**Kocaeli Üniversitesi**

**Bilgisayar Mühendisliği Bölümü**

**Programlama Laboratuvarı II**

**Şirinler Oyunu**

*Oğuz Narlı Alperen Tan*

[*180201074@kocaeli.edu.tr*](mailto:180201074@kocaeli.edu.tr)[*190201054@kocaeli.edu.tr*](mailto:190201054@kocaeli.edu.tr)

**1.Giriş:**

Raporumuz Google Döküman kullanılarak yazılmıştır. Projemiz Java programlama diliyle arayüz ise Swing yapısıyla geliştirilmiştir. Bu proje ile ilgili UML diyagramı, yalancı kod ve örnek çıktılar bölüm sonunda belirtilmiştir.

**2. Özet:**

Programlama Laboratuvarı dersimiz kapsamında ilk proje ödevi olarak bizden şirinler oyunu geliştirmemiz beklenmektedir. Oyundaki amacımız ise oyunun başlangıcında seçtiğimiz 2 şirinden (Tembel ve Gözlüklü) biri ile Şirineye çeşitli zorlukları aşarak puanımız sıfırlanmadan erişmek. Bahsettiğimiz zorluklar Gargamel ve Azman, Şirineye ulaşmamıza engellemek için bizi kovalamaktadır. Gargamel ve Azman bize erişmesini sağlamak için Graf Veri Yapısı içeriğinde yer alan, en kısa yol hesabı yapmamızı sağlayan Dijkstra Algoritması kullanılması beklenmektedir. Bu zorluklardan yanı sıra bizden oyuncumuzun puanlarını arttırmamızı sağlayacak, belli sürelerde gözüken mantar ve altın objeleri de oluşturmamız istenmektedir. Bu işlemleri gerçekleştirdiğimiz tasarım ve algoritma aşağıdaki bölümlerde bahsedilmiştir.

**3. Dosyadan Veri Okuma**

Okuyacağımız dosya içeriğinde aşağıda verdiğimiz harita Azman ve Gargamelin başlangıç noktalarıdır. Dosyadan okuyacağımız veri aşağıda verilen formattadır:

*Karakter:Gargamel,Kapi:A*

*Karakter:Azman,Kapi:D*

*0 0 0 1 0 0 0 0 0 0 1 0 0*

*0 1 1 1 1 0 1 1 1 1 1 1 0*

*0 1 0 1 1 1 1 1 1 0 0 1 0*

*0 1 1 1 1 0 1 0 1 1 0 1 0*

*0 1 0 1 0 0 1 0 1 0 0 1 0*

*1 1 0 1 1 1 1 0 1 0 1 1 0*

*0 1 0 0 1 1 1 0 1 1 1 1 0*

*0 1 0 1 1 1 1 1 1 1 1 1 1*

*0 1 0 1 0 0 0 0 0 1 1 1 0*

*0 1 1 1 1 1 1 1 1 1 1 1 0*

*0 0 0 1 0 0 0 0 0 0 0 0 0*

Dosyada karakterle veriyi ayırmak için “kontrol” ve “step” isimli tamsayı değişkeni oluşturduk. Bunu kullanma sebebimiz şudur. Bizden karakterin dinamik oluşturulması istenmektedir. Yani bir olmama ihtimaline göre değiştirmektedir.

Step her satır okuduğunda kontrol ise 2.satır karakter ise 1 artacaktır. İlk satırda karakter kesinlikle içerecektir. Bu yüzden bu yapılarımızı 2. satırdan itibaren kullandık. Eğer step 1 iken kontrol 0 olduğunda satırımızın ilk karakteri ‘K’ harfi iste karakter olacaktır ve kontrol 1 artacaktır. Her satırı okuduğumuzda karakter ise karakteri içeren stringi ayırır ve kendisi ile ilgili objesine aldığımız özelliklerini atarız. Eğer harita değeri ise oluşturduğumuz 11’e 13’lük matrise atarız.

**4. Dijkstra Algoritması (En Kısa Yol Hesabı)**

Yukarıdaki haritada kullanacağımız yol üzerindeki 1 numaralı yollardır. Bu yüzden 1 numaralı yollar üzerinde graf düğümleri oluşturalım. Bunun için içerisinde satır, sütun, uzunluk (başlangıçta sonsuz olduğundan -1 değerinde), komşu düğümleri içeren 4 uzunlukta Edge türünde dizi(->,<-,d,u) ve düğüme kadar alınan en kısa yolu içeren Edge sınıfı tanımladık. Sonrasında Edge türünde arraylist yapısı oluşturduk ve harita üzerinde dolaşarak her 1 gördüğümüzde yeni düğüm oluşturduk arrayliste atadık. Daha sonra ilgili arrayliste iç içe for-each döngüsü kurarak düğümüzün komşuluklarını atadık. Şimdi esas kısım olan kısa yolun hesaplanmasına gelelim. Bunun için recursive fonksiyon geliştirdik. İlk olarak arraylist üzerinden başlangıç düğümümüzü seçtik. Ardından o düğünümüzün uzunluğunu sıfırladık. Sonrasında fonksiyonumuza parametre olarak yolladık. İlk olarak fonksiyonumuz komşularını kontrol edecek, eğer komşulukları Null değer taşımıyorsa onlar üzerinde işlem gerçekleşecek. Komşu değerin uzunluğu -1 ise sonsuz demektir ve onu 1 yapıp o komşu düğümümüzün en kısa yolu içeren düğüm arraylistimize mevcut düğümün koordinatları “x,y” olacak şekilde atanacak ardından fonksiyonumuz öz yinelemeli şekilde komşu düğüm parametre olarak kullanılarak çağırılacak. Ancak komşu düğümümüz uzunluğu sonsuz değil yani -1 değilse öncelikle mevcut düğümün uzunluğunun 1 fazlası ile komşunun uzunluğu karşılaştırılacak eğer komşunun uzunluğu büyükse mevcut düğümün en kısa yol arraylistinin içeriği komşununkinin içeriğine eşitlenecek. Ardından mevcut düğümün koordinatları komşunun arraylistine eklenir. Arraylistin uzunluğu da, uzunluğuna eşitlenir. Böylece başlangıç bölümünden tüm diğer yollara en kısa gidiş bulunur. Şimdide fonksiyonumuzun algoritmik yapısını yalancı kodda görelim:

dijkstra(baslangıç){

eğer komşu[0,1,2,3]!=Null(0,1,2,3 ayrı ayrı if yapısında kontrol edilcek ){

eğer komşu[0].uzunluk=-1{

komşu[0].eky.add(başlangıçx,y);

komşu.uzunluk=1;

dijksta(komşu[0,1,2,3]);}

eğer başlangıç.komuşu[0].uzunluk>-1{

eğer başlangıç.uzuluk+1<komşu.uzunluk{

komşu[0,1,2,3].eky=başlangıç.eky;

komşu.uzunluk=başlangıç.uzunluk+1;

dijksata(komşu[0,1,2,3]);}}}}

Algoritmamızın karmaşıklık hesabı:

Fonksiyonumuz n kez döneceğini farz sayalım. İlk başta 4 farklı komşu için null olup olmama durumunu kontrol edecek dolayısıyla 4 farklı if kontrol edilecek. O if lerin içerisinde de biri sonsuz uzunlukta olan için diğeri ise daha kısa olup olmadığını incelemek için 2 tane akış kontrol yapısı içermektedir. Bunların içerisinde de bir tane k kez döndüğünü farz ederekten birer döngü içermekte. Ayrıca döngü dışında 3 tane işlem gerçekleşmektedir. O zaman null kontrolü içerisindeki akış kontrol yapıları içink+3 işlem gerçekleşmektedir ve ekstra bir tane daha if else kontrolü olduğundan k+4 olcaktır.4 farklı durum için bu kontrol olacağından, fonksiyonumuz 1 kez özyineleme gerçekleştiğinde turladığında 4x(k+4) tane işlem olacak. Biz bunu n kez yaptığını farz saydığımızdan bu sayısı n le çarpacağız. Sonuç ise 4nk+4n olcaktır.4nk sayısı 4n sayısına göre çok büyük olduğundan 4n sayısı göz ardı edilecek. Bu yüzden karmaşıklık 4nk’dır.

**5. Karakter Sınıfı**

Karakter sınıfımız oyunda kullanacağımız karakterlerin özelliklerini taşımaktadır. Bu özellikler; Id(int), Isim(String), tur(boolean(false- oyuncu/true-dusman)), konum(Konum). Bu özellikleri atama ve çağırılması için yapıcı metot, setter ve getter metotları yer almaktadır. Setter getter metotlarımız diğer Oyuncu ve Dusman sınıfında içeriği düzenlenmesi için abstract olarak tanımlanmıştır. Ayrıyeten düsman sınıfımız oyuncumuza erişmek için önceki bölümde bahsettiğimiz Dijkstra Algoritması kullanarak en kısa yol hesabını sağlayan calculateDistance() fonksiyonunu da içermektedir. Bu sınıfımız direk kullanılmaması ve yalnızca kendisini extend eden sınıflar tarafından kullanılması için abstract olarak tanımladık. Bu sınıfımızı extend eden sınıflar Oyunucu ve Dusman sınıflarıdır.

**6. Konum Sınıfı**

Konum sınıfımız uygulama boyunca kullandığımız koordinatları tutmak için oluşturulmuştur. Az önce belirttiğimiz Karakter sınıfında da özellik olarak da tanımlanmıştır. Sınıfımızda koordinatları tutan x ve y (int) değerleri vardır. Bununla beraber konum özelliklerini atamamızı sağlayan yapıcı metot setter ve getterları da içermektedir.

### 7. Oyuncu Sınıfı

Bu sınıfta oyuncumuzun özellikleri belirtilmiştir.  Özellikleri Karakter sınıfından extend edilmiştir. Özellikleri: skor (int), oyuncuId (int), oyuncuIsim (String), oyuncuTur (boolean). Bunlarla beraber Gozluklu, Tembel, Puan olmak üzere 3 tane alt sınıfa sahibiz.2 tane yapıcı metotla beraber Karakter sınıfından override ettiğimiz setter getter, mevcut sınıfın sahip olduğu skorGoster setSkor fonksiyonu içermektedir. Override ettiğimiz setter getter fonksiyonları bu sınıfımızın özellikleri üzerinde işlem yapacak şekilde yazdık. Şimdi de alt sınıflarımızdan bahsedelim.

#### **7.1 Gozluklu Sınıfı:**

Gozluklu sınıfımız oyuncumuzun seçebileceği 2 karakterden biri olan Gözlüklünün özellikleri taşımaktadır. Özelliklerini İçinde yer aldığı Oyuncu sınıfından extend etmektedir. Ayrıca yapıcı metot içermektedir. Bu metot içerisinde başlangıç noktasını bizden istenildiği gibi 6,5 (x,y şeklinde) olarak ayarlamaktadır. Ek olarak ilerle isimli fonksiyon içermektedir. Bu fonksiyon oyuncumuzun konumunu güncellemektedir. Parametre olarak yön (1(sağ), 2(sol), 3(aşağı), 4(yuk.)) ve haritayı içeren matrisi almaktadır. Bu fonksiyonumuzda 4 tane if vardır ve aldığı yön parametresine göre bu if yapılarından birine girecektir. Bu yapılarımızın içerisinde de if else bulunmaktadır. Eğer haritamızda 2(Gozluklu 2 adım gidebiliyor) adım hareket ettiğinde harita dışında değilse şirinimizin konumu 2 birim kadar güncellenir, eğer kalırsa 1 birim kadar güncellenecek.

#### **7.2 Tembel Sınıfı:**

Tembel sınıfı özellikleri Gozluklu gibi Oyuncu sınıfından extend etmektedir. Sahip olduğu yapıcı metot da aynı şekilde 6,5 olarak ayarlamaktadır. Sahip olduğu ilerle fonksiyonu ise Gozluklu’den farklı olarak sadece yon parametresi almaktadır. Aldığı yön parametresine göre if yapısı yönü kontrol edilerek ilerleyeceği yönde konumu 1 birim kadar güncellenecektir.

**8. Dusman Sınıfı:**

Bu sınıfta düşmanlarımızın özellikleri belirtilmiştir. Özellikleri Karakter sınıfından extend edilmiştir. Özellikleri: skor (int), dusmanId (int), dusmanIsim (String), dusmanTur (boolean). Bunlarla beraber Gargamel, Azman olmak üzere 2 tane alt sınıfa sahibiz.2 tane yapıcı methola beraber Karakter sınıfından override ettiğimiz setter getter içermektedir. Override ettiğimiz setter getter fonksiyonları bu sınıfımızın özellikleri üzerinde işlem yapacak şekilde yazdık. Şimdi de alt sınıflarımızdan bahsedelim.

**8.1 Gargamel Sınıfı:**

Bu sınıfımız içinde bulunduğu sınıftan özelliklerini extend etmiş, bx, by başlangıç konumunu tutan özelliği ve 2 fonksiyon içermektedir. İlki paint fonsiyonu. Bu fonksiyon kenarları içeren arraylist ve oyuncu almaktadır. İlk olarak karakter sınıfından calculateDistance fonksiyonunu çağırarak haritadaki her noktaya kısa yollar güncellenir. Ardından graf kenarları içeren arraylisten oyuncumuzun koordinatına denk gelen kenarda bulunur. Bu kenarın içerdiği path özelliğinden gidiş yolu alınır ayrıştırılıp tam sayıya çevrilir(String olarak yer almakta).Bu tam sayı değerlerinde int[](0.x,1.y) tipinde arrayliste atanır. En sonunda bu arraylist döndürülür. Bu arraylisteki değerlere göre arayüzde gidilecek yol boyanır. 2. fonksiyonumuz ise ilerle fonksiyonudur. Bu fonksiyon kenar arraylistimiz ve oyuncu nesnesini parametre olarak almaktadır. Arraylistemizdeki kenarlara gidilecek kısa yol önceki fonksiyonla beraber güncellenmişti. Bu arrayliste dolaşacağız. Eğer geldiğimiz kenar oyuncuya aitse işlemlerimizi gerçekleştireceğiz. Öncelikle eğer path 3 ten küçükse hemen şirinimize ulaşacağımızdan konumu düşmanımızın konumu bx by eşitlenir ve oyuncunun puanı 5 azaltılır. Eğer değilse ilk olarak path özelliğimiz 1. ve 2. indeksteki koordinatları ayırır ve tam sayıya çevirir. Ardından if else ile Gargamelin ilerleyişi incelenir. Eğer 2 koordinatımız aynı doğrultudaysa düşmanın konumu 2. indekstekine göre güncellenir.2. indeksteki değerler şirin üstündeyse şirinin puanı azaltılır Gargamelin konumu başlangıca göre ayarlanır, değilse 2. indeksteki değerler yeni konumu olacaktır. Fakat aynı doğrultuda değillerse aynı işlemler 1. indeksteki değerlere göre yapılır.

**8.2 Azman Sınıfı**

Bu sınıfımız içinde bulunduğu sınıftan özelliklerini extend etmiş, bx, by başlangıç konumunu tutan özelliği ve 2 fonksiyon içermektedir. İlki paint fonksiyonu. Bu fonksiyon kenarları içeren arraylist ve oyuncu almaktadır. İlk olarak karakter sınıfından calculateDistance fonksiyonunu çağırarak haritadaki her noktaya kısa yollar güncellenir. Ardından graf kenarları içeren arraylisten oyuncumuzun koordinatına denk gelen kenarda bulunur. Bu kenarın içerdiği path özelliğinden gidiş yolu alınır ayrıştırılıp tam sayıya çevrilir (String olarak yer almakta). Bu tam sayı değerlerinde int[](0.x,1.y) tipinde arrayliste atanır. En sonunda bu arraylist döndürülür. Bu arraylisteki değerlere göre arayüzde gidilecek yol Azman için boyanır. 2. fonksiyonumuz ise ilerle fonksiyonudur. Bu fonksiyon kenar arraylistimiz ve oyuncu nesnesini parametre olarak almaktadır. Arraylistemizdeki kenarlara gidilecek kısa yol önceki fonksiyonla beraber güncellenmişti. Bu arrayliste dolaşacağız. Eğer geldiğimiz kenar oyuncuya aitse işlemlerimizi gerçekleştireceğiz. İlk olarak path özelliğimiz 1. indeksteki koordinatları ayırır ve tam sayıya çevirir. Ardından aldığımız koordinat oyuncuya eşitliği incelenir. Eğer eşitse Azmanın konumu başlangıca alınır ve oyuncunun puanı 5 azaltılır. Değilse yeni konum 1. indeksteki değer olur.

**9. Obje Sınıfı**

Obje sınıfımız oyunda bize puan kazandıracak mantar ve altının özelliklerini içermektedir. Bu sınıf özellik olarak Mantar ve Altın olmak üzere iki sınıf içermektedir. Bu sınıfların içerisinde rastgele fonksiyonu içermektedir. Bu fonksiyon objelerimizi haritada karakterlerin üstüne denk gelmeyecek şekilde rastgele noktalarda oluşturacak. Bu sınıflarımız içerisinde de konumları tutan oluştukları konumu tutan konum dizisi içermektedir. Altın için 5, mantar için 1(ne yazık ki yaşadığımız hatayı anlamadığımız şekilde böyle tanımlayarak düzeldi) uzunluktadır. Fonksiyonumuz graf kenarlarını tutan edges ve karakterleri tutan karakter dizisini (0.Gar,1.Azm, 2.Goz, 3.Tem) parametre olarak almaktadır. Fonksiyonlarımızın içeriği şu şekildedir: İlk olarak haritada oluşacak obje sayısı kadar yani az önce bahsettiğimiz dizimizin uzunluğu kadar for döngüsü dönmektedir. O döngünün içerisinde ise her tur attığından kenar sayısı kadar rastgele sayı oluşturulacak. O sayının edges deki indeksindeki kenarın koordinatı alınır sonra konum dizimizde olup olmadığının karakterin olduğu koordinatına denk eşi olup olmadığı kontrol edilir. Bu işlemde for gene for içerisinde yapılır ve dizi kadar turlar. Eğer varsa üst döngünün tur değeri 1 azaltılır dizimize konum eklenmez eğer yoksa konum dizimize eklenir.

**10. Main Sınıfı**

Bu sınıf, JFrame’i extend ederek oyunun gösterileceği pencerenin özelliklerini (isim, boyut, açılış konumu vs.) belirleyip pencereyi açtık. Açılan pencereye de Oyun adlı JPanel classını ekler. Ayrıca readmap(); fonksiyonunu kullanarak harita verisini map adlı 2 boyutlu arrayde tutttuk. Getter ve setter metotları ile classlar arası veri akışını sağladık.

**11. Oyun Sınıfı**

Bu sınıfımız ise JPanel’i extend edip klavye tuşlarına basıldığında gerçekleştirilecek aksiyonlar için KeyListener ve ActionListener fonksiyonuna ekler. Oyunumuzun görüntüsünü 50 milisaniyede bir yenilemek için timer isimli bir timer oluşturup bu classımıza ekledik. Ardından ekrandaki altın ve mantarın rastgele saniyelerde gösterilmesi için rand isimli değişken ekledik. Classımızda ilk olarak readMap() fonksiyonu ile belirtilen dosya yolundaki txt dosyasını okuyup, düşman sayısına, düşman konumunu ve 2 boyutlu array şeklinde verilen haritayı okuyup değişkenlere aktardık. Ardından tur değişkenini tanımlayıp oyuncunun her hareketi ile bir artan tur sayısını tuttuk. Constructor içinde ise img klasörüne yerleştirdiğimiz resimleri okuyup her bir objenin değişkenine aktardık. Son olarak zamanlayıcıyı başlatıp oyun başlar başlamaz bir pencere açılmasını sağlayıp kullanıcıya hangi karakteri seçeceğini sorduk. Aldığımız cevabı bazı boolean değişkenleri üzerinde tuttuk. Ardından paint fonksiyonu ile oyun devam ediyorsa sırasıyla haritayı, puan ve skoru, altın ve mantarı, son olarak oyuncuyu ekrana çizdirdik. Oyun devam etmiyorsa yani bitmişse ekranı temizleyen fonksiyonu çağırdık. Paint fonksiyonunu repaint(); fonksiyonu ile timer sayesinde actionPerformed fonksiyonu içerisinde her 50 milisaniyede bir çağırdık. drawMap(); fonksiyonu ile dosyadan okunan veriye göre 30’a 30 boyutunda ekrana kare çizdirdik. Verimiz 0 ise yani duvar ise siyah, 1 ise yani hareket edilebilir bir alan ise beyaz renkte kare çizdirdik. Alanlar arasında fark belli olsun diye çizilen karelerin etrafını siyah çizgiler ile belirttik. Ek olarak düşmanların oyuncuya gideceği yolu farklı renklerle arraylist değişkenlerinde tutarak ekranda gösterdik. Son olarak bu fonksiyonda başlangıç koordinatı olarak verilen alanı mavi renk ile gösterdik. drawClearMap(); fonksiyonu ile ilk olarak bütün ekranı temizledik. Ardından kullanıcının seçtiği şirin karakterine göre belirlenen koşullara uyarak ekrana “Kaybettiniz” ya da “Kazandınız!” yazdırdık. drawPuanveSkor(); fonksiyonu ile ekranın en alt kısmına oyuncunun puanını gösterdik. 20 puan ile başlayan oyuncu hareket ettiği her turda 1 puan kaybettiği için skordan tur sayısını çıkarmamız bu fonksiyon için yeterli oldu. drawAltin(); ve drawMantar(); fonksiyonlarında ise öncelikle System.currentTimeMilis(); ile fonksiyonlar çağrıldığındaki sistem zamanını çağırdık. “bastime” ve “existancetime” değişkenleri ile objelerin ekranda maksimum görünme saniyesini geçmemesini sağladık. Oyun başlar başlamaz belirlenen altın için 10000, mantar için 20000 milisaniyeye kadar olan random sayıya göre ekrana altın ve mantarı çizdirdik. Bu fonksiyonlar sonunda altın ve mantar için tekrar random milisaniye belirleyerek fonksiyonların devamlılığını sağladık. Son olarak processKeyEvent fonksiyonu içerisinde klavyede basılan yön tuşuna göre oyuncun ve düşmanın hareket etmesini, altın/mantar bulunan bloğa geldiğinde oyuncunun skorunun artmasını sağladık.

**Sonuç:**

Bu projede Java programlama diliyle 2 boyutlu oyun kodlamanın temelini ve mantığını, Dijkstra algoritması ile iki nokta arasındaki en kısa yolu bulmayı kodlayarak öğrendik.

**Yalancı kod:**

Main ekran=new ekran;

Ekran.setsize;

Oyun oyun=new oyun;

Ekran.addkeylistener(oyun);

Ekran.add(oyun);

int[][] map=Oyun.readMap;

Timer timer=new Timer(50); //50 msde bir gerçekleşen timer oluştur;

Random rand=new Random;

Altın altin = new altin;//altın objesi oluştur

Mantar mantar = new mantar;//mantar objesi oluştur

Oyuncu.gozluklu gozluklu = new Oyuncu.gozluklu;

Oyuncu.tembel tembel = new Oyuncu.tembel;

Dusman.Azman azman = new Dusman.azman;

Dusman.gargamel gargamel = new Dusman.gargamel;

//Oyuncuları ve düşmanı oluştur

int running=false;//oyunun çalışıp çalışmadığını kontrol eden kod;

static int[][] readMap //haritayı oku;

running=true//oyunu başlat;

Resimleri dosyadan okuyup değişkene ata;

timer.start//timer’ı başlat;

Seçim ekranını oyuncuya göster;

Oyun çalışır durumdaysa {

Haritayı çiz (1 için siyah kare, 0 için beyaz kare)

Düşmanların kullanıcıya olan yolunu çiz;

Oyuncunun skorunu ekranın altında çiz {

puan 0 ise running=false;

Kullanıcı şirineye ulaşırsa running=false;}

Altın ve mantarı rastgele aralıklarla belli bir saniye boyunca ekrana çiz;

Kullanıcıyı ve düşmanı ekrana çiz;}

Oyun çalışır durumda değilse {

ekranı temizle;

kullanıcı kazandıysa kazandı, kaybettiyse kaybetti yazdır;

}

kullanıcı yukarı tuşa basarsa{

gideceği yer duvar değilse{

altın veya mantar içeriyorsa puanını arttır;

gözlüklü ise 2 adım yukarı git;

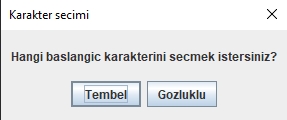
tembel ise 1 adım yukarı git;

gargamel ve azmanı ilerlet;}

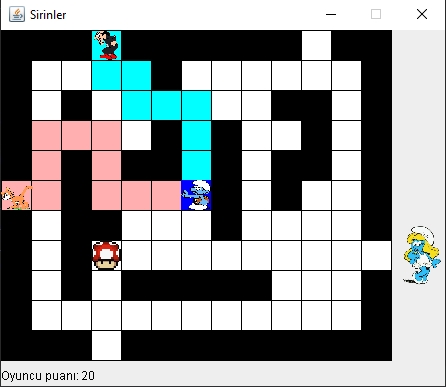
Yukarıdaki işlemi her tuş için gerçekleştir;

işlemi bitir;

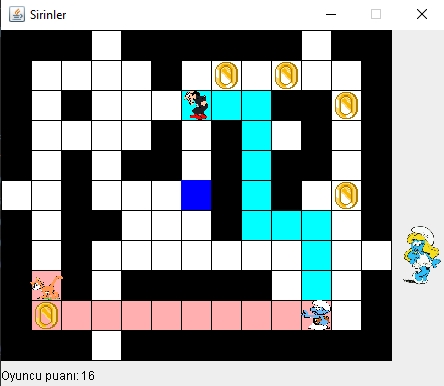
**Deneysel Sonuçlar:**



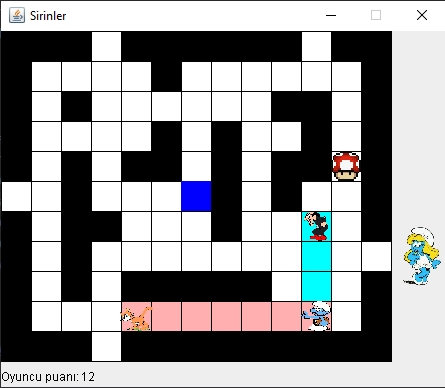
*Karakter seçim penceresi*



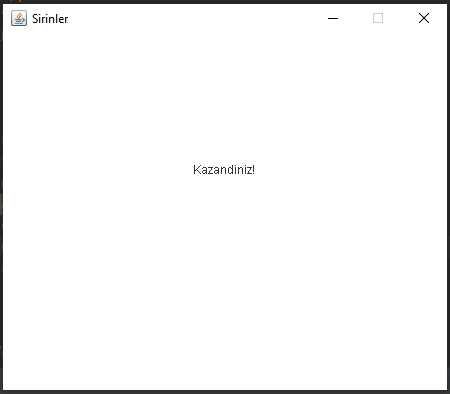
*Oyun ekranı*



*Altınlar*



*Mantar*



*Oyun bitiş ekranı*

**Kaynakça:**

*-*[*https://stackoverflow.com/questions*](https://stackoverflow.com/questions)

*-*[*https://www.udemy.com/course/sifirdan-ileri-seviyeye-komple-java-gelistirici-kursu*](https://www.udemy.com/course/sifirdan-ileri-seviyeye-komple-java-gelistirici-kursu)

*-*[*https://docs.google.com/presentation/d/e/2PACX-1vQD849Ts0E3yea5g7GlPDXe281HTOXkbbmUNbXaHYfMunaTodQG\_QVopY49IgozvljcZgQYLh8dOOXs/pub?start=false&loop=false&delayms=60000&slide=id.g2a53650c7a\_1\_45*](https://docs.google.com/presentation/d/e/2PACX-1vQD849Ts0E3yea5g7GlPDXe281HTOXkbbmUNbXaHYfMunaTodQG_QVopY49IgozvljcZgQYLh8dOOXs/pub?start=false&loop=false&delayms=60000&slide=id.g2a53650c7a_1_45)

[*http://bilgisayarkavramlari.com/2010/05/13/dijkstra-algoritmasi-2*](http://bilgisayarkavramlari.com/2010/05/13/dijkstra-algoritmasi-2)

**UML Diyagramı:**

