ŞİRİNLER

Hasan ÇOLAK – Teyfik CANER

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

hasancolk99@gmail.com – teyfikcaner@gmail.com

ÖZET

Nesneye yönelik programlama ve veri yapıları algoritmalarını kullanaraktan arayüz ekranında "Şirinler Oyunu" tasarlamamız istenmiştir. Arayüz ekranında oluşturulan labirent üzerinde ; düşman karakterler (Azman ve Gargamel) , oyuncu karakteri (Tembel Şirin veya Gözlüklü Şirin) ve objeler (mantar ve altın) olacaktır. Kullanıcı , seçtiği oyunculardan birisini klavye yoluyla kontrol edip , puanlarını bitirmeden Şirine'ye ulaşmaya çalışacaktır. Düşman karakterler ise oyuncunun hareketine göre en kısa yolu kullanarak oyuncuyu yakalamaya çalışacaktır.

GİRİŞ

Bu projenin oluşturulmasındaki temel amaç ; nesneye yönelik programlama ve veri yapıları algoritmalarının birlikte kullanıldığı projeleri oluşturabilme yetkinliğine ulaşmamızdır. Projede , nesneye yönelik programlama için çok büyük önem arz eden encapsulation , inheritance , polymorphism , abstraction yapılarını ihtiyaç doğrultusunda mutlaka kullanmamız , her sınıf için biri parametreli biri parametresiz olmak koşuluyla iki constructor yazmamız , tüm özellikler için get ile set metodlarını tanımlamamız istenmiştir. Bu bağlamda ; Java ve C++ programlama dillerinden birisini seçip , görselleştirme için de seçtiğimiz programlama diline uygun kütüphaneyi kullanaraktan "Şirinler" adında bir oyun tasarlamamız istenmiştir. Programlama dili olarak Java'yı , arayüz tasarımı için ise Swing kütüphanesini tercih ettik.

Oyundaki amaç ; kullanıcının seçtiği oyuncunun , labirent içerisinde ilerleyerek puanını bitirmeden Şirine'ye ulaşmasıdır.

Oyunda ; Gargamel ve Azman olmak üzere iki düşman karakter ; Tembel Şirin ve Gözlüklü Şirin olmak üzere iki oyuncu karakter yer almaktadır.

Düşman karakterlerden Gargamel tek bir harekette 2 birim ilerlerken Azman 1 birim ilerleyebilmektedir. Ayrıca Gargamel , oyuncu karaktere dokunduğunda oyuncu karakterin 15 puanı azalırken Azman dokunduğunda 5 puanı azalır. Oyuncu karakterlerden Gözlüklü Şirin tek bir harekette 2 birim ilerlerken , Tembel Şirin 1 birim ilerleyebilmektedir.

Oyun başlamadan önce karakter seçimleri yapılır. Düşman karakterlerden biri veya ikisi seçilmeli , oyuncu karakterlerinden ise biri seçilmelidir. Düşman karakterlerin seçimi ve hangi kapıdan giriş yapacakları "harita.txt" dosyasında belirtilirken oyuncu karakterinin seçimi arayüz ekranında yapılır. Karakter seçimleri tamamlandıktan sonra seçilen düşman karakterler belirtilen kapılardan , oyuncu karakteri ise (6,5) koordinatından oyuna başlar. Oyuncu kontrolü kullanıcıda olacak ve klavye (oklar yardımı ile) üzerinden yapılacaktır. Oyun başlar başlamaz düşman karakterlerin oyuncu karakterine olan en kısa uzaklığı "dijksta algoritması" kullanılarak hesaplanır ve arayüz ekranında çizdirilir. Oyuncu karakterinin her hareketinde bu yol tekrardan hesaplanarak güncellenir. Düşman karakter , oyuncu karaktere her dokunduğunda oyuncu karakterin puanı azaltılır ve düşman karakter başlangıçta bulunduğu kapıdan tekrar başlar. Oyun süresince, oyuncunun puan kazanmasını sağlayacak altın ve mantarlarlar oluşacaktır. Random olarak maksimum 10 saniyede bir olacak şekilde 5 adet altın oluşur ve ekranda 5 saniye boyunca gözükür. Bir diğer objemiz olan mantar ise random olarak maksimum 20 saniyede bir olacak şekilde bir adet oluşup ekranda 7 saniye boyunca gözükür. Objelerin ekranda gözüktüğü süre zarfında oyuncu karakteri objelere dokunursa puanında artış meydana gelir. Bu artış ; altın için 5 puan iken mantar için 50 puandır. Oyuncu , puanlarını bitirmeden Şirine'ye ulaştığı takdirde oyunu kazanır. Eğer ulaşamadan puanları biterse , oyun sonlanacaktır.

YÖNTEM

Arayüz ekranı "OyunEkrani" classında dizayn edileceği için "OyunEkrani" classından "JFrame" extends edilir. "Oyun" classı ise oyunun takip edileceği class olup "OyunEkrani" classı sayesinde arayüz ekranına yansıtılır. "Oyun" classında ilk olarak oyuncu seçimi yapılmak üzere "secim" değişkeni oluşturulur. Kullanıcıdan alınan "secim" değeri 1 ise oyuncu olarak "Tembel Şirin" , 2 ise "Gözlüklü Şirin" seçilir.

Daha sonra "Metin" classında yer alan "dusman_dondur()" metoduyla "Oyun" classındaki çift boyutlu "dusmanlar" string dizisine düşmana ait bilgileri "harita.txt" dosyasından almış oluruz. "dusmanlar" dizinin birinci boyutu hangi düşman karakterin seçileceğini , ikinci boyutu ise düşman karakterin hangi kapıdan giriş yapacağını tutar. Karakter seçimleri tamamlandıktan sonra arayüz ekranında labirent oluşturulma aşamasına geçilir. Her bir karesi 60x60 pikselden oluşacak labirentte ; duvarlar siyaha , yollar beyaza boyanır. Daha sonra seçilen düşman karakterler için ; "Dusman" classında oluşturulmuş "yol_ciz()" metodu çağrılır. Bu metod içerisinde ilk olarak "dijksta()" metodu çağrılarak en kısa yol hesaplatılır. Sonrasında "yol_dondur()" metodu çağrılıp "takipYolu" arrayListine , düşman karakterle oyuncu arasındaki en kısa yolun bilgileri aktarılır. "Oyun" classında "yol_ciz()" metodu çalıştırıldıktan sonra , x ve y değişkenlerine "takipYolu" arrayListinden çizilecek yolun değerleri gönderilir. x değerleri için mod 13 işlemi uygulanırken y değerleri için ise 13'e bölme işlemi uygulanır. Çünkü "takipYolu" arrayListinde yolları tutan değerler çift boyutlu değil tek boyutlu olarak belirtilmiştir. Mod 13 işlemiyle kaçıncı sütunda olduğunu - x koordinatını - , 13'e bölme işlemi ile de kaçıncı satırda olduğunu - y koordinatını - buluruz. Bulunan bu koordinat değerlerindeki kareler boyanarak oyuncu ile düşman arasındaki en kısa yol ekranda gösterilmiş olur. Oyuncunun her ilerleyişinde bu işlemler tekrar uygulanarak en kısa yol gösterilmiş olur.

Oyuncu ilerleyişinin kontrolünün yapılacağı klavye yön tuşlarının her biri özelleştirilmiştir. Oyuncunun yön tuşuna bastığında , başlangıçta hangi oyuncu karakteri olduğu sorgulatılır. Oyuncu karakterlerinin hızları farklı olduğu için oyuncuya göre konum değişiklikleri farklı olacaktır. Oyuncu hareketi sağlandıktan sonra , oyuncunun oyunu kazanıp kazanmadığını sorgulayan "kazandi_mi()" metodu çağrılır , oyuncu bitiş noktasına gelmiş ise oyun sonlanacaktır. Gelmemiş ise düşman hareketini gerçekleştirmek üzere "dusman_hareket()" metodu çağrılır.

"dusman_hareket()" metodu çağrıldığında , ilk olarak oyunun bitip bitmediğini sorgulatır. Eğer oyun bitmediyse , metodun hangi düşman karakter için çağrıldığını sorgulatır. Sonrasında "yol_ciz()" metodu çağrılarak "takipYolu" güncellenir. Oyuncu hareketinde , düşmana dokunmuş mu diye ilk olarak düşmanın konumuyla oyuncunun konumu karşılaştırılır. Eğer oyuncu , düşmana dokunmuşsa , "dusman_degdi()" metodu çağrılarak oyuncunun puanı düşürülür(Gargamel değmişse 15 , Azman değmişse 5 puan düşürülür). Oyuncu hareketinde sonra düşmana değmemişse , bu sefer düşmanın hareketi gerçekleştirilir. Düşman hareketinden sonra tekrardan düşman karakter ile oyuncunun koordinatları karşılaştırılır. Düşman , oyuncuya değmişse "dusman_degdi()" metodu çağrılarak oyuncunun puanı azaltılır.

Oyunda , oyuncu karakterin puan kazanmasını sağlayan "Altın" ve "Mantar" sınıfında türeyen objeler de yer alacaktır. Objelerin oluşması ve ekranda kalması da belirli bir süre içinde yapılır. Altın için random olarak maksimum 10 saniyede bir altın oluşumu gerçekleştirmemiz istenmiştir. "altinSure" adında bir değişken oluşturup , her seferinde bu değişkene 1 ile 10 arasında random bir değer atanır. Bu değer , altının kaç saniye sonra oluşacağını belirtir. Altın oluştuktan sonra ise 5 saniye ekranda kalması , sonra kaybolması istenmiştir. Tüm bu süre işlemlerini "System.currentTimeMillis()" metodunu kullanarak ayarlarız. "altinBaslangic" adındaki değişkene , sürenin başlangıcında "System.currentTimeMillis()" değeri atanırken "altinBitis" adındaki değişkene de her 5 milisaniye de bir "System.currentTimeMillis()" değeri atanır. "altinSayac" değişkeni ise ("altinBitis"-"altinBaslangic")/1000 değerini tutar. Her 5 milisaniyede bir "altinBitis" değeri güncellendiğinden "altinSayac" değeri de güncellenir. Yani "altinSayac" değişkeni , "altinBaslangic" değişkenine "System.currentTimeMillis()" değeri atandığı andan itibaren geçen süreyi tutar. "altinSure" değişkeni (altının kaç saniye sonra ekranda görüleceğini tutan değişken) , "altinSayac" değişkenine (kaç saniye geçtiğini tutan değişken) eşit olduğu zaman altın oluşturma işlemlerinin olduğu if bloğuna girer ve 5 saniye boyunca bu blok çalışır. Blok içerisinde 5 tane altının oluşturulması için while döngüsü oluşturulmuştur. Altınlar oluşana kadar while döngüsü çalışır. Bu döngü içerisinde random olarak x ve y değerleri belirlenir. Daha sonra , o koordinatta ; yol , altın , mantar veya oyuncu olup olmadığı sorgulanır. Sorgulama işleminden sonra altın

oluşturulup koordinatları atandıktan sonra "altinlar" arrayListine eklenir. Oyuncu , altının üstüne geldiği zaman , "altinIptal" metodu çağrılıp "altinlar" arrayListindeki hangi altının üzerine geldiği bulunarak indis değeri "iptal" değişkenine atanır. Silinecek altın , "altinlar" arrayListinin "iptal" indeksindedir. Silme işlemi yapıldıktan sonra oyuncunun puanı arttırılır. Altında yapılan bu işlemler , aynı şekilde mantarlar için de yapılmıştır.

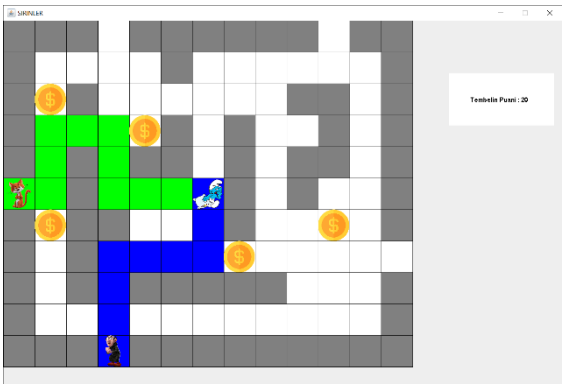
KARMAŞIKLIK ANALİZİ

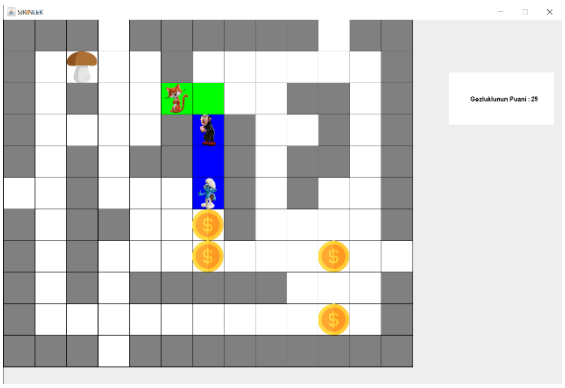
Dijkstra metodu içindeki for düğüm sayısı kadar dönmektedir.Burada labirentin her kutucuğu birdüğüm sayılmıştır.

Düğüm sayısına n dersek for n defa dönmektedir. For içinde min_uzaklık_bul()metodu çağrılmakta ve bir adet for döngüsü barındırmaktadır bu forlarda düğüm sayısı kadar dönmektedir.

Sonuç olarak bir forun içinde iki adet for dönmektedir içteki forlar n defadan toplamda 2n yaparlar dıştakı forun n defası ilede çarparsak toplamda $2n^2$ defa dönmüş olur bu sebeple $O(n)=n^2$ olur.

SONUÇLAR





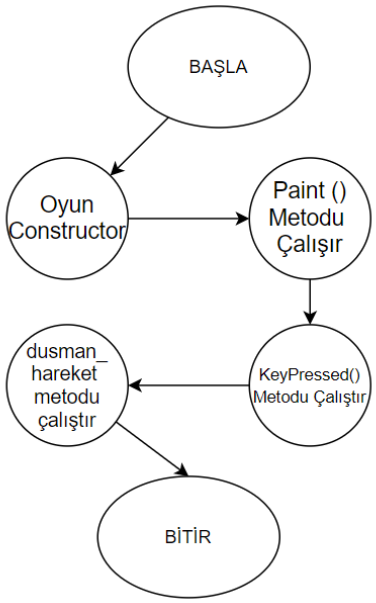
DENEYSEL SONUÇLAR

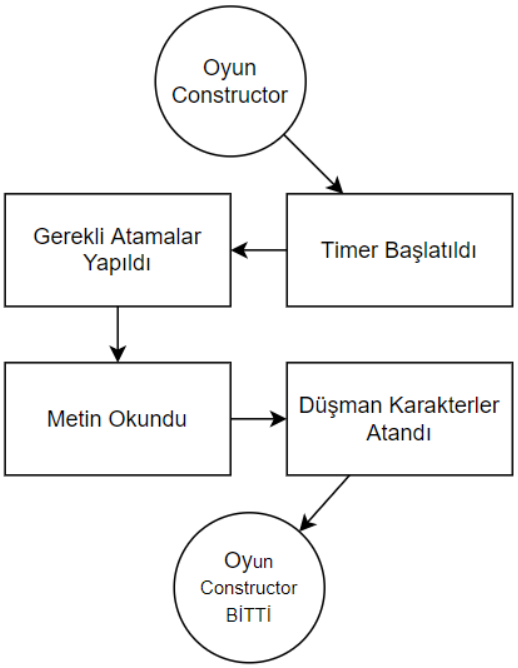
Projede tanımlanan probleme uygun çözümü üretip , bizden istenen tüm isterleri eksiksiz olarak yerine getirerek projeyi tamamladık.

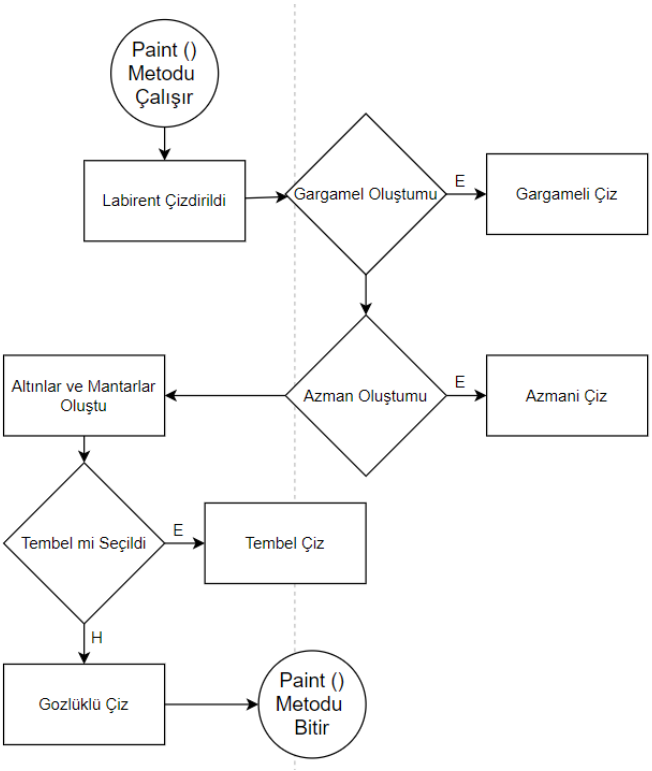
KAYNAKÇA

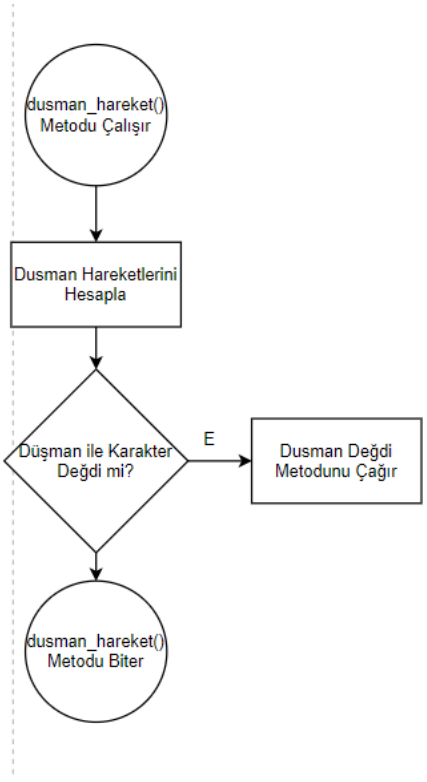
https://www.devglan.com/datastructure/dijkstra-algorithm-java

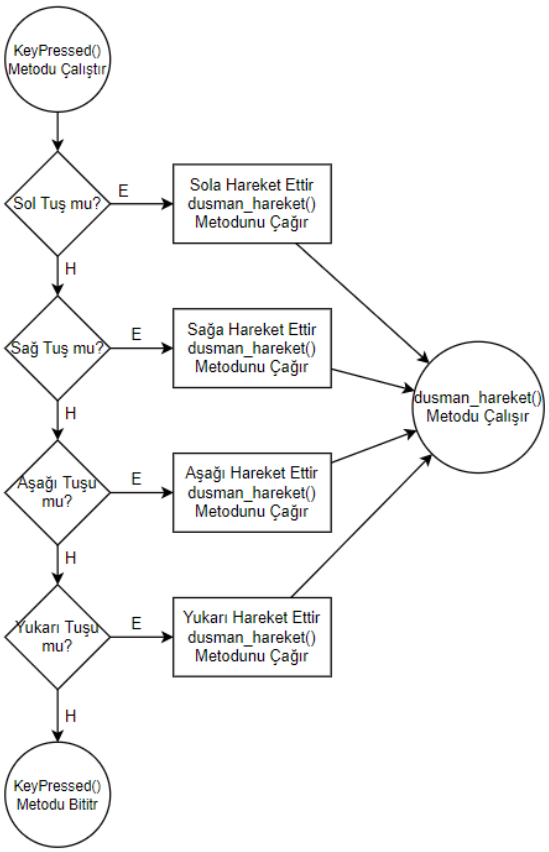
https://www.udemy.com/course/sifirdan-ileri-seviyeye-komple-java-gelistirici-kursu/

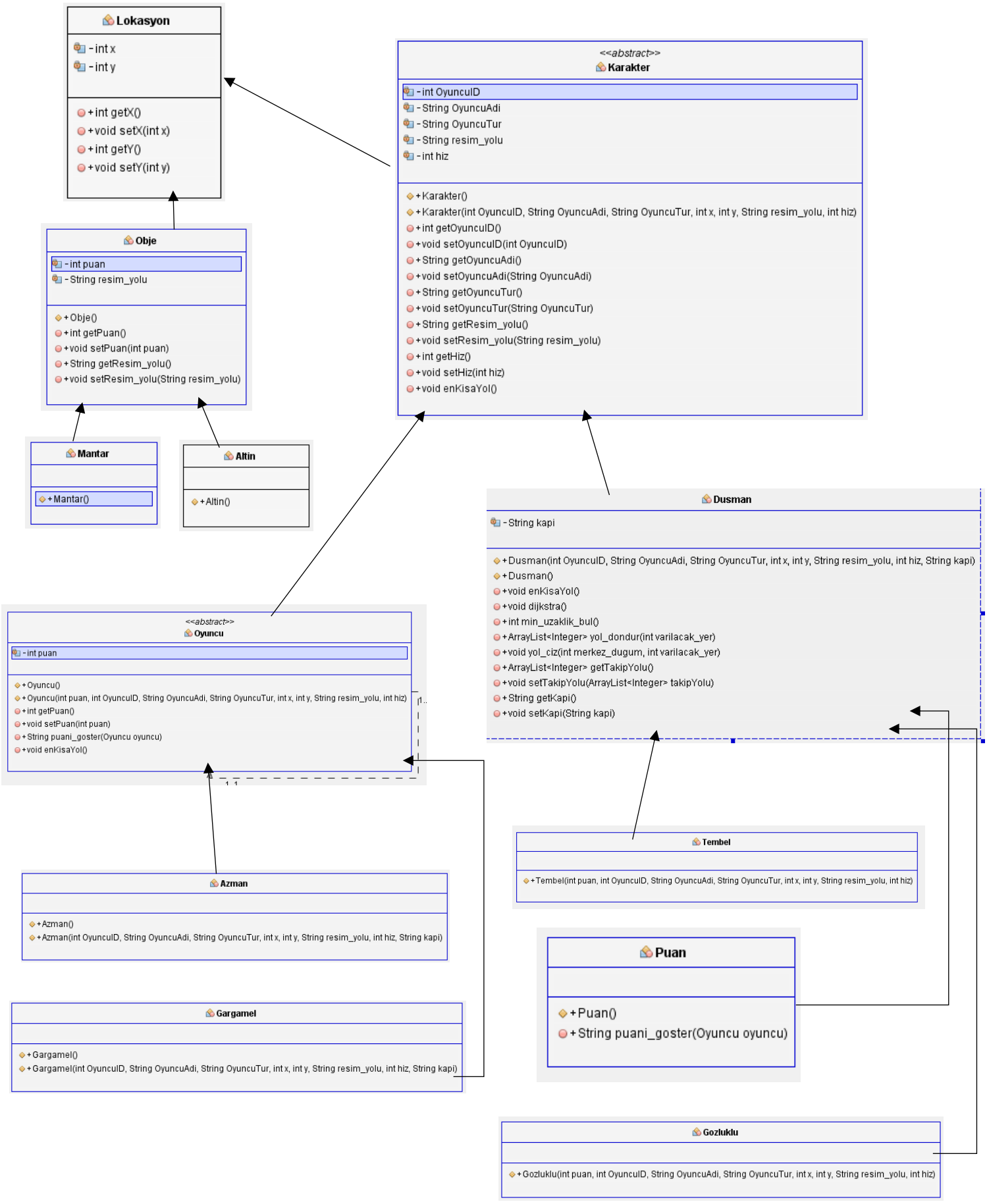

BAŞLA
Oyun Constructor
Paint () Metodu Çalışır
dusman_ hareket metodu çalıştır
KeyPressed() Metodu Çalıştır
BİTİR

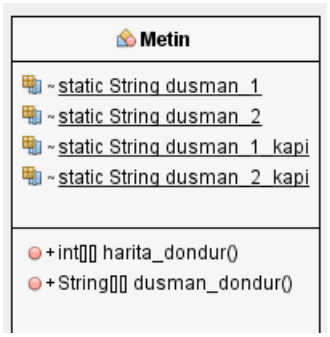

Oyun Constructor
Gerekli Atamalar Yapıldı
Timer Başlatıldı
Metin Okundu
Düşman Karakterler Atandı
Oyun Constructor BİTTİ

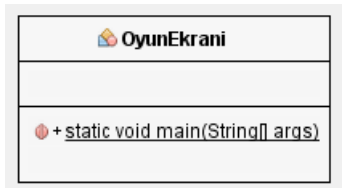

Paint () Metodu Çalışır
Labirent Çizdirildi
Gargamel Oluştumu
E
Gargameli Çiz
Azman Oluştumu
E
Azmani Çiz
Altınlar ve Mantarlar Oluştu
Tembel mi Seçildi
E
Tembel Çiz
H
Gozlüklü Çiz
Paint () Metodu Bitir

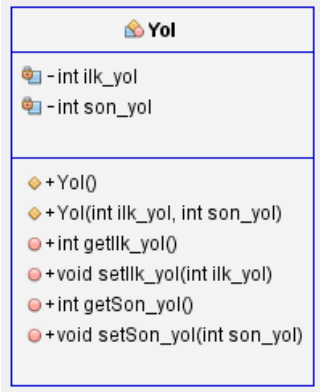

dusman_hareket() Metodu Çalışır
Dusman Hareketlerini Hesapla
Düşman ile Karakter Değdi mi?
E
Dusman Değdi Metodunu Çağır
dusman_hareket() Metodu Biter

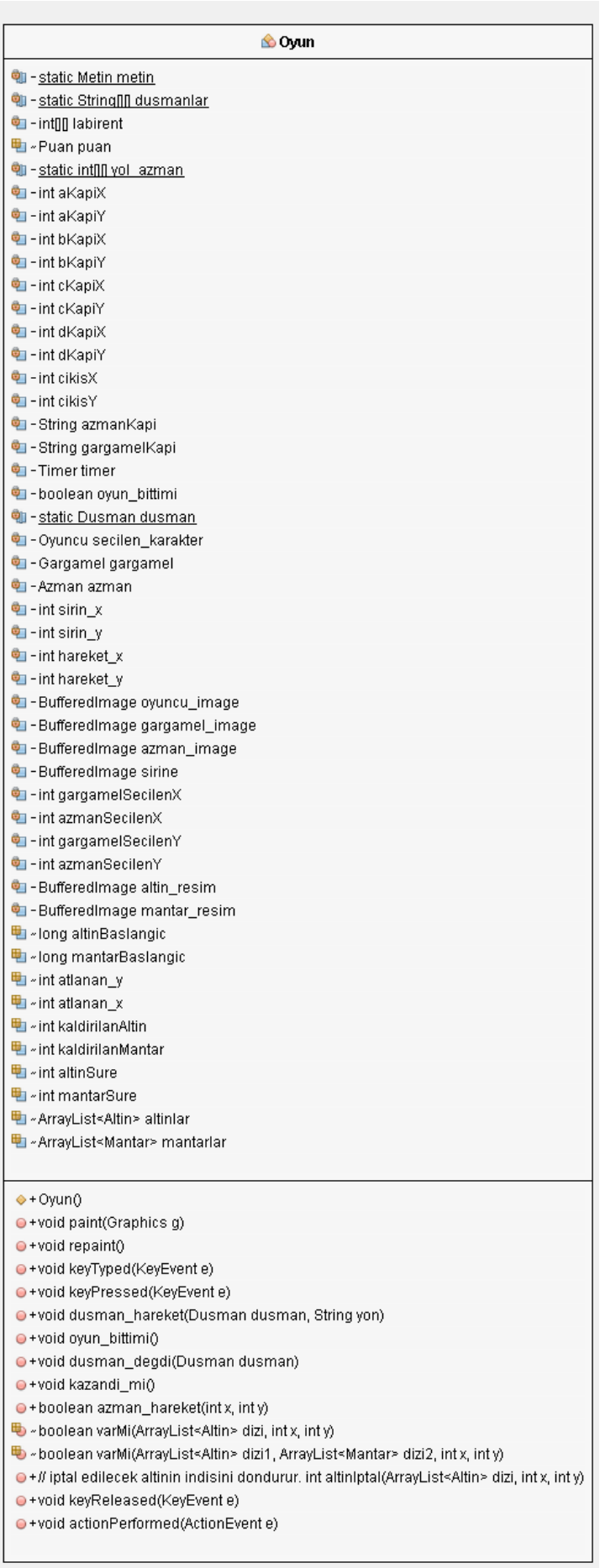

KeyPressed()
Metodu Çalıştır
Sol Tuş mu?
E
Sola Hareket Ettir
dusman_hareket()
Metodunu Çağır
H
Sağ Tuş mu?
E
Sağa Hareket Ettir
dusman_hareket()
Metodunu Çağır
H
Aşağı Tuşu
mu?
E
Aşağı Hareket Ettir
dusman_hareket()
Metodunu Çağır
H
Yukarı Tuşu
mu?
E
Yukarı Hareket Ettir
dusman_hareket()
Metodunu Çağır
H
KeyPressed()
Metodu Bititr
dusman_hareket()
Metodu Çalışır


Lokasyon
- int x
- int y
+ int getX()
+ void setX(int x)
+ int getY()
+ void setY(int y)
<<abstract>>
Karakter
- int OyuncuID
- String OyuncuAdi
- String OyuncuTur
- String resim_yolu
- int hiz
+ Karakter()
+ Karakter(int OyuncuID, String OyuncuAdi, String OyuncuTur, int x, int y, String resim_yolu, int hiz)
+ int getOyuncuID()
+ void setOyuncuID(int OyuncuID)
+ String getOyuncuAdi()
+ void setOyuncuAdi(String OyuncuAdi)
+ String getOyuncuTur()
+ void setOyuncuTur(String OyuncuTur)
+ String getResim_yolu()
+ void setResim_yolu(String resim_yolu)
+ int getHiz()
+ void setHiz(int hiz)
+ void enKisaYol()
Obje
- int puan
- String resim_yolu
+ Obje()
+ int getPuan()
+ void setPuan(int puan)
+ String getResim_yolu()
+ void setResim_yolu(String resim_yolu)
Mantar
+ Mantar()
Altin
+ Altin()
Dusman
- String kapi
+ Dusman(int OyuncuID, String OyuncuAdi, String OyuncuTur, int x, int y, String resim_yolu, int hiz, String kapi)
+ Dusman()
+ void enKisaYol()
+ void dijkstra()
+ int min_uzaklik_bul()
+ ArrayList<Integer> yol_dondur(int varilacak_yer)
+ void yol_ciz(int merkez_dugum, int varilacak_yer)
+ ArrayList<Integer> getTakipYolu()
+ void setTakipYolu(ArrayList<Integer> takipYolu)
+ String getKapi()
+ void setKapi(String kapi)
<<abstract>>
Oyuncu
- int puan
+ Oyuncu()
+ Oyuncu(int puan, int OyuncuID, String OyuncuAdi, String OyuncuTur, int x, int y, String resim_yolu, int hiz)
+ int getPuan()
+ void setPuan(int puan)
+ String puani_goster(Oyuncu oyuncu)
+ void enKisaYol()
Tembel
+ Tembel(int puan, int OyuncuID, String OyuncuAdi, String OyuncuTur, int x, int y, String resim_yolu, int hiz)
Azman
+ Azman()
+ Azman(int OyuncuID, String OyuncuAdi, String OyuncuTur, int x, int y, String resim_yolu, int hiz, String kapi)
Puan
+ Puan()
+ String puani_goster(Oyuncu oyuncu)
Gargamel
+ Gargamel()
+ Gargamel(int OyuncuID, String OyuncuAdi, String OyuncuTur, int x, int y, String resim_yolu, int hiz, String kapi)
Gozluklu
+ Gozluklu(int puan, int OyuncuID, String OyuncuAdi, String OyuncuTur, int x, int y, String resim_yolu, int hiz)


Metin
~ static String dusman_1
~ static String dusman_2
~ static String dusman_1_kapi
~ static String dusman_2_kapi
+ int[][] harita_dondur()
+ String[][] dusman_dondur()


OyunEkrani
+ static void main(String[] args)


Yol
- int ilk_yol
- int son_yol
+ Yol()
+ Yol(int ilk_yol, int son_yol)
+ int getIlk_yol()
+ void setIlk_yol(int ilk_yol)
+ int getSon_yol()
+ void setSon_yol(int son_yol)


Oyun
- static Metin metin
- static String[][] dusmanlar
- int[][] labirent
~ Puan puan
- static int[][] yol_azman
- int aKapiX
- int aKapiY
- int bKapiX
- int bKapiY
- int cKapiX
- int cKapiY
- int dKapiX
- int dKapiY
- int cikisX
- int cikisY
- String azmanKapi
- String gargamelKapi
- Timer timer
- boolean oyun_bittimi
- static Dusman dusman
- Oyuncu secilen_karakter
- Gargamel gargamel
- Azman azman
- int sirin_x
- int sirin_y
- int hareket_x
- int hareket_y
- BufferedImage oyuncu_image
- BufferedImage gargamel_image
- BufferedImage azman_image
- BufferedImage sirine
- int gargamelSecilenX
- int azmanSecilenX
- int gargamelSecilenY
- int azmanSecilenY
- BufferedImage altin_resim
- BufferedImage mantar_resim
~ long altinBaslangic
~ long mantarBaslangic
~ int atlanan_y
~ int atlanan_x
~ int kaldirilanAltin
~ int kaldirilanMantar
~ int altinSure
~ int mantarSure
~ ArrayList<Altin> altinlar
~ ArrayList<Mantar> mantarlar
+ Oyun()
+ void paint(Graphics g)
+ void repaint()
+ void keyTyped(KeyEvent e)
+ void keyPressed(KeyEvent e)
+ void dusman_hareket(Dusman dusman, String yon)
+ void oyun_bittimi()
+ void dusman_degdi(Dusman dusman)
+ void kazandi_mi()
+ boolean azman_hareket(int x, int y)
~ boolean varMi(ArrayList<Altin> dizi, int x, int y)
~ boolean varMi(ArrayList<Altin> dizi1, ArrayList<Mantar> dizi2, int x, int y)
+ // iptal edilecek altinin indisini dondurur. int altinIptal(ArrayList<Altin> dizi, int x, int y)
+ void keyReleased(KeyEvent e)
+ void actionPerformed(ActionEvent e)