202014@kocaeli.edu.tr[Emir]

Projenin Özeti:

Programlama laboratuvar II Projesi olarak bizden "Şirinler" adlı oyun geliştirmemiz beklenmektedir.

Biz proje için Java programlama dilini kullanmayı ve Netbeans ile Intelij geliştirme ortamında yazmayı tercih ettik.

Proje dökümanında bizden nesneye yönelik programlama ve veri yapıları algoritmalarını kullanarak Şirinler oyununu tasarlamamız istenmektedir.

Projede biz Java programlama dilinde bulunan "Swing" adlı arayüz tasarım kütüphanesinden yararlandık.

İlk olarak "oyuncu" ve "obje" nesneleri programda tanımlanıyor. Sonra "harita.txt" karakter adlarını ve haritayı okuyarak bu bilgiler "arrayList" 'lerde tutuluyor. Düşman karakterleri "Dusman" nesnesi olarak programda tanımlıyoruz. Karakterlerin seçimine ve "txt" deki kapı seçimine göre karakterlerin oyunun haritasında hangi noktada gözüküceğini belirlemek için kordinatları "gargamel" ve "azman" 'ın kordinatlarını güncelliyoruz. "addKeyListener" ile oyunu ok tuşlarıyla kontrol etme mekaniğini ekliyoruz . Daha sonra altın ve mantar objelerin rasgele belirmesi , haritanın arayüzde çizilmesi , düşman karakterin haritadaki hareketleri , "Intro ve karakter seçimi" ekranının devreye girmesi , skor ve arayüz göstericek belli başlı algoritmaların devreye girmesini kontrol eden "Timer" ' lar devreye girerek bunun kontrolünü sağlıyoruz.

Oyuncu klavyeden "Escape" tuşuna bastığında oyun sırası veya "Game Over" ya da "Win Smurfs" ekranında programın sonlandırılması sağlanılıyor.

1.GİRİŞ

Proje için Java programlama dilini kullanmayı ve Netbeans ile Intelij geliştirme ortamında yazmayı tercih ettik.

Java programlama dili ; Sun Microsystems mühendislerinden James Gosling tarafından geliştirilmeye başlanmış açık kodlu , nesneye yönelik , zeminden bağımsız , yüksek verimli , çok işlevli , yüksek seviye , adım adım işletilen (yorumlanan-interpreted) bir dildir.

Netbeans platformu; Oracle tarafından geliştirilen bir java geliştirme ortamıdır (IDE) ve ücretsiz olarak dağıtılmaktadır. Özellikle kullanıcı arayüzü tasarımında sağladığı kolaylıklardan dolayı tercih edilmektedir

IntelliJ IDEA Java'da bilgisayar yazılımı geliştirmek için kullanılan bir tümleşik geliştirme ortamıdır (IDE). JetBrains (resmi adı ile IntelliJ) şirketi tarafından geliştirilmektedir. Hem Apache 2 lisansı altında yayınlanan topluluk sürümü hem de sahipli ticari sürümü bulunmaktadır. Her ikisi de ticari amaçlı geliştirmelerde kullanılabilmektedir.

2.TEMEL BİLGİLER

Projeyi tasarımını dosya okuma,arayüz,oyun mekanikleri ve nesnelerin türleri olmak üzere 4 farklı alt başlıklara ayırarak anlatıcaz.

2.1. Dosya Okuma

İlk olarak “harita.txt” dosyasından dosyadaki tüm satırları “satırList” adlı “string” tipindeki listede depoluyoruz. Alttaki resimde gördüğünüz gibi ;

```
try (Stream<String> stream = Files.lines(Paths.get(fileName))) {
    satirList = stream.collect(Collectors.toList());
} catch (IOException e) {
    e.printStackTrace();
}
```

Dosyayi “satırList” kaydettikten sonra ilk satırlarındaki karakter ismi ve karakterin olduğu kapının ismini depolayacak “karakter” ve “kapi” adlı listelerde depolaniyor. Bunu döngü içinde yaparken “String” metodlarını kullanarak ilk olarak karakter ismini “Karakter :” daki “:” işaretindeki kısımla “,” işareti arasındaki “String ” ‘ i “karakter” isimli listeye atıyorum. Sonra kapı türünü de “i” den sonra “+2” ekleyerek sonraki kelimeyi alarak kapı adını da “kapi” isimli listeye atıyoruz . Böylece “substring” adlı metod yardımıyla dosyadan Karakter ve Kapı değişkenlerini okumuş oluyoruz. Satır okurken satırın “0” ya da ”1” gibi rakam ile başlamasıyla bu döngüden çıkıyoruz. Alttaki resimde gördüğümüz gibi ;

```
for (int i = 0; i < satirList.size(); i++) {
    if (satirList.get(i).charAt(0) == '1' || satirList.get(i).charAt(0) == '0') {
        break;
    }
    karakter.add(satirList.get(i).substring(satirList.get(i).indexOf(':') + 1, satirList.get(i).indexOf(',')));
    kapi.add(satirList.get(i).substring(satirList.get(i).indexOf('i') + 2));
}
```

Dosyadan ilk okuma kısmını hallettikten sonra ikinci aşama oyunun “1” ve “0” lardan oluşan harita kısmını da “bord” dizisine aktararak dosya okuması yapılır. Bunu yaparken ilk olarak dosyanın okunucağı satır “karakter.size()” ile aldığımız listenin boyutu neyse örnek veriyorum “2” olsun , dosya okuyan döngüyü “2.” satırdan başlatarak satırları ilk olarak “\t” karakterini satırlardan silerek rakamları sade haliyle bırakıyoruz. Sade halleriyle satırların rakamlarını okuyarak “bord” adlı haritayı tutan dizide depoluyoruz değişkenleri. Alttaki resimde gördüğünüz gibi ;

```
// "Harita.txt" ' den 0 ile 1 leri okuyarak harita belirliyor.
for (int i = karakter.size(); i < satirList.size(); i++) {
    // Ilgili satırdaki TAB karakterlerini siliyor.
    String a = satirList.get(i).replace( target: "\t", replacement: "");
    for (int j = 0; j < a.length(); j++) {
        bord[i - karakter.size()][j] = a.charAt(j) - 48;
    }
}
```

2.2. Arayüz

2.2.1. Oynanış Ekranı

Dosya okuma yolu ile elde ettiğimiz “0” ve “1” lerden oluşan harita bilgimiz ile harita tasarımı yapıyoruz. Bunu yaparken “JFrame” nesnesini bizim bu işlemlerimizi yaptığımız nesnenin içinde kalıtım yoluyla “extends JFrame ” diyerek ekliyoruz. Daha sonra “JFrame” içindeki “paint” metodunu “public void paint(Graphics g)” metodunun içine “super.paint(g)” metodunun mirasını alıyorum böylelikle haritayı çizerken “Graphics” nesnesinden yararlanıyorum. Haritamın çizimini yaparken “0” ları “Gri” “1”’leri beyaz şekilde haritanın kalıbını çıkartıyorum. Burada “0” ‘lar duvarı “1” ler ise karakterlerin dolaşabildiği zemini temsil ediyor.

Daha önce altınları “3” , mantarları “4”, Azman’ı “5” , Gargamel’i “6”, Şirine’yi “7” , olarak tanımlayıp haritanın içindeki konumlarına “drawImage” ,”setColor” ve “fillRect” metodları yardımıyla haritayı döngü yardımıyla çiziyorum.Alttaki resimlerde gördüğünüz gibi ;

```
public void paint(Graphics g) {

    super.paint(g);

    for (int i = 0; i < bord.length; i++) {
        for (int j = 0; j < bord[0].length; j++) {
            if ((i == 0 & j == 3) | (i == 0 & j == 10) | (i == 5 & j == 0) | (i == 10 & j == 3)) {
                g.setColor(Color.PINK);
                g.fillRect( x: (j * scale) + 20,  y: (i * scale) + 32, scale, scale);
            }
            if (bord[i][j] == 1) {
                g.setColor(Color.WHITE);
            } else if (bord[i][j] == 0) {
                g.setColor(Color.GRAY);
            } else if (bord[i][j] == 3) {
                g.setColor(Color.YELLOW);
                g.drawImage(altin,  x: (j * scale) + 20,  y: (i * scale) + 32,  observer: this);
            } else if (bord[i][j] == 4) {
                g.setColor(Color.RED);
                g.drawImage(mantar,  x: (j * scale) + 20,  y: (i * scale) + 32,  observer: this);
            } else if (bord[i][j] == 9) {
                g.setColor(Color.BLUE);
                if (check_oyuncu == 1) {
                    g.drawImage(gozluklu,  x: (j * scale) + 20,  y: (i * scale) + 32,  observer: this);
                } else {
                    g.drawImage(uykucu,  x: (j * scale) + 20,  y: (i * scale) + 32,  observer: this);
                }

            } else if (bord[i][j] == 5) {
                g.drawImage(azmann,  x: (j * scale) + 20,  y: (i * scale) + 32,  observer: this);
            } else if (bord[i][j] == 6) {
                g.drawImage(gargamell,  x: (j * scale) + 20,  y: (i * scale) + 32,  observer: this);
            } else if (bord[i][j] == 7) {
                g.drawImage(sirine,  x: (j * scale) + 20,  y: (i * scale) + 32,  observer: this);
            }
```

```
            if (bord[i][j] == 1 | bord[i][j] == 0) {
                if ((i == 0 & j == 3) | (i == 0 & j == 10) | (i == 5 & j == 0) | (i == 10 & j == 3)) {

                } else {
                    g.fillRect( x: (j * scale) + 20,  y: (i * scale) + 32, scale, scale);
                }
            }
        }
    }
}
```

Daha sonra Dusman karakterlerin girebiliceği kapıların konumlarını girerek bu kısımları da ayrı bir renk(pembe) ile kapı olduğunu belirliyoruz aşağıdaki resimde görüldüğü gibi ;

```
if ((i == 0 & j == 3) | (i == 0 & j == 10) | (i == 5 & j == 0) | (i == 10 & j == 3)) {
    g.setColor(Color.PINK);
    g.fillRect( x: (j * scale) + 20,  y: (i * scale) + 32, scale, scale);
}
```

Karakterlerin(Düşman,Oyuncu) ve objelerin(Altin ve Mantar) resimlerini “loadImage” metoduyla resimleri değişkenlere atıyoruz ve haritayı çizdirirken kullanıyoruz .

```
public Image mantar, altin, gozluklu, uykucu, sirine, gargamell, azmann, background, gozluklu_bg, uykucu_bg;
```

```
public void loadImage() {
    mantar = new ImageIcon( filename: "mantar.jpg").getImage();
    altin = new ImageIcon( filename: "gold.png").getImage();
    gozluklu = new ImageIcon( filename: "gozluklu.jpg").getImage();
    gargamell = new ImageIcon( filename: "gargamel.jpg").getImage();
    azmann = new ImageIcon( filename: "azman.jpg").getImage();
    sirine = new ImageIcon( filename: "sirine.jpg").getImage();
    uykucu = new ImageIcon( filename: "uykucu.jpg").getImage();
    background = new ImageIcon( filename: "TheSmurfs.jpg").getImage();
    uykucu_bg = new ImageIcon( filename: "uykucu1.jpg").getImage();
    gozluklu_bg = new ImageIcon( filename: "gozluklu1.jpg").getImage();
}
```

Şekilde kırmızı kutucuklarla işaretlenmiş olan kısımlar yukarıdaki yöntemle çizdirilen karakterler ve objelerdir.(Oyun içinde kırmızı renkli kutucuklar yoktur sadece anlatım açısından daha iyi anlaşılsın diye çizilmiştir.)

Daha sonra skorlarımız “scoreShow” metodu yardımıyla oyun içi ekranında gösteriyoruz . Bunu yaparken “Graphics” nesnesinden yardım alarak “drawString” metodu ile arayüzün “setFont”,”setColor” metoduyla font tipini ve rengini ayarlayarak oyuncuyu bilgilendirmek için çizdiriyoruz .

Bununla birlikte oyuncunun oyundan nasıl çıkabiliceği bilgisini de oyuncuya aktarmakta kullanıyoruz yine skor gösterirken kullandığımız metodlarla.

```
public void scoreShow(Graphics g, int skor) {
    String score = "Score : " + skor;
    g.setColor(Color.YELLOW);
    g.setFont(new Font( name: "Arial", Font.BOLD,  size: 14));
    g.drawString(score,  x: (bord[0].length) + 10,  y: (bord.length * scale) + 20);

    String exit = "EXIT GAME ==> ESC ";
    g.setColor(Color.BLACK);
    g.setFont(new Font( name: "Arial", Font.BOLD,  size: 20));
    g.drawString(exit,  x: (bord[0].length * scale) - 200,  y: (bord.length * scale) + 20);
}
```

Oyunun skorun 0 ya da 0 ‘ın altında bir değer olunca “Game Over” ekranını çıkaran “gameOverShow” metodundaki bir koşul devreye girerek “Game Over ” ekranının belirmesine neden oluyor.

```
public void gameOverShow(Graphics g, int skor) {
    if (skor > 0) {
        String score_ = "WINNER SMURFS !";
        g.setColor(Color.BLACK);
        g.setFont(new Font( name: "Arial", Font.BOLD,  size: 50));
        g.drawString(score_,  x: ((bord.length * scale) / 2) - 150, ((bord[0].length * scale) / 2));
    } else {
        String score_ = "GAME OVER";
        bitti = 1;
        g.setColor(Color.red);
        g.setFont(new Font( name: "Arial", Font.BOLD,  size: 50));
        g.drawString(score_,  x: ((bord.length * scale) / 2) - 80, ((bord[0].length * scale) / 2));
    }
}
```


```
} else {
    String score_ = "GAME OVER";
    bitti = 1;
    g.setColor(Color.red);
    g.setFont(new Font( name: "Arial", Font.BOLD,  size: 50));
    g.drawString(score_,  x: ((bord.length * scale) / 2) - 80, ((bord[0].length * scale) / 2));
}
```

Eğer karakterimiz başarılı bir şekilde Şirine’ye ulaşırsa o zamanda kazanma ekranı devreye giriyor.


```
if (skor > 0) {
    String score_ = "WINNER SMURFS !";
    g.setColor(Color.BLACK);
    g.setFont(new Font( name: "Arial", Font.BOLD,  size: 50));
    g.drawString(score_,  x: ((bord.length * scale) / 2) - 150, ((bord[0].length * scale) / 2));
} else {
```

Son olarak düşman karakterinin oyuncuya olan uzaklığıylanı belirten ve eğer bazı durumlar devreye girmezse(duvar,para,mantar karşısına çıkmazsa) standart olarak takip ediceği yol çizilir.

Yol çizilmeden önce ilk olarak “Karakter” nesnesinin içindeki “enKisaYol” metodundan düşman karakter ile oyuncu karakter arasinda en kısa mesafe hesabı yapılıyor.

```
public int[][] enKisaYol(int x,int y){
    int distx;
    int disty;

    int [][] kordinatlar = new int[1][2];

    distx = y-lokasyon.getY_ekseni();
    disty = x-lokasyon.getX_ekseni();

    kordinatlar[0][0] = distx;
    kordinatlar[0][1] = disty;


    return kordinatlar;
}
```

Daha sonra bu mesafenin x ve y nin negatif pozitif etkenleri göz önünde bulundurarak düşman zekasının içinde haritadaki koşula göre yol çizimi yapılıyor “drawPath” metodunun içinde.

```
if (check_oyuncu == 1) {
    kordinat_kisayol = azman.enKisaYol(oyuncu1.LokasyonX(), oyuncu1.LokasyonY());
} else if (check_oyuncu == 2) {
    kordinat_kisayol = azman.enKisaYol(oyuncu2.LokasyonX(), oyuncu2.LokasyonY());
}

int x_kisa = kordinat_kisayol[0][0];
int y_kisa = kordinat_kisayol[0][1];

int sayac1_a = 0, sayac2_a = 0, sayac3_a = 0, sayac4_a = 0;

drawPath(getGraphics(),x_kisa,y_kisa,azman.LokasyonX(),azman.LokasyonY(), select: true);
```

“drawPath” metodu ile alınan kordinatlara uygun yol haritası negatif ve pozitifliği kontrol ederek yolu çiziyor.

```
public void drawPath(Graphics g, int x, int y, int dusman_x, int dusman_y, boolean select) {

    if (select) {
        g.setColor(Color.GREEN);
        if (x > 0) {
            for (int i = x; i > 0; i--) {
                g.fillRect( x: ((dusman_y + i) * scale) + 20, y: (dusman_x * scale) + 32, scale, scale);
            }
        } else if (x < 0) {
            for (int i = x; i < 0; i++) {
                g.fillRect( x: ((dusman_y + i) * scale) + 20, y: (dusman_x * scale) + 32, scale, scale);
            }
        }

        if (y > 0) {
            for (int i = y; i > 0; i--) {
                g.fillRect( x: ((dusman_y + x) * scale) + 20, y: ((dusman_x + i) * scale) + 32, scale, scale);
            }
        } else if (y < 0) {
            for (int i = y; i < 0; i++) {
                g.fillRect( x: ((dusman_y + x) * scale) + 20, y: ((dusman_x + i) * scale) + 32, scale, scale);
            }
        }
    }
```

```
    if (!select) {
        g.setColor(Color.RED);
        if (y > 0) {
            for (int i = y; i > 0; i--) {
                g.fillRect( x: ((dusman_y) * scale) + 20, y: ((dusman_x + i) * scale) + 32, scale, scale);
            }
        } else if (y < 0) {
            for (int i = y; i < 0; i++) {
                g.fillRect( x: ((dusman_y) * scale) + 20, y: ((dusman_x + i) * scale) + 32, scale, scale);
            }
        }

        if (x > 0) {
            for (int i = x; i > 0; i--) {
                g.fillRect( x: ((dusman_y + i) * scale) + 20, y: ((dusman_x + y) * scale) + 32, scale, scale);
            }
        } else if (x < 0) {
            for (int i = x; i < 0; i++) {
                g.fillRect( x: ((dusman_y + i) * scale) + 20, y: ((dusman_x + y) * scale) + 32, scale, scale);
            }
        }

    }

    if (check_oyuncu == 1) {
        g.drawImage(gozluklu, x: ((dusman_y + x) * scale) + 20, y: ((dusman_x + y) * scale) + 32, observer: this);
    } else {
        g.drawImage(uykucu, x: ((dusman_y + x) * scale) + 20, y: ((dusman_x + y) * scale) + 32, observer: this);
    }

  }
}
```

Oyuncunun isteğine bağlı olarak bu yol çizim mekaniğini istediği gibi açma ve kapatma tuşu vardır .

2.2.2.Giriş Ekranı

Oyunu çalıştırdığımızda ilk olarak bizi karakter seçim ekranı karşılıyor. Burada oyuncuya bilgi vererek 2 karakter arası seçim yaptırılıyor.

Giriş ekranında "showIntro" metodunun yardımıyla oyuncuya yapıcağı eylem hakkında bilgi verilir. Ekrana "Graphics" nesnesi sayesinde ulaştığımız "setFont","setColor","drawString" kullanılarak ekrana bilgi verilir.

```
public void showIntro(Graphics g) {
    g.drawImage(background, x: ((bord.length * scale) / 2) - 130, y: 40, observer: this);
    String start = " Welcome The Smurfs Game";
    g.setColor(Color.BLACK);
    g.setFont(new Font( name: "Arial", Font.BOLD, size: 40));
    g.drawString(start, x: ((bord.length * scale) / 2) - 220, ((bord[0].length * scale) / 2));

    String karakter = "Karakter Secimi Yapmak Icin Ok Tuslarini Kullanin";
    g.setColor(Color.RED);
    g.setFont(new Font( name: "Arial", Font.BOLD, size: 20));
    g.drawString(karakter, x: ((bord.length * scale) / 2) - 180, y: ((bord[0].length * scale) / 2) + 50);

    String karakter1 = "Gozluklu Sirin(Sol OK Tusu)            Uykucu Sirin(Sag Ok Tusu) ";
    g.setColor(Color.BLUE);
    g.setFont(new Font( name: "Arial", Font.BOLD, size: 20));
    g.drawString(karakter1, x: ((bord.length * scale) / 2) - 240, y: ((bord[0].length * scale) / 2) + 90);

    String karakter2 = "<===  ===>";
    g.setColor(Color.BLACK);
    g.setFont(new Font( name: "Arial", Font.BOLD, size: 45));
    g.drawString(karakter2, x: ((bord.length * scale) / 2) - 50, y: ((bord[0].length * scale) / 2) + 170);

    g.drawImage(gozluklu_bg, x: ((bord.length * scale) / 2) - 150, y: ((bord[0].length * scale) / 2) + 107, observer: this);
    g.drawImage(uykucu_bg, x: ((bord.length * scale) / 2) + 200, y: ((bord[0].length * scale) / 2) + 107, observer: this);

}
```

2.3. Oyun Mekanikleri

2.3.1.Skor İşlenmesi Algoritması

Oyuncunun oynadığı karakter düşman karakterlerden birine yakalandığı zaman düşman karakterin türüne göre puan kayıp ederek skor tablosu güncelleniyor.

Aşağıdaki resimde görüldüğü üzere Gargamel karakterinin oyuncu karakterine dokunduktan sonra oyuncunun güncel skorundan Gargamel karakterinin içinde yazılı olan "puanKaybet" metodu ile oyuncu karakterinden puan eksiltme mekaniğini gösteriyor.

```
if (kordinat_kisayol[0][0] == 0 & kordinat_kisayol[0][1] == 0) {
    if (check_oyuncu == 1) {
        oyuncu1.setSkor(oyuncu1.getSkor() - gargamel.puanKaybet());
    } else if (check_oyuncu == 2) {
        oyuncu2.setSkor(oyuncu2.getSkor() - gargamel.puanKaybet());
    }
    gargamel.LokasyonGir(first_X, first_Y);
```

Aynı mekaniğe Azman karakterininde sahip olduğunu görüyoruz.

```
if (kordinat_kisayol[0][0] == 0 & kordinat_kisayol[0][1] == 0) {
    if (check_oyuncu == 1) {
        oyuncu1.setSkor(oyuncu1.getSkor() - azman.puanKaybet());
    } else if (check_oyuncu == 2) {
        oyuncu2.setSkor(oyuncu2.getSkor() - azman.puanKaybet());
    }
    azman.LokasyonGir(first_X_a, first_Y_a);
```

Oyuncudan puanı eksiltikten sonra oyun ekranı her saniye ekrana yeniden basıldığı için ve bununla birlikte skor ekranı da her saniye güncellendiğinden dolayı alttaki resimde "scorShow" metodunun "Graphics" ve "skor" parametreleri alarak arayüze çizdirmesini görüyoruz.

```
public void scoreShow(Graphics g, int skor) {
    String score = "Score : " + skor;
    g.setColor(Color.YELLOW);
    g.setFont(new Font( name: "Arial", Font.BOLD,  size: 14));
    g.drawString(score,  x: (bord[0].length) + 10,  y: (bord.length * scale) + 20);
```

2.3.2.Düşman Karakterin Hareket Algoritması

Düşman karakterleri adımlarını "Timer" ' ın içinde kontrol edilir . İlk koşul olarak oyuncu karakterinin adım atıp atmadığı "dusman_harekter" değişkeni ile kontrol edilir eğer oyuncu adım attı ise(dusman_harekter==1) düşman karakterleri de adım atar böylece sıra tabanlı şekilde hareket edilir.

```
if (dusman_hareket == 1) {
```

Daha sonra oyuncu tipine göre duşman karakterleri içerisindeki "enKisaYol" metoduyla "kordinat_kisayol" dizisinin içine "kordinatlar" dizisinin atamasını yapar.

```
if (check_oyuncu == 1) {
    kordinat_kisayol = gargamel.enKisaYol(oyuncu1.LokasyonX(), oyuncu1.LokasyonY());
} else if (check_oyuncu == 2) {
    kordinat_kisayol = gargamel.enKisaYol(oyuncu2.LokasyonX(), oyuncu2.LokasyonY());
}
```

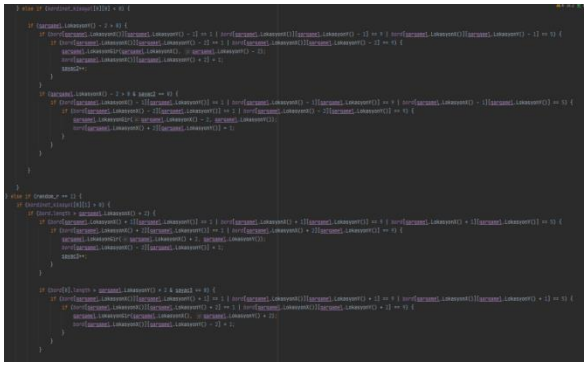
Daha sonra "drawPath" algoritmasının içinde kullanılması için "kordinat_kisayol" dizisinin içindeki "x" ve "y" bilgileri değişkenlere atanır.

```
int x_kisa = kordinat_kisayol[0][0];
int y_kisa = kordinat_kisayol[0][1];
```

Bu atama yapıldıktan sonra kullanıcının "Space" tuşunu kullanarak "path_secim" değişkeni ile "drawPath" metodunun çalışması için tercih koşulu devreye girer.

```
if(path_secim % 2 == 1) {
    drawPath(getGraphics(), x_kisa, y_kisa, gargamel.LokasyonX(), gargamel.LokasyonY(),  selectt: false);
}
```

Koşullar ile düşman karakterinin oyuncu karakter ile aynı karede olup olmadığı kontrol edilir. Eğer aynı karede ise "Oyuncu" karakterden puan eksilir ve "Düşman" karakteri haritadaki geldiği kapıya geri döner. Aynı kordinatta değilseler düşman karakteri rasgele bir biçimde x ve y kordinatlarından birini seçerek "Oyuncu" ve "Düşman" karakterinin arasindaki x ve y her saniyede düşman türüne göre ilerleme katederek "Oyuncu" karakterini yakalamaya çalışır. Oyuncuyu yakalamaya çalışırken duvarlara dikkat ederek yolunu bulmaya çalışır.



2.3.3.Oyuncu Karakterin Yürüme Algoritması

İlk olarak “KeyAdepter” nesnesinin içindeki “keyPressed” metodu sayesinde kullanıcının klavyedeki ok tuşları ile karakteri kontrol etmesini sağlıyoruz. Bunu yaparken ok tuşlarına basılmasıyla “adim” değişkeninin “1-3-5-2” gibi değerleri almasıyla karakterin hangi yöne hareket etmesini sağlıyacağı “while” döngüsüne gidiyor.

```
addKeyListener(new KeyAdapter() {

    @Override
    public void keyPressed(KeyEvent e) {

        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            if (secim == 0) {
                check_oyuncu = 1;
                secim = 1;
            } else if (bitti == 0) {
                adim = 1;
            }

        } else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {

            if (secim == 0) {
                check_oyuncu = 2;
                secim = 1;
            } else if (bitti == 0) {
                adim = 3;
            }

        } else if (e.getKeyCode() == KeyEvent.VK_UP) {
            if (bitti == 0) {

                adim = 5;
            }
        } else if (e.getKeyCode() == KeyEvent.VK_DOWN) {
            if (bitti == 0) {

                adim = 2;
            }

        } else if (e.getKeyCode() == KeyEvent.VK_ESCAPE) {
            adim = 4;
        } else if (e.getKeyCode() == KeyEvent.VK_SPACE) {
            path_secim++;
        }
    }

});
```

Girdiği “while” döngüsünde karakterin “yukarı-aşağı-sağ-sol ” gibi yönlere hareketi kontrol ediliyor. Örneğin karakter “Sol” yöne doğru ilerlemek isterse ilk olarak gidiceği yöndeki karenin içini kontrol ediyor.İlk kontrol olarak gidiceği yerin haritanın içinde mi dışında mı olduğu kontrol ediliyor daha sonra gidiceği karenin duvar olup olmadığı kontrol edliliyor . Duvar ise oyuncu hareket edemiyor. Duvar değil ise karenin içinde altın veya mantar gibi objeler var mı onu kontrol edip objelerin verdiği puana göre oyuncu skoruna puan eklemesi yapılıyor.

```
if (adim == 3) {
    adim = 0;
    if (bord[0].length > oyuncu2.LokasyonY() + 1) {
        if (bord[oyuncu2.LokasyonX()][oyuncu2.LokasyonY() + 1] != 0) {
            if (bord[oyuncu2.LokasyonX()][oyuncu2.LokasyonY() + 1] == 3) {
                oyuncu2.setSkor(oyuncu2.getSkor() + altin.PuanKazan());
            }
            if (bord[oyuncu2.LokasyonX()][oyuncu2.LokasyonY() + 1] == 4) {
                oyuncu2.setSkor(oyuncu2.getSkor() + mantar.PuanKazan());
            }
```

Eğer oyuncunun gittiği karenin içinde “Şirine” karakteri var ise oyun sonlanıyor. Böylelikle oyunu sonlandırma sistemi devreye girerek oyuncuya skoru hakkında son bilgisi verilip “Timer”lar sonlandırılıp döngüyü bitiriyor.

```
if (bord[oyuncu2.LokasyonX()][oyuncu2.LokasyonY() + 1] == 7) {
    bord[oyuncu2.LokasyonX()][oyuncu2.LokasyonY() + 1] = 9;
    oyuncu2.LokasyonGir(oyuncu2.LokasyonX(), y: oyuncu2.LokasyonY() + 1);
    bord[oyuncu2.LokasyonX()][oyuncu2.LokasyonY() - 1] = 1;

    System.out.println("Game Over");
    oyuncu2.PuaniGoster();
    myTimer.cancel();
    game = false;
    paint(getGraphics());
    scoreShow(getGraphics(), oyuncu2.getSkor());
    gameOverShow(getGraphics(), oyuncu2.getSkor());
    bitti = 1;
    adim = 7;
}
```

Karakter oyun sırasında haritada “9” numara ile hareket ediyor . Her seferinde haritada ilerlerken arkasındaki “9” numarayı “1” numara ile değiştirmemiz gerekiyor . Bunu yapmazsak karakter haritada eski konumunda “9” olucak ve böylelikle haritanın her yerinde karekterimiz oluşucak ve oyunumuz “bug” ‘ a girmiş olucak.

```
if (adim != 7) {
    bord[oyuncu2.LokasyonX()][oyuncu2.LokasyonY() + 1] = 9;
    oyuncu2.LokasyonGir(oyuncu2.LokasyonX(), y: oyuncu2.LokasyonY() + 1);
    if (hamle != 0) {
        bord[oyuncu2.LokasyonX()][oyuncu2.LokasyonY() - 1] = 1;
    }
}
```

Tüm yönlerde bu kontrolleri yaparak karakterimizin haritada ilerlemesini sağlıyoruz.

2.3.4.Mantar ve Altın Algoritması

Oyun başladıktan sonra 5 altın haritada 5 farklı konumda 5 saniye gözükücek şekilde rasgele belirirler. Mantar ise haritada rasgele şekilde 1 tane olucak biçimde belirir. Daha sonra "Timer" lar eşliğinde bu döngü oyunun sonuna kadar devam eder. Haritada bu tür objeler belirlenirken dikkat edilen bazı durumlar var. Bu durumlar objelerin duvar,oyuncu,düşman gibi şeylerin üstünde belirmemesi için koşul durumları ile kontrol edilerek rasgele haritada belirmesini sağlanır "Timer" algoritmaların içindeki koşullar sayesinde.

```
TimerTask gorev = new TimerTask() {
    @Override
    public void run() {

        // Random Altin Uretir
        for (int i = 0; i < sayacAltin; i++) {
            random1 = (int) (Math.random() * bord.length);
            random2 = (int) (Math.random() * bord[0].length);
            if (bord[random1][random2] == 1) {
                sifirla[k][0] = random1;
                sifirla[k][1] = random2;
                bord[random1][random2] = 3;
                k++;
            } else {
                sayacAltin++;
            }
        }
        sayacAltin = 5;
        temp_k = k;
        k = 0;

        if (!game) {
            myTimer.cancel();
        }
    }
};

myTimer.schedule(gorev,  delay: 0,  period: 5050);
```

Yukarıdaki resimde görüldüğü üzere rasgele sayı üretilir haritanın boyutları içinde. Bu üretilen rasgele sayılarla haritada üzerinde düşman,karakter ya da duvar var mı diye kontol edilir koşul ile. Eğer duvar ,düşman,oyuncu karakter var ise döngü "+1" arttırılarak altın sayısının beş olana kadar döngünün devam ettirilmesi sağlanır. Koşul durumunda haritada duvar,düşman,oyuncu karakter yok ise orada altın üretilir ve bu rasgele kordinatlar bir diziye kayıt edilir böylece sonradan bu üretilen altınlar sıfırlanması sağlanır.

```
TimerTask gorev2 = new TimerTask() {
    @Override
    public void run() {
        int a, b;

        // Altinlar Uretildigi yerden silindigi kod satiri

        for (int i = 0; i < temp_k; i++) {
            a = sifirla[i][0];
            b = sifirla[i][1];
            bord[a][b] = 1;

        }
    }
};

myTimer.schedule(gorev2,  delay: 5000,  period: 5000);
```

Mantarlar için altınlardan farkı yedi saniyede bir üretilmesi ve sadece bir kez üretilmesidir.

```
TimerTask gorev3 = new TimerTask() {
    @Override
    public void run() {
        // Random Mantar Uretir
        for (int i = 0; i < sayacMantar; i++) {
            random1 = (int) (Math.random() * bord.length);
            random2 = (int) (Math.random() * bord[0].length);
            if (bord[random1][random2] == 1) {
                sifirla2[k1][0] = random1;
                sifirla2[k1][1] = random2;
                bord[random1][random2] = 4;
                k1++;
            } else {
                sayacMantar++;
            }
        }
        sayacMantar = 1;
        temp_k1 = k1;
        k1 = 0;
    }
};

myTimer.schedule(gorev3,  delay: 0,  period: 7050);
```

```
TimerTask gorev4 = new TimerTask() {
    @Override
    public void run() {
        int a, b;

        // Mantarlar Uretildigi yerden silindigi kod satiri
        for (int i = 0; i < temp_k1; i++) {
            a = sifirla2[i][0];
            b = sifirla2[i][1];
            bord[a][b] = 1;

        }
    }
};

myTimer.schedule(gorev4,  delay: 7000,  period: 7000);
```

2.3.5. SPACE Tuşunun Oyuna Etkisi

Oyunda düşman karakterin oyuncuya gidiceği yolu çizen "drawPath" metodu vardır.Bu metodun bazı hatalara sahip olması ve bu hataları belirtilen sürede gideremiceğimiz anladığımız için bu durumu geliştirdik. "Space"

tuşuna basarak “drawPath” metodunun açıp kapatma özelliği getirdik. Böylece oyuncu rahatsız olduğunda kapatmak isterse kapatabiliyor bunun yanı sıra eğer ihtiyacı olduğu zaman da “Space” tuşuna basarak tekrar açabilmekte.

```
} else if (e.getKeyCode() == KeyEvent.VK_SPACE) {
    path_secim++;
}
```

“Space” tuşuna basıldığında “path_secim” değişkeni “+1” artıyor böylece bu değişkenin tek ve çift sayı olması kontrol edilerek bu özelliğin açılıp kapanması kontrolü sağlanılıyor.

```
if(path_secim % 2 == 1) {
    drawPath(getGraphics(), x_kisa, y_kisa, gargamel.LokasyonX(), gargamel.LokasyonY(), select: false);
}
```

2.3.6. ENTER Tuşunun Oyuna Etkisi

Oyun oynanışı sırasında “ENTER” tuşuna basarak oyunda gerçek zamanlı oynanış moduna geçerek oyundaki karakterler sıra tabanlı ilerlemekten çıkıyor.

2.3.7. Gargamel’in Azman’nın Üzerinden Zıplayabilmesi

Gargamel 2 birim hareket etmesi ile birlikte 1 birim gidiceği yerde Azman karakteri var ise onun üstünden de geçip harekettine devam ediyor. Aşağıdaki resimde “== 5” koşulundaki “5” Azman karakterini temsil etmektedir.

```
if (bord[gargamel.LokasyonX()][gargamel.LokasyonY() + 1] == 1 |
        bord[gargamel.LokasyonX()][gargamel.LokasyonY() + 1] == 9 |
        bord[gargamel.LokasyonX()][gargamel.LokasyonY() + 1] == 5)
```

2.3.8. Gözlüklü Şirin ve Gargamel’in İki Birim İlerlemesi ve Çevreyle Etkileşimleri

Gözlüklü şirinden başlamak gerekirse 2 birim hareket eder ve bu hareketi sırasında altın ve mantar gibi nesnelerin üzerinden geçerken puan olarak eklenir ve cisimlerin yok olması mekanikleri de devreye giricek şekilde ayarlandı.

```
if (bord[oyuncu1.LokasyonX()][oyuncu1.LokasyonY() + 1] == 3) {
    oyuncu1.setSkor(oyuncu1.getSkor() + altin.PuanKazan());
    bord[oyuncu1.LokasyonX()][oyuncu1.LokasyonY() + 1] = 1;
}
if (bord[oyuncu1.LokasyonX()][oyuncu1.LokasyonY() + 2] == 3) {
    oyuncu1.setSkor(oyuncu1.getSkor() + altin.PuanKazan());
}
if (bord[oyuncu1.LokasyonX()][oyuncu1.LokasyonY() + 1] == 4) {
    oyuncu1.setSkor(oyuncu1.getSkor() + mantar.PuanKazan());
    bord[oyuncu1.LokasyonX()][oyuncu1.LokasyonY() + 1] = 1;
}
if (bord[oyuncu1.LokasyonX()][oyuncu1.LokasyonY() + 2] == 4) {
    oyuncu1.setSkor(oyuncu1.getSkor() + mantar.PuanKazan());
}
```

Gargamel karakterinin 2 birim ilerlemesiyle bazı mekanikleri ile beraber geldi örneğin Azman karakterinin üzerinden zıplamak gibi mekanikler eklenebildi. Bunun yanı sıra sorunlar olarak Gargamel belli bölge dışına çıkamamakta çünkü harita 2 birim ilerlemesi için çok kısıtlı. Bununla birlikte karakterin duvar ,altin,mantar gibi nesneler üzerinden ya da içinden ilerleyemediği için 2 birim giderken ilk gidiceği karelerde bu tür nesnelerin kontrolleri yapılarak hareket edilmektedir.

```
if (bord[gargamel.LokasyonX()][gargamel.LokasyonY() + 1] == 1 |
    bord[gargamel.LokasyonX()][gargamel.LokasyonY() + 1] == 9 |
    bord[gargamel.LokasyonX()][gargamel.LokasyonY() + 1] == 5) {
  if (bord[gargamel.LokasyonX()][gargamel.LokasyonY() + 2] == 1 | bord[gargamel.LokasyonX()][gargamel.LokasyonY() + 2] == 9) {
    gargamel.LokasyonGir(gargamel.LokasyonX(), y: gargamel.LokasyonY() + 2);
    bord[gargamel.LokasyonX()][gargamel.LokasyonY() - 2] = 1;
    sayac1++;
  }
}
```

2.4. Nesne Türleri

2.4.1. MazeBord

Bu nesne türünde oyunun temel mekaniklerinin olduğu ve oyunun kuralları çizilerek oynatıldığı nesne türüdür. Daha önce “Temel Bilgiler” başlığında anlattığım tüm mekanikler burada gerçekleşir. Burası yazılan programın kalbidir. Tüm nesnelerin özellikleri buradan işlenerek depolanmaya tekrar yollanır.

2.4.2.Karakter

Karakter nesnesi “Abstract” yapılı bir nesne türüdür. Bu yapısı sayesinde kendisi tam olarak türetilemez. Alt sınıfları türetilebilir.Bu sınıf “Dusman” ve “Oyuncu” sınıfına miras bırakır tüm özelliklerini. Burada oyundaki karakterlerin “ID”,”AD”,”playerkind” değişkenleri depolar ve işler. İçeride karakterlerin kordinat yapısını da tutar “Lokasyon” nesnesinden yardım alarak.

```
abstract public class Karakter{

    public String ID;
    public String AD;
    public String PlayerKind;
    public Lokasyon lokasyon = new Lokasyon( x_ekseni: 0, y_ekseni: 0);


    public Karakter(String ID, String AD, String playerkind) {
        this.ID = ID;
        this.AD = AD;
        this.PlayerKind = playerkind;
    }

    public Karakter() {
        this.ID = "ID";
        this.AD = "AD";
        this.PlayerKind = "playerkind";

    }
```

Daha sonra “encapsulation” (kapsülleme) yöntemiyle buradaki özellikleri “get” ve “set” metodları sayesinde bilgilerin alınması ve gönderilmesi sağlanır.

```
public String getID() {
    return ID;
}

public String getAD() {
    return AD;
}

public String getPlayerKind() {
    return PlayerKind;
}
```

```
public void setID(String ID) {
    this.ID = ID;
}

public void setAD(String AD) {
    this.AD = AD;
}

public void setPlayerKind(String playerKind) {
    PlayerKind = playerKind;
}
```

Bu nesne “Abstract” yapısını kullanmak için miras bıraktığı nesnelerin de “Abstract” yapıda metod ile o nesnenin alt nesnelerde “Abstract” yapıda metodunun farklı özelliklerle nesneleri özelleştirmesi sağlanılır.

```
abstract public int[][] enKisaYol(int x,int y);
```

2.4.3. Oyuncu

Oyuncu nesnesi “Karakter” nesnesinden miras alır bazı özellikleri kullanılması için. Bunun yanı sıra eğer oyuna farklı bir isimde oyuncu eklenmesi istenirse onun da alt yapısını yaparak “ID” , “AD”,”playerkind” değişkenlerini alarak da

tanımlanabilmekte ama oyundaki hazır karakterler kullanılmasından dolayı hiçbir parametre girilmedende bu nesne oluşturulabilmektedir.

```
public class Oyuncu extends Karakter{

    public int Skor;

    public Oyuncu(String ID, String AD, String playerkind ) {
        super(ID, AD, playerkind);
        this.Skor = 20;
    }

    public Oyuncu() {
        this.Skor = 20;
    }
```

Daha sonra "encapsulation" (kapsülleme) yöntemiyle buradaki özellikleri "get" ve "set" metodları sayesinde bilgilerin alınması ve gönderilmesi sağlanır.

```
public int getSkor() {
    return Skor;
}

public void setSkor(int skor) {
    Skor = skor;
}
```

Nesneyi özelleştiren bazı metodlara sahiptir.Bunlardan birisi "PuanGoster" diğeri "enKisaYol" dur. Bu "enKisaYol" metodu normalde oyuncu sınıfı için kullanılmaz ,kullanılmasının nedeni "Abstract" yapıda bir nesneden miras aldığı için bu metodu kendisi oluşturmas ise kendisi de "Abstract" yapıya dönüşür.Bunun olmaması için "enKisaYol" metodu tanımlanmaktadır.

```
public void PuaniGoster(){
    System.out.println("Oyuncu Skoru : "+this.Skor);
}

public int[][] enKisaYol(int x,int y){
    int distx;
    int disty;

    int [][] kordinatlar = new int[1][2];

    distx = y-lokasyon.getY_ekseni();
    disty = x-lokasyon.getX_ekseni();

    kordinatlar[0][0] = distx;
    kordinatlar[0][1] = disty;


    return kordinatlar;
}
```

2.4.4 .Oyuncu_1

Oyuncu nesnesinden miras alır ve "Gözlüklü Şirin" isminde karakter olarak özelleşir. Bu sınıf oyunda bazı farklarından dolayı ayrı olarak tanımlanır. Böylece oyuncu çeşitliliği sağlanır.

```
public class Oyuncu_1 extends Oyuncu{

    public Oyuncu_1(String ID, String AD, String playerkind) {
        super(ID, AD, playerkind);
    }

    public Oyuncu_1() {
        super( ID: "Oyuncu1",  AD: "Gozluklu Sirin",  playerkind: "Oyuncu1");
    }
```

Nesne kendi içerisinde metod yapısına da sahiptir böylece diğer nesnelerden farklılaştıran özelliklere sahip olurlar. Oyuncu nesnesinden "PuanGoster" metodunu "Override" ederek kendi içinde özelleştirerek kullanır.

```
@Override
public void PuaniGoster() {
    System.out.println("Gozluklu Sirin");
    super.PuaniGoster();
}
```

2.4.5.Oyuncu_2

Oyuncu nesnesinden miras alır ve "Uykucu Şirin" isminde karakter olarak özelleşir. Bu sınıf oyunda bazı farklarından dolayı ayrı olarak tanımlanır. Böylece oyuncu çeşitliliği sağlanır.

```
public class Oyuncu_2 extends Oyuncu{

    public Oyuncu_2(String ID, String AD, String playerkind) {
        super(ID, AD, playerkind);
    }

    public Oyuncu_2() {
        super( ID: "Oyuncu2",  AD: "Uykucu Sirin",  playerkind: "Oyuncu2");
    }
```

Nesne kendi içerisinde metod yapısına da sahiptir böylece diğer nesnelerden farklılaştıran özelliklere sahip olurlar. Oyuncu nesnesinden "PuanGoster" metodunu "Override" ederek kendi içinde özelleştirerek kullanır.

```
@Override
public void PuaniGoster() {
    System.out.println("Uykucu Sirin");
    super.PuaniGoster();
}
```

2.4.6.Dusman

Dusman nesnesi karakter sınıfından miras alınarak oluşturuludu.

```
public class Dusman extends Karakter{


    public Dusman(String ID, String AD, String playerkind) {
        super(ID, AD, playerkind);
    }
```

Kendine ait 2 farkı metod bulunmakta “puanKaybet” ve “enKisaYol” . Bu metodlardan “puanKaybet” dusman türüne göre oyuncudan puan eksiltir,”enKisaYol” metodu ise düşman karakter ile oyuncunun arasindaki yol hesaplamasında kullanılır.

```
public int puanKaybet(){
    return 0;
}

public int[][] enKisaYol(int x,int y){
    int distx;
    int disty;

    int [][] kordinatlar = new int[1][2];

    distx = y-lokasyon.getY_ekseni();
    disty = x-lokasyon.getX_ekseni();

    kordinatlar[0][0] = distx;
    kordinatlar[0][1] = disty;


    return kordinatlar;
}
```

2.4.7.Dusman_1

Kendisi “Dusman” nesnesini miras alır.Bu nesne Azman karakterini temsil eder.Nesne “puanKaybet” metodunu “Dusman” nesnesinden “Override” yaparak kullanır. Bu metodu kendisine özelleştirir.

```
public class Dusman_1 extends Dusman{

    public Dusman_1(String ID, String AD, String playerkind) {
        super(ID, AD, playerkind);
    }

    public Dusman_1() {
        super( ID: "Dusman1",  AD: "Azman",  playerkind: "Dusman1");
    }

    @Override
    public int puanKaybet() {
        return 5;
    }
}
```

2.4.8.Dusman_2

Kendisi “Dusman” nesnesini miras alır.Bu nesne Gargamel karakterini temsil eder.Nesne “puanKaybet” metodunu “Dusman” nesnesinden “Override” yaparak kullanır. Bu metodu kendisine özelleştirir.

```
public class Dusman_2 extends Dusman{

    public Dusman_2(String ID, String AD, String playerkind) {
        super(ID, AD, playerkind);
    }

    public Dusman_2() {
        super( ID: "Dusman2",  AD: "Gargamel",  playerkind: "Dusman2");
    }

    @Override
    public int puanKaybet() {
        return 15;
    }
```

2.4.9.Obje

Bu nesne kendisine ait özel değişkeni yoktur ve kimseden miras almaz.Kendisi “PuanKazan” metoduna sahiptir bu metod ile kendisinden miras alan nesnelere özelleşmesi için yardımcı olur.

```
public class Obje {

    public Obje(){
    }

    public int PuanKazan(){
        return 0;
    }
}
```

2.4.10.Altin

Bu sınıf “Obje” nesnesinden miras alır ve böylece onun içindeki “PuanKazan” metodunu “Override ” yaparak 5 puan eklenmesi için özelleştirerek bu metodu kullanır.

```
public class Altin extends Obje{

    public Altin() {

    }

    @Override
    public int PuanKazan() {
        return 5;
    }
```

2.4.11.Mantar

Bu sınıf “Obje” nesnesinden miras alır ve böylece onun içindeki “PuanKazan” metodunu “Override ” yaparak 15 puan eklenmesi için özelleştirerek bu metodu kullanır.

```
public class Mantar extends Obje{

    public Mantar() {

    }

    @Override
    public int PuanKazan() {
        return 50;
    }
}
```

2.4.12.Lokasyon

Lokasyon nesnesinin kendisine ait 2 tane değişkeni vardır ve bu değişkenler sayesinde “x_ekseni” ve “y_ekseni” kordinatları buralarda depolanır.Nesneyi tanımlarken x ve y parametreleri girilerek nesnenin oluşturulması gerekir.Kapsülleme yöntemi ile “get” ve “set” metodları kullanılmış olur “getX_ekseni” , ”getY_ekseni” , ”setX_ekseni”,”setY_ekseni” metodlarını kullanarak.

```
public class Lokasyon {

    private int x_ekseni;
    private int y_ekseni;

    public Lokasyon(int x_ekseni, int y_ekseni) {
        this.x_ekseni = x_ekseni;
        this.y_ekseni = y_ekseni;
    }

    public Lokasyon() {
        this.x_ekseni = 0;
        this.y_ekseni = 0;
    }

    public int getX_ekseni() {
        return x_ekseni;
    }

    public int getY_ekseni() {
        return y_ekseni;
    }

    public void setX_ekseni(int x_ekseni) {
        this.x_ekseni = x_ekseni;
    }

    public void setY_ekseni(int y_ekseni) {
        this.y_ekseni = y_ekseni;
    }
```

2.4.13.Baslat

Bu sınıfın yardımı ile “MazeBord” nesnesi oluşturulup çağırılması ve oyunun başlanması sağlanır.

```
import javax.swing.*;


public class Baslat extends JFrame {


    public Baslat() {
        add(new MazeBord());
    }


    public static void main(String[] args) {
        MazeBord d = new MazeBord();


    }
```

3.ALGORİTMANIN ZAMAN VE BELLEK ANALİZİ

Zaman karmaşıklığı(Time Complexity) , bir programın ya da fonksiyonun işlevini tam anlamıyla yerine getirebilmesi için her işlemden kaç kere yapması gerektiğini gösteren bir bağıntıdır.Bu uygulamamızdaki zaman karmaşıklığı konusunda “Timer” ’lar kullandığımız ve sonsuz döngüler olduğu için tam olarak big O etkisinin ne olduğunu bilemiyoruz ama “Timer” ‘ ların içinin BigO ‘su O(1) olarak düşünüyoruz.Bazı “Timer” algoritmalarımda da harita eni ve boyu (MXN) döngümüzün boyutlarını belirlediği fakat bu projede harita daima 11X13(MXN) sayılarla döndüğü için döngüler O(1) olarak kabul ediyoruz.

```
for (int i = 0; i < bord.length; i++) {
    for (int j = 0; j < bord[0].length; j++) {
```

Alan karmaşıklığı(Space Complexity),Bir algoritmanın verilen bir girdi için çıktı üretirken kullandığı veya ihtiyaç duyduğu bellek alanıdır.Amacımız hızlı olup çok belleğe sahip olmasıydı algoritmamızın .

```
static int odie = 0, k = 0, k1 = 0, temp_k = 0, temp_k1 = 0, check_azman = 1, check_gargamel = 1, check_oyuncu = 1, bitti = 0;
static int hamle = 0, eskiX = 0, eskiY = 0, sayacAltin = 5, sayacMantar = 1, secim = 0, path_secim = 0, dusman_hareket = 0;
static int[][] bord = new int[11][13];
static int random1, random2;
static boolean game = true;
static int[][] sifirla = new int[28][2];
static int[][] sifirla2 = new int[28][2];
static int scale = 50;
static int[][] kordinat_kisayol = new int[1][2];
static int first_X, first_Y, first_X_a, first_Y_a, mod = 0;
public Image mantar, altin, gozluklu, uykucu, sirine, gargamell, azmann, background, gozluklu_bg, uykucu_bg;
```

4.DENEYSEL SONUÇLAR

5.KARŞILAŞILAN SIKINTILAR VE ÇÖZÜMLERİ

5.1.Karakterlerin Sıra Tabanlı İlerlemesi

Oyunumuzu ilk olarak karakterlerin gerçek zamanlı ilerlemeli şekilde yaptık.Fakat proje dosyasında bizden istenenleri tekrar okuduğumuzda sıra tabanlı olucağını fark ettik ve düşman karakterinin hareketini kontrol eden "Timer" 'ların içine koşul durumu koyduk ve karakterin ilerleme bilgisini değişkene atadık. Böylelikle karakter hareket edince düşmanların hareket etmesini sağladık.

```
if (dusman_hareket == 1) {
```

5.2.Düşman Karakterinin Gidiceği Yolları Çizmek

Düşman karakterleri gidiceği yönü "drawPath" metodu yardımıyla çizdiriyoruz . Buradaki sıkıntı grafik çizimini önceden yaptığım için çizilen yolun altında duvarların ve altınların kalması gibi bir grafik hatasına sahip bu durum ilk önce yolun çizilmesi sonra haritanın çizilmesi sağlanabilirdi fakat önümüzde çıkıcak hatalar ve sürenin kısıtlı olması nedeniyle bu hatayı düzeltemedik.

5.3.Dosya Okumada Space Karakteri Okumaya Çalışmak

Dosya okurken haritadaki değerleri alıcağımız sırada "Space" karakteri olduğunu düşünüyordum düşünüyordum fakat o karakterler "TAB"(\t) karakteri olduğunu öğrenip ona göre okuma yaptığım için harita okumayı başarabildim.

```
// İlgili satırdaki TAB karakterlerini siliyor.
String a = satirList.get(i).replace( target: "\t", replacement: "");
```

5.4.Timer Sürelerinin Mantar ve Altındaki Sorunu

İlk olarak mantar ve altin nesnelerinin haritada rasgele belirirken eski nesnelerin silinmeyip haritanın her yerini altin ve mantar kaplaması sorunu aldım. Bu sorun obje belirmesini ilk başta yapıp ertelemeli olarak objeleri tekrardan sıfırlama sürelerini ardından yapmasını programladım örneğin altınlar program başlar başlamaz beliriyor daha sonra programda 5 saniye periyotlar halinde devam ederken 5 saniye 50 milisaniye gecikmeler olucak şekilde sıfırlamaları yapıp bu sorunu hallettik.

```
myTimer.schedule(gorev, delay: 0, period: 5050);
```

```
myTimer.schedule(gorev2, delay: 5000, period: 5000);
```

6.SONUÇ

Bu projede "Timer" 'ların nasıl kullanıldığı ve algoritmaların nasıl sırayla başlatılıp hangi algoritma altında devam ediceğini çok kullanarak öğrendik. Bununla beraber arayüz kullanımı ,veri yapıları , nesnelerin birbiriyle ilişki kurması konusunda tecrübeler edindik. Veri yapılarının bu projede tam anlamıyla kullanamadık bunu projeye uyarlamada sorun yaşadık ama araştırma yaparak öğrenimi tamamlayabildik bununla tecrübe kazandığımızı düşünüyoruz.

7.REFERANSLAR

7.1.Oyun Arayüzü İle İlgili

https://www.youtube.com/watch?v=ATz7bIqOjiA

https://www.youtube.com/watch?v=r7i25lbmBd4

7.2.Veri Yapıları İle İlgili

https://www.happycoders.eu/algorithms/shortest-path-algorithm-java/

https://medium.com/@urna.hybesis/pathfinding-algorithms-the-four-pillars-1ebad85d4c6b

7.3.Nesneye Yönelik Programlama İle İlgili

Udemy-Tim Buchalka

https://www.udemy.com/course/java-the-complete-java-developer-course/

7.4.Akış Şeması Çiziminde Kullanılan Kaynak

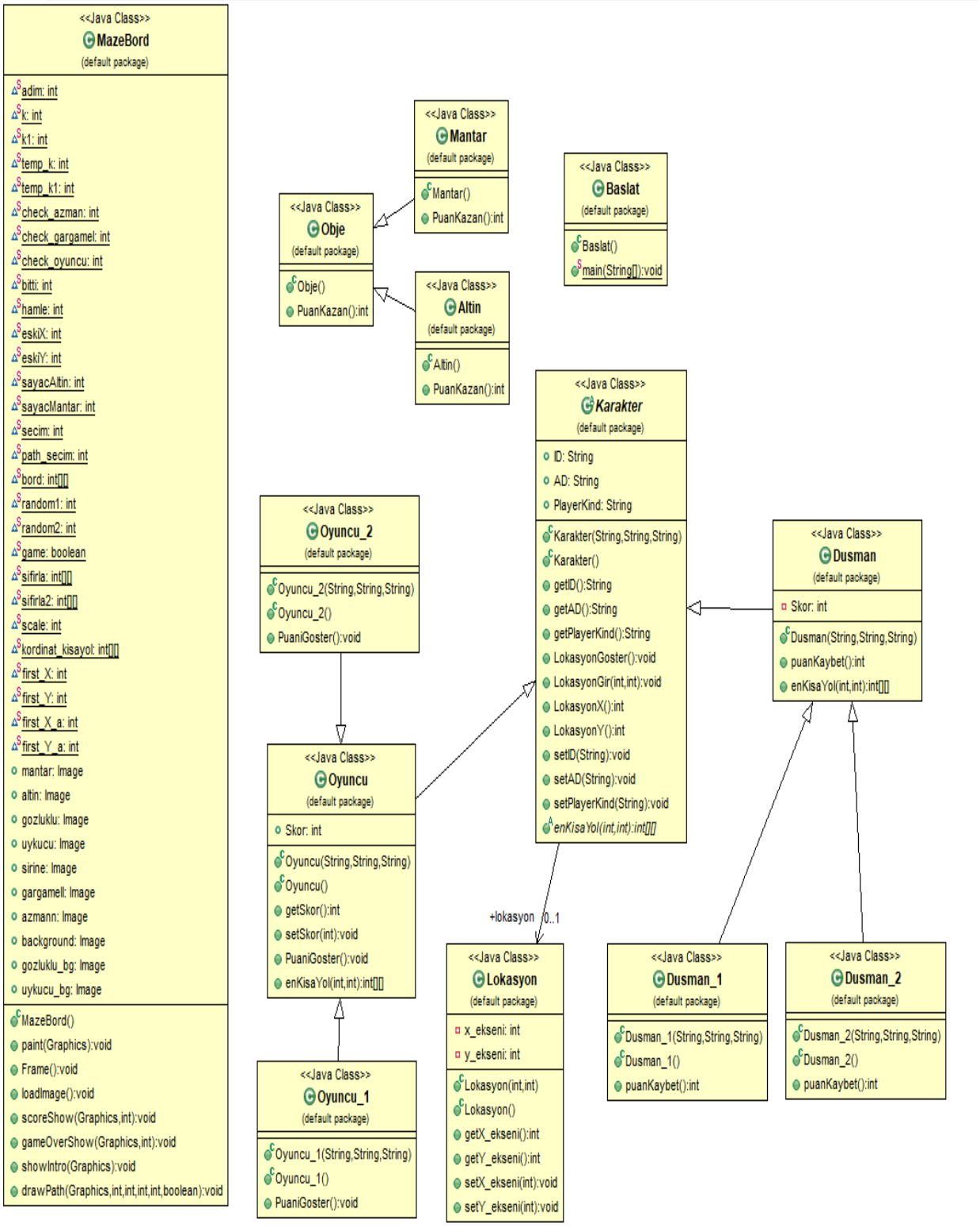
https://app.diagrams.net/

7.5.Klavye Komutları İle İlgili

https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyListener.html

https://www.codota.com/code/java/methods/java.awt.Component/addKeyListener

8.UML SINIF DİAGRAMI



9.AKIŞ ŞEMASI