ŞİRİNLER İLE LABİRENT OYUNU PROJESİ

Aslı Özen,*Beyza Hatip*
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
asliiozen@hotmail.com, beyzahatip17@gmail.com

Özet

Bu projede txt dosyasından karakter ve harita bilgisi alınarak harita oluşturuldu ve düşman karakterler kapı koordinatlarında oluşturuldu. Swing kullanılarak yapılan oyunun başlangıç penceresinde oyuncu seçimi soruldu. Oyuncu seçimine göre oyuna başlandı. Karakter 5,6 noktasında oluşturuldu. Her harekette düşman dijkstra algoritmasını kullanarak bir adım oyuncuya yaklaştı. Rastgele sürede mantar ve 5 altın oluşumu sağlandı. Puanın 0'ın altına düşmesi durumunda kaybetme ekranı tasarlandı ve oluşturuldu. Şirineye ulaşılması durumunda kazanma ekranı tasarlandı ve oluşturuldu.

Giriş

Çalışmada amaç; bizlerin nesneye yönelik programlama yapısını ve veri yapıları algoritmalarını anlanması ve çözüm sağlayabilmesidir. Çalışmada bizden Character sınıfı, Object, Location sınıfı oluşturulması istenmiştir. Character sınıfının alt sınıfı olarak Enemy ve Player sınıfları oluşturulmuştur. Enemy sınıfının alt sınıfı olarak Gargamel ve Azman sınıfları oluşturulmuştur. Player sınıfının alt sınıfı olarak BrainySmurf, LazySmurf ve Score sınıfları oluşturulmuştur. Object sınıfının alt sınıfı olarak Mushroom ve Gold sınıfı oluşturuldu. Character sınıfında bizden istenen özellikler tanımlandı, consturctor ve getter setter metodları yazıldı. Object sınıfında constructor ve abstract rand() metodu yazıldı. Location sınıfında X ve Y özelliği tanımlandı, constructor ve getter setter metodları yazıldı. Enemy sınıfında istenen özellikler tanımlandı, constructor ve getter setter metodları yazıldı. shortestRoad() metodu yazıldı. Player sınıfında bizden istenen özellikler tanımlandı. Constructor ve getter setter metodları yazıldı. Azman ve Gargamel sınıfında gerekli özellikler tanımlandı shortestRoad() metoduyla hareketleri sağlandı. boldijkstrapath() ile en kısa yol x ve y dizilerine atandı. Getter ve setterlar tanımlandı. BrainySmurf ve LazySmurf sınıfında gerekli özellikler getter setterlar tanımlandı. move() metodu ile hareket sağlandı. Score sınıfında ekrana skorun yazdırılması sağlandı. Mushroom ve Gold sınıfında gerekli özellikler tanımlandı. rand() metodu ile oluşacakları random x ve y değerleri hesaplandı. Board sınıfı oluşturuldu ve bu sınıfta oyunun akışı sağlandı. Map sınıfı oluşturuldu ve bu sınıfta harita ve karakter, kapı bilgileri okundu. Maze sınıfı oluşturuldu ve burada oyun çalıştırıldı.

Yöntem

Projede öncelikle projenin çalıştırılacağı Maze sınıfı oluşturuldu. Burada ara yüzde kullanacağımız java.swing ve java. awt kütüphaneleri import edildi. Ara yüz için JFrame tipinde frame adlı bir değişken tanımlandı. Main fonksiyonunda bu sınıfın constructorı çağrıldı. Bu constructurda setTitle() fonksiyonu ile oyun başladığında açılacak olan pencerenin adı koyuldu. setBackground() fonksiyonu ile bu pencerenin arka planını ayarlandı. add() fonksiyonu ile framemimize JPanel görevi görmesi için yeni bir Board nesnesi eklendi. setLocationRelativeTo() fonksiyonu ile açılan pencerenin ekranın ortasında oluşması sağlandı. setVisible() fonksiyonu ile penceremizi görünür kılındı. setDefaultCloseOperation() fonksiyonu ile pencerenin kapatma butonuna basılınca pencerenin kapanması sağlandı.

Map sınıfında harita.txt dosyasının okunması sağlandı. Öncelikle ara yüz için gerekli olan javax.swing, java.awt ve dosya okuması için gerekli olan java.io.File, java.util.Scanner kütüphaneleri import edildi. Sınıf içinde iki Scanner, Map adlı boyutu on bir olan bir String dizisi, karakter okuması için boyutu iki olan bir Character dizisi, haritanın

tüm yolları için road adında ve azman ve gargamel için dijsktra yollarının çizimi için djG djA adında Image tipinde, Şirinenin çizimi için Image tipinde sirine adlı, başlangıç ekranında karakter seçimi koyulacak olan 2 buton resmi için Image tipinde but1 but2 adlı, haritanın duvarları için wall adlı Image tipinde, kapılar için entryA entryB entryC entryD adında ve Image tipli, başlangıç ekranı için startBackground ve nameTag adında ve Image tipinde, kazanma ekranı için winScreen adında ve Image tipinde, kaybetme ekranı için lostScreen adında ve Image tipinde son olarak String dizisi tipinde ENEMY ve DOOR adında fieldlar oluşturuldu. Constructor içinde ImageIcon tipinde img adlı bir değişken oluşturuldu ve Image tipindeki fieldlarımıza fotoğraf atamamız burada yapıldı. Dosya okumada kullanılacak openFile(),readFile(),closeFile() metodları burada çağırıldı. Fieldlarımız için getter ve setterlar yazıldı. Openfield() fonksiyonunda try catch kullanarak dosyanın açılamaması vb. gibi durumlarda yazılacak olan hata mesajı belirlendi ve s,s1 scannerlarımızla dosya açıldı. Dosyanın satır sayısını bulabilmek için kullanacağımız int tipinde count değişkeni tanımlandı ve 0'a eşitlendi. While döngüsü s1.hasNext() koşulu ile kontrol edilerek döngünün içinde s1.next() fonksiyonu ile dosya okundu ve count her seferinde bir artırıldı bu şekilde count değerim satır sayıma eşit oldu. Yukarıda tanımdalıdığım Character, ENEMY ve DOOR fieldları initialize ederek boyutları count - 11 e eşitlendi. Bu şekilde kaç tane karakter tanımlandığı bulundu. Bir for döngüsü oluşturuldu ve bu döngü count-11 kadar döndürüldü. Bu for döngüsünün içinde karakter olan satırları s.next() ile okundu ve Character dizisine eklendi. Karakterlerin yazıldığı stringden karakter ve kapı adlarını çekebilmek için split() metodu kullanılarak bölünen string splitted adlı diziye atandı. Her karakter stringinin 4'e bölüneceği bilindiğinden splitted dizisinin 1 numaralı elemanı ENEMY dizisine, 3 numaralı elemanı DOOR dizisine eklendi. Haritanın okunması için s.hasNext() koşullu bir while döngüsü açıldı. Döngü içinde 0 'dan 11'e kadar dönen bir for döngüsü açıldı. Bu for döngüsünde her satır Map dizisinin bir indexi olarak atandı.

Character sınıfımızda int tipinde ID adında, String tipinde name adında ve characterKind adında, locC adında Location nesnesi şeklinde fieldlar oluşturuldu. Constructor ve shortestRoad isimli metodları, getter ve setterlar yazıldı. Character sınıfını extend eden Player sınıfında öncelikle java.awt kütüphanesi import edildi. playerID ve score adında int tipinde, playerName ve PlayerKind adında ve string tipinde fieldlar yazıldı. Constructor ve getter setter metotları yazıldı. Character sınıfını extend eden Enemy sınıfında String tipinde enemyKind ve enemyName adında, int tipinde enemyID adında field oluşturudu. Constructor, shortestRoad ve getter setter metotları yazıldı.

Player sınıfını extend eden LazySmurf sınıfında öncelikle javax.swing ve java.awt sınıfları import edildi. Location nesnesi olarak lsLoc adında, Image nesnesi olarak LazySmurf adında ve int tipinde score adında fieldlar oluşturuldu ve score'a 20 değeri atandı. Constructorda lsLoc initialize edildi, ImageIcon tipinde img adlı bir değişken tanımladı ve getImage() metodu kullanılarak LazySmurf nesnesine oyunda kullanacağımız tembel şirin fotoğrafı atandı. Karakterin başlangıç noktası olan 6,5 noktasını lsLoc.setX() ve lsLoc.setY() metodları kullanılarak atandı. Void tipinde move() adlı metodda int değerinde bir temp oluşturuldu ve bu temp değişkenine gözlüklü şirinin sahip olduğu x ve y değerlerine metoda gelen parametrelerin eklenmiş hali atandı. lsLoc.setX() ve lsLoc.setY() fonksiyonları kullanılarak gözlüklü şirinin x ve y 'si güncellendi. Getter ve setter metotları yazıldı.

Player sınıfını extend eden BrainySmurf sınıfında öncelikle javax.swing ve java.awt sınıfları import edildi. Location nesnesi olarak bsLoc adında, Image nesnesi olarak BrainySmurf adında ve int tipinde score adında fieldlar oluşturuldu ve score'a 20 değeri atandı. Constructorda bsLoc initialize edildi, ImageIcon tipinde img adlı bir değişken tanımladı ve getImage() metodu kullanılarak BrainySmurf nesnesine oyunda kullanacağımız gözlüklü şirin fotoğrafı atandı. Karakterin başlangıç noktası olan 6,5 noktasını bsLoc.setX() ve bsLoc.setY() metotları kullanılarak atandı. Void tipinde move() adlı metotda int değerinde bir temp oluşturuldu ve bu temp değişkenine gözlüklü şirinin sahip olduğu x ve y değerlerine metoda gelen parametrelerin eklenmiş hali atandı. bsLoc.setX() ve bsLoc.setY() fonksiyonları kullanılarak gözlüklü şirinin x ve y 'si güncellendi. Getter ve setter metotları yazıldı.

Enemy sınıfını extend eden Azman sınıfında önce javax.swing ve java.awt kütüphaneleri import edildi. azLoc adında Location nesnesi, int tipinde doorX, doorY adında, int dizisi tipinde intDivideX ve intDivideY adında, Image nesnesi Azman adında fieldlar oluşturuldu. Constructorda azLoc nesnesi initialize edildi. ImageIcon tipinde img adlı bir değişken tanımladı ve getImage() metodu kullanılarak

Azman nesnesine oyunda kullanacağımız azman fotoğrafı atandı. azLoc.setX() ve azLoc.setY() ile azmanın x ve y başlangıç değerleri atandı. shortestRoad() metodunda önce int tipinde tempX,tempY ve tmp değişkenleri tanımlandı. tempX değişkenine getIntDivideX()[1]-getIntDivideX()[0] değeri atandı. tempY değişkenine getIntDivideY()[1]-getIntDivideY()[0] değişkeni atandı. tmp değişkenine azLoc.getX() ile azmanın X değeri atandı. tmp değişkenine tempX değişkeni atandı ve azLoc.setX() ile azmanın X değerine tmp atandı. Aynı işlem azmanın Y değeri için de tekrarlandı. Getter ve setterlar yazıldı. setDoorX(int doorX) ve setDoorY(int doorY) metotlarında haritadan okunan kapı x ve y değerleri azmanın x ve ye değerleri olarak atanır.

Enemy sınıfını extend eden Gargamel sınıfında önce javax.swing ve java.awt kütüphaneleri import edildi. grLoc adında Location nesnesi, int tipinde doorX, doorY adında, int dizisi tipinde intDivideX ve intDivideY adında, Image nesnesi Gargamel adında fieldlar oluşturuldu. Constructorda grLoc nesnesi initialize edildi. ImageIcon tipinde img adlı bir değişken tanımladı ve getImage() metodu kullanılarak Gargamel nesnesine oyunda kullanacağımız azman fotoğrafı atandı. grLoc.setX() ve grLoc.setY() ile azmanın x ve y başlangıç değerleri atandı. shortestRoad() metodunda if koşulunda karakterin X'i eksi düşmanın X'i 1 veya -1'e eşit ise ve Y'leri eşitse veya karakterin Y'si eksi düşmanın Y'si 1 veya -1'e eşitse ve X'leri eşitse tempX değişkenini intDivideX dizisinin 0 elemanı eksi 1. elemanına eşitlendi ve tempY değişkenini intDivideY dizisinin 0 elemanı eksi 1. elemanına eşitlendi. If koşulu yanlış ise tempX değişkenini intDivideX dizisinin 0 elemanı eksi 2. elemanına eşitle ve tempY değişkenini intDivideY dizisinin 0 elemanı eksi 2. elemanına eşitle.Gargamel'in sahip olduğuğu X ve Y değerlerine tempX ve tempY değerlerini ekle. Getter ve setterlar yazıldı. setDoorX(int doorX) ve setDoorY(int doorY) metotlarında haritadan okunan kapı x ve y değerleri azmanın x ve ye değerleri olarak atandı.

Abstract olan Object sınıfında constructor ve abstract rand() metodu tanımlandı. Object sınıfını extend eden Gold sınıfında öncelikle javax.swing, java.awt, java.util.ArrayList ve java.util.Random kütüphaneleri import edildi. glLoc adında Location nesnesi, int tipinde value adında 15 değerine eşit, Gold nesnesi tutan goldL adında ArrayList, Map nesnesi olan m adında, Image nesnesi olan Gold adında fieldlar oluşturuldu. Constructor içinde glLoc ve m initialize edildi ve ImageIcon tipinde img adlı bir değişken tanımladı ve getImage() metodu kullanılarak Gold nesnesine oyunda kullanacağımız altın para fotoğrafı atandı. Getter ve setter metotları yazıldı. rand( ) metotunda öncelikle int tipinde bir temp değişkeni tanımlandı ve 0 değeri atandı. Koşulu (temp==0) olan while döngüsü içinde iki adet Random nesnesi oluşturuldu. Adları x1 ve y1 olan iki adet int değişken oluşturuldu. x1 değişkenine rnd.nextInt(13) değerini atayarak x1 değişkenine 0 dahil ve 13 dahil olmamak üzere 0 ile 13 arasında rastgele bir değer atandı. y1 değişkenine rndy.nextInt(11) değerini atayarak y1 değişkenine 0 dahil ve 11 dahil olmamak üzere 0 ile 11 arasında rastgele bir değer atandı. Koşulu (m.getMap(x1, y1).equals("1")) olan ve x1 ve y1'e atanan değerlerin haritadaki yerlerinin duvarların üzerine gelip gelmediğini kontrol eden if koşulunun içinde glLocsetX() ve glLoc.setY() setterları ile x1 ve y1 değerleri altının x ve y değerlerine atanır ve temp bir artırılır. Eğer if koşulu sağlanmıyorsa temp sıfıra eşitlenir.

Object sınıfını extend eden Mushroom sınıfında öncelikle javax.swing, java.awt ve java.util.Random kütüphaneleri import edildi. mshrmLoc adında Location nesnesi, int tipinde value adında 50 değerine eşit, Map nesnesi olan m adında, Image nesnesi olan Mushroom adında fieldlar oluşturuldu. Constructor içinde mshrmLoc ve m initialize edildi ve ImageIcon tipinde img adlı bir değişken tanımladı ve getImage() metodu kullanılarak Mushroom nesnesine oyunda kullanacağımız mantar fotoğrafı atandı. Getter ve setter metotları yazıldı. rand( ) metotunda öncelikle int tipinde bir temp değişkeni tanımlandı ve 0 değeri atandı. Koşulu (temp==0) olan while döngüsü içinde iki adet Random nesnesi oluşturuldu. Adları x1 ve y1 olan iki adet int değişken oluşturuldu. x1 değişkenine rnd.nextInt(13) değerini atayarak x1 değişkenine 0 dahil ve 13 dahil olmamak üzere 0 ile 13 arasında rastgele bir değer atandı. y1 değişkenine rndy.nextInt(11) değerini atayarak y1 değişkenine 0 dahil ve 11 dahil olmamak üzere 0 ile 11 arasında rastgele bir değer atandı. Koşulu (m.getMap(x1, y1).equals("1")) olan ve x1 ve y1'e atanan değerlerin haritadaki yerlerinin duvarların üzerine gelip gelmediğini kontrol eden if koşulunun içinde mshrmLoc.setX() ve mshrmLoc.setY() setterları ile x1 ve y1 değerleri altının x ve y değerlerine atanır ve temp bir artırılır. Eğer if koşulu sağlanmıyorsa temp sıfıra eşitlenir.

Location sınıfında int tipinde X ve Y fieldları tanımlandı. Constructor içinde X ve Y fieldlarına 0

değeri atandı. X ve Y fieldları için getter ve setter metotları yazıldı.

Player sınıfını extend eden Score sınıfında önce java.awt kütüphanesi import edildi. String tipinde Score fieldı tanımladı. Constructor içinde bu fielda "SCORE: 20" değeri atandı. Parametre olarak boolean kind değerini, Graphics g , BrainySmurf ve LazySmurf nesnesi alan showScore() metotunda öncelikle bir if açılır ve kind koşul olarak seçilir. kind doğru ise karakterimin BrainySmurf olduğu anlaşılır ve if içindeki işlemler ona göre yapılır. İf içinde Score değişkeni "SCORE: "+ BrainySmurf nesnesinin skoruna eşitlenir. g.setColor() ile ekrana yazdırılacak Score değişkeninin rengi ayarlandı. g.setFont() ile değişkenin fontu ayarlandı.g.drawString() ile değişkenim ekrana yazdırıldı. Aynı işlem else koşulunda LazySmurf nesnesi için yapıldı. Getter ve setter metotları tanımlandı.

Dijkstra için ShowGraph, Graph, Node ve Edge sınıfları oluşturduk. Node sınıfında Edge nesnelerinin listesini tuttuk böylece belirli bir düğümden tüm kenarlara ulaşabildik.Ayrıca kendi ağırlıklı graphımızı yaratabilmek, dinamik bir yapı elde edebilmek için; her Edge nesnesi kaynak(Node source) ve hedef (Node destination) Node nesnesini içeriyor. Edge sınıfı ile başlayalım. Node nesneleri, ağırlıklı grafiğimizdeki gerçek düğümleri temsil eder. Bu sınıfı kenarlardan kısa bir süre sonra uygulayacağız. Node source ile Node destination Node nesneleri ile double tipinde weight oluşturduk ve `Edge(Node s, Node d, double w)` ile atamalarını gerçekleştiriyoruz.Daha sonra nesneleri yazdırmak için toString () ve CompareTo () metodlarını uyguluyoruz. NodeWeigted nesnesi oluşturduğumuz için o sınıfın name özelliğini kullanarak toString() metodunda `return String.format("(%s -> %s, %f)", source.name, destination.name, weight); ile kaynak ve hedef düğümün adlarını ve ağırlığı string formatında return ediyoruz.`

Node sınıfına geçelim. Haritadaki nodeların adı için String tipinde name özelliği; node'un ziyaret edilip edilmediğini kontrol edebilmemiz için boolean tipinde bir visited özelliği ve `LinkedList<Edge> edges;` ile Edge nesnelerinin listesini tutuk. Böylece belirli bir düğümden tüm kenarlara ulaşabildik. Constructorda name'e atama yaptık, visited'ı false ettik ve yukarıda tanımladığımız edges'i `edges = new LinkedList<>();` ile initalize ettik. `boolean isVisited(),` ile visited ne ise onu return ediyoruz.

`void visit(), void unvisit()` metodları ile node ziyaret edildiyse true; edilmediyse false' a eşitledik. Bu fonksiyon ve özellikler diğer sınıflarda çağrıldığı zaman daha açık şekilde anlatılacaktır. Şimdi, bir grafiği temsil etmek için önceki her iki sınıfı da kullanacak olan Graph sınıfını anlatalım. Düğümlerimizi grafikte saklamak için Set kullandık(`private Set<Node> nodes;`). Boolean directed özelliği tanımladık. Constructorda gönderilen directed değerini directed özelliğimize atadık ve `nodes = new HashSet<>();` ile nodes'u initalize ettik.`public void addEdge(Node source, Node destination, double weight)` metodu ShowGraph sınıfında ona göndereceğimiz iki düğüm ve aralarındaki mesafe ile iki düğümün bağlanmasını sağlar.`nodes.add(source); nodes.add(destination);` ile daha önce olmayan düğümlerse nodes' a ekleriz.Set yinelenen nesnelere izin vermediği için onu seçmiştik. Yinelenen kenarlarımız olmadığından emin olmak için addEdgeHelper() kullanıyoruz.addEdgeHelper(source, destination, weight) ile metodumuza bağlayacağımız iki düğümü ve aralarındaki mesafeyi göndermiş oluyoruz.Metodda `for (Edge edge : a.edges)` ile bütün kenarları geziyoruz.`if (edge.source == a && edge.destination == b)` koşulu ile bu kenarın daha önce bu iki düğüm arasında olup olmadığına bakıyoruz.Eğer varsa `edge.weight = weight` ile kenarın ağırlığın güncelliyoruz.Eğer bu kenarı bulamadıysak `a.edges.add(new Edge(a, b, weight))` ile bu kenarı ekliyoruz.Edge sınıfı kaynak(Node a) ve hedef (Node b) Node nesnesini içerdiği için ve Node clasında Edge nesnesinin listesini tuttuğumuz için böyle bir ekleme işlemi yapabildik. Tüm nodelar ve kenarlar oluşturulduktan sonra dijikstra algoritmasını uygulamaya geçiyoruz. `public String DijkstraShortestPath(Node start, Node end)` metodu daha sonra ShowGraph sınıfında kendisine parametre olarak gönderilen 2 node arasındaki en kısa yolu bize String tipinde döndürecek. Öncelikle `HashMap<Node, Node> changedAt = new HashMap<>();` ile

HashMap sınıfından bir nesne oluşturuyoruz.Burada 2'si de Node tipinde anahtar ve onun değerini put ile ekliyoruz.`(changedAt.put(start, null);)`

Bunları her düğüm için en kısa yolun takibini yapmak için yazıyoruz.`HashMap<Node, Double> shortestPathMap = new HashMap<>();`ile yine HashMap sınıfından bir nesne oluşturuyoruz.Aşağıda bunu kullanıcaz.

`for (Node node : nodes)` döngüsü ile bütün nodeları geziyoruz. `{ if (node == start`)if koşulu ile başlangıç düğümünde olup olmadığımız kontrol ettikten sonra eğer öyleysek `shortestPathMap.put(start, 0.0) ile` başlangıç nodeunun ağırlığını 0 olarak giriyoruz.Başlangıç node'u dışındaki tüm nodelara positif sonsuz ataması yapıyoruz.Böylece dijikstra algoritmasının ilk adımını uygulamış olduk.`for (Edge edge : start.edges)` döngüsü ile başlangıç düğümünden gidebileceğimiz tüm düğümlerden geçiyoruz.`{ shortestPathMap.put(edge.destination, edge.weight);` ile grapha hedef düğümü ve ağırlığını eklerken `changedAt.put(edge.destination, start); }` ile hedef düğümü ve başlangıç düğümünü ekliyoruz.Başlangıç düğümü gezildiği için `start.visit();` ile boolean visit değerini true ediyoruz.`while (true)` döngüsü ziyaret edebileceğimiz ziyaret edilmemiş bir düğüm olduğu sürece çalışır. `Node currentNode = closestReachableUnvisited(shortestPathMap);` ile anlık düğümü closestReachableUnvisited metodunun bize return ettiği düğüm ile bulmuş oluruz.Eğer bu fonksiyon null return ettiyse`(if (currentNode == null) /` bu düğümler arasında yol yok demektir.Bunu konsola yazdırıyoruz.Yani, henüz bitiş düğümüne ulaşmadıysak ve erişilebilir başka bir düğüm yoksa, başlangıç ve bitiş arasındaki yol mevcut değildir (bağlı değiller).En yakın ziyaret edilmemiş düğüm hedefimizse, yolu yazdırmak istiyoruz. Ve `if (currentNode == end)` koşulu altında bunu yazdırıyoruz.ve child dediğimiz son düğüm en son gelen anlık düğüme eşit oluyor.`(Node child = end; )` Daha sonra bize en kısa yolu string tipinde gösterecek path'i end'e eşitliyoruz.While döngüsü açıyoruz.changedAt,child -> ebeveyn ilişkilerini izler.Yolu yazdırmak için parent'ı child'dan ve onun soyundan önce eklememiz gerekir.Artık parent child key'inin objesine eşit olur. (`Node parent = changedAt.get(child);)` Eğer bu işlemin sonunda parent=null bir değere eşit olursa döngüden çıkılır.Daha sonra `path = parent.name + ", " + path; child = parent;` işlemleri yaptırılır.Yolu yazdırmak için parent'ı child'dan ve onun soyundan önce eklememiz gerekir.path'i konsola yazdırıp kontrol ediyoruz `shortestPathMap.get(end));` ile de maaliyeti kontrol ediyoruz.Yani iki düğüm arasındaki en kısa yolu ve en az maaliyeti kontrol ediyoruz.`currentNode.visit();` ile bu node'u de gezdiğimiz için visited özelliğini true ediyoruz.Şimdi, mevcut düğümümüzün bir kenarı olan tüm ziyaret edilmemiş düğümlerden geçiyoruz ve mevcut düğümümüzden geçerken en kısa yol değerinin daha önce sahip olduğumuzdan daha iyi olup olmadığını kontrol ediyoruz.`for (Edge edge : currentNode.edges)` döngüsü bütün nodeları gezmemizi; `if (edge.destination.isVisited()) continue;` ise ziyaret edilmemiş nodelara geçmemizi sağlıyor. Daha sonra yapılan işlemi daha rahat anlayabilmek için şöyle düşünebilriz.Ulaşmak istediğimiz bir m node'u olsun. S node'undan n node'una ordan da m node'una gidebiliyor olalım. Eğer önce S node'undan n node'una ordan da m node'una olan yol S node'undan direkt m node'una giden yoldan kısaysa bu daha iyi bir yoldur ve güncel yol olarak burayı seçeriz. `if (shortestPathMap.get(currentNode) + edge.weight < shortestPathMap.get(edge.destination))` ile bu durumu kontrol ediyor;altındaki işlemlerle güncellememizi yapıyoruz.Son olarak, ulaşabileceğimiz ve daha önce ziyaret etmediğimiz en yakın düğümün hangisi olduğunu değerlendiren en yakınReachableUnvisited () metoduna geçelim.Bu metod `(HashMap<Node, Double> shortestPathMap)` map'ini parametre olarak alır.Yani düğüm ve mesafe bilgisini.Bize gidebileceğimiz düğümlerden en kısa mesafede olanı return eder.Öncelikle en kısa yol pozitif sonsuza eşitlenir.(`shortestDistance = Double.POSITIVE_INFINITY;)` ve seçebileceğimiz en yakın node'a null atanır.`(Node closestReachableNode = null;). for (Node node : nodes) { if (node.isVisited()) continue;` ile gezilmiş nodeları geçip gezilmemiş

bir node arıyoruz.`double currentDistance = shortestPathMap.get(node);` ile anlık nodumuza bize gönderilen node'un double değerindeki ağırlığına eşitliyoruz.

```
if (currentDistance == Double.POSITIVE_INFINITY) continue;
```

Ile bize gönderilen ağırlık değeri +sonsuz' dan farklı olana kadar arıyoruz.Çünkü zaten en kısa mesafeyi bu metodun başında +sonsuza eşitlemiştik.

```
if (currentDistance < shortestDistance) {
shortestDistance = currentDistance;
closestReachableNode = node;
```

ile bize gönderilen değer shortesDistance'tan küçük olduğu zaman(zaten herhangi bir değer +sonsuzdann küçük olacağından) en kısa mesafeyi artık bize gönderilen ağırlığa eşitliyoruz ve en yakın ulaşılabilir node'u bu node yapıp onu return ediyoruz.Son sınıfımız olan ShowGraph sınıfına geçiyoruz.Bu sınıf çağırıldığında dijikstra algoritması kullanılarak bulunmuş en kısa yolu String tipinde return eder.Bunun için düşmanın ve oyunucumuzun x ve y koordinatlarını parametre olarak alır.DijkstraShortestPath() metodu bize yolu göndereceğinden Graph sınıfından bir nesne oluşturuyoruz.Daha sonra string tipinde, tablo adında iki boyutlu bir dizi oluşturuyoruz.Bunu haritamızı bir matris gibi düşünerek dolduruyoruz.Duvar olan yerlere "d" geri kalan yerlere o satır ve sütunda bulunan karenin konumlarını giriyoruz.Bunu yapmamızın sebebi birazdan nodeları oluştururken nodeların isimlerine buradan atama yapmamız.Daha sonra bu nodelardan oluşan yolu kullanacağımızdan harita ile uyumlu bir iki boyutlu dizi oluşturduk.Düğümlerin adları için Node nesnelerinin 2 boyutlu dizisini tuttuk..harita 11*13 lük olduğundan boyutları o şekilde belirledik.

```
dugumler[i][j] = new Node(tablo[i][j]);
```

ile düğümlerin adlarını tablodaki sıralarıyla atamış olduk.Böylece haritamızdaki her düğümün adı konumu olmuş oldu.Daha sonra node'ları bağlamak ve aralarındaki mesafeleri girmek için yine 11*13'lük 2 adet for döngüsü açtık.Her node bir sonraki x veya y konumundaki node ile bağlanacağından bir sonraki konumun bizim satır veya sütunumuzn dışında olmaması koşulunu koyduk`.if (i + 1 <= 10 && j + 1 <= 12)` .Nodelar duvar olan yerlerle bağlanmasın diye seçilen node'un veya bağlanacak "d" ye eşit olmaması koşullarını da koyduk.Daha sonra Graph sınıfı addEdge metoduna iki adet node'u ve aralarındaki mesafeyi göndererek nodeları bağladık.Bizim haritamızda her kare bir düğüm olduğundan ve kareler ile komşu kareleri arasında 1 br uzaklık olduğundan "1" değerini;ç ift yönlü olması açısından nodeların yerlerini değiştirerek de gönderdik.Daha sonra String tipinde bir yol değişkeni tanımladık ve DijkstraShortestPath() metodu kullanılarak bize gönderilen en kısa yolu bu değişkene atadık.Bu metod ShowGraph sınıfındaki bu enemyRoad() metoduna gönderilen düşman ve oyuncu konumlarını alarak bu iki node arasındaki en kısa yolu bize döndürür.Biz de bu yolu tanımladığımız yol değişkenine eşitleyip bunu return ediyoruz.

Şimdi Gargamel ve Azman sınıfının dijikstra uygulamalarına bakalım. Önce boldijikstrapath() metotlarını açıklayalım. Bu metod şirinlerin sınıflarından objeleri, hangi şirin olduğunu belli eden boolean bir değer ve ShowGraph sınıfı enemyRoad() fonskiyonuna göndermek için düşman ve şirinlerin konum bilgilerini alıyor.Bir ShowGraph nesnesi oluşturuyoruz.Boolean tipindeki brainy kullanılarak; hangi şirinle oynuyorsak, onun konum bilgileri;hangi düşman sınıfındaysak, onun konum bilgileri ile enemyRoad() fonksiyonuna gönderiliyor.Bu fonksiyon en kısa yolu bize String olarak gönderdiğinden String tipinde bir değişken tanımlıyoruz.(pathstring).Bu değişkene return edilen yolu atadıktan sonra konumlar (x,y) formatında olduğundan ve her konum arasında da (,) bulunduğundan split() hazır fonksiyonu kullanarak virgüllere göre bu diziyi bölüyor String tipinde başka bir diziye atıyoruz.(divide1)Biz X ve Y konumlarını ayrı dizilerde tutmak istediğimizden ve bu değerleri int olarak kullanmamız gerektiğinden,2 adet int tipinde x ve y dizileri oluşturuyoruz.Virgüllere göre bölünmüş String dizimizi ikiye böleceğimizden bu dizilerin boyutlarını String dizimizin boyutunun yarısı olarak ayarlıyoruz.Bir for döngüsü açıp bu dizileri doldurmak istiyoruz.for döngüsü bu dizilerin boyutu kadar döneceğinden ikisnden birini seçtik.String tipindeki virgüllere göre ayrılmış

dizide çift sayılar x'lere tek sayılar y'lere denk geldiğinden if koşulunu buna göre yazıyor ve o diziye atama yapıyoruz. Fakat atama yaparken String tipinden int' e dönüşüm yapmamız gerektiği için önce Integer.valueOf() hazır fonksiyonu ile int değere dönüşüp öyle dizilere atıyoruz. Böylece iki node arası en kısa yol için geçtiğimiz konumlar ayrı x ve y dizilerine int tipinde atanmış oldu. Bu işlem Gargamel ve Azman sınıflarında aynıdır.

JPanel sınıfını extend eden ve ActionListener sınıfını implements eden Board sınıfında öncelikle Ara yüzde kullanılacak javax.swing, java.awt, java.awt.event.ActionEvent,java.awt.event.ActionListener, java.awt.event.KeyAdapter, java.awt.event.KeyEvent, java.util.Random kütüphaneleri import edildi. m adında Map, lySmurf adında LazySmurf, brSmurf adında BrainySmurf, gl adında Gold, mrm adında Mushroom, gargamel adında Gargamel, azman adında Azman, rand adında Random, scr adında Score nesneleri, boolean tipinde win, lost, isDrawObjectG, isDrawObjectM, startScreen, brainy değişkenleri, int tipinde whichEnemy, time, duration, random, randomMantar adında özellikler oluşturuldu. win, lost, isDrawObjectG, isDrawObjectM özelliklerinin başlangıç değeri false, brainy, startScreen özelliklerinin başlangıç değeri true atandı. whichEnemy ve time özelliğine 0, duration özelliğine 200 başlangıç değeri atandı. randomMantar ve random özelliklerine 0 ile 7 arasında rastgele bir tam sayı başlangıç değeri atandı. Constructor tanımlandı. Constructor içerisinde lySmurf, bsSmurf, mrm, gl, azman, gargamel, scr, m özellikleri initialize edildi. addKeyListener() metodu ile karakterin hareketinin sağlanmasında kullanacağımız bir Key Listener eklendi. setFocusable(true) metodu ile bir tuşa bastığımızda panelimizin bunu fark etmesi sağlandı. setBackground() metodu ile panelimizin arka plan rengi belirlendi. initGame() metodu çağırıldı. actionPerformed() metodunda iki if koşulu yazıldı. Birinci if koşulunda şirinlerin bulundukları konum kontrol edilerek şirine ile aynı konumdalar ise win özelliğinin değeri true yapıldı. İkinci koşulda şirinlerin puanı 0 olursa veya 0'ın altına düşerse lost özelliğinin değeri true yapıldı. time değişkeni 1 artırıldı ve repaint() metodu çağırıldı. initGame() metodunda öncelikle readEnemyAndDoor() metodu çağırılarak düşman karakterlerin ve kapıların bilgisi alındı. gl1, gl2, gl3, gl4 ve gl5 olmak üzere 5 adet Gold nesnesi oluşturuldu ve gl özelliğinin GoldL ArrayListine eklendi. For döngüsünde gl nesnesinin rand metodu çağırıldı ve random x ve y değerleri ArrayList'in elemanları olan Gold nesnelerinin x ve y değerlerine eşitlendi. mrm nesnesinin rand metodu çağırıldı. If else koşulu ile hangi karakter için işlem yapacağımız belirlendi. If else içerisinde switch case kullanılarak whichEnemy özelliğinin değerlerine göre 0 ise gargamel.boldijkstrapath, 1 ise azman.boldijkstrapath, 2 ise hem gargamel.boldijkstrapath hem azman.boldijkstrapath fonksiyonu çağırıldı. timer adlı Timer nesnesi oluşturuldu. delay değeri 25 ve listener değeri this olarak atandı. Timer'ı başlatmak için timer.start() metodu kullanıldı. Çizme işleminde kullanılacak paint(Graphics g) sınıfında öncelikle super.paint(g) metodu çağırıldı. If koşuluyla startScreen true ise startScreen(g) metodu çağırıldı. If koşuluyla win, startScreen ve lost değerleri false olduğu sürece drawMap(), drawShortestRoad(), drawCharacter(), drawObjects(), checkCollision() metodları çağırıldı. If koşuluyla win doğru ise winScreen() metodu, lost doğru ise lostScreen() metodu çağırıldı. drawObjects() metodunda If koşuluyla time değerinin random*40' değerinden büyük olup olmadığını kontrol edildi. Bu koşul içerisinde bir If koşulu daha açıldı ve bu koşulda time değişkeninin random*40 + duration değerinden küçük olması koşulu kontrol edilir. Amaç altın fotoğraflarının belirli bir süre ekranda kalmasını sağlamaktır. Koşul doğru ise isDrawObjectG true değerini alır. For döngüsü içinde 5 adet Gold resmi oluşturulması sağlandı. If koşuluyla time değerinin randomMantar*40' değerinden büyük olup olmadığını kontrol edildi. Bu koşul içerisinde bir If koşulu daha açıldı ve bu koşulda time değişkeninin randomMantar*40 + duration değerinden küçük olması koşulu kontrol edilir. Amaç mantar fotoğraflarının belirli bir süre ekranda kalmasını sağlamaktır. Koşul doğru ise isDrawObjectM true değerini alır. For döngüsü içinde Mushroom resmi oluşturulması sağlandı. drawCharacter() metodunda brainy özelliğinin değerine göre drawImage() metodu ile gözlüklü şirin ya da tembel şirinin fotoğrafı çizdirildi. checkCollision() metodunda brainy koşuluna ait if döngüsünde whichEnemy ile switch case yazıldı. whichEnemy 0 ise if koşuluyla gargamel ve oyuncunun aynı karede olup olmadığı kontrol edildi. Aynı karede iseler gargamel gargamel.setTileX() ve gargamel.setTiley() metodu kullanılarak başlangıç noktasına gönderildi. Oyuncunun puanı lySmurf.setScore(-15) metoduyla eksiltildi. whichEnemy 1 ise if koşuluyla azman ve oyuncunun aynı karede olup olmadığı kontrol edildi. Aynı karede iseler azman gargamel.setTileX() ve azman.setTiley()

metodu kullanılarak başlangıç noktasına gönderildi. Oyuncunun puanı lySmurf.setScore(-5) metoduyla eksiltildi. whichEnemy 2 ise yukarıdaki iki işlem gerçekleştirildi. Koşulu isDrawObjectG olan if koşulunda for döngüsü yazıldı. For döngüsü içerisinde oyuncunun ve altınların aynı karede olup olmadığı kontrol ettirildi. Aynı karede iseler lySmurf.setScore(gl.getValue()) metodu ile oyuncunun skoru artırıldı. Altınlar haritanın dışına konumlandırıldı. . Koşulu isDrawObjectM olan if içerisinde oyuncunun ve mantarın aynı karede olup olmadığı kontrol ettirildi. Aynı karede iseler lySmurf.setScore(mrm.getValue()) metodu ile oyuncunun skoru artırıldı. Mantar haritanın dışına konumlandırıldı. readEnemyAndDoor() metodunda m.getDOOR() ve m.getENEMY metodları çağırıldı. for döngüsünde m nesnesinden gelen DOOR dizisinin elemanlarının değeri switch case kullanılarak kontrol edildi. Her case içerisinde if koşulu ile ENEMY dizinin elemanlarının "Gargamel" veya "Azman"a eşit olup olmadığı kontrol edildi. Case "A" ise ve if koşulunda Gargamel'e eşitlik çıkmış ise whichEnemy 0 değerine eşitlenir ve gargamel nesnesinin DoorX ve DoorY değerleri 3 0 olmak üzere atanır. Case "A" ise ve if koşulunda Azman'a eşitlik çıkmış ise whichEnemy 1 değerine eşitlenir ve azman nesnesinin DoorX ve DoorY değerleri 3 0 olmak üzere atanır. . Case "B" ise ve if koşulunda Gargamel'e eşitlik çıkmış ise whichEnemy 0 değerine eşitlenir ve gargamel nesnesinin DoorX ve DoorY değerleri 10 0 olmak üzere atanır. Case "B" ise ve if koşulunda Azman'a eşitlik çıkmış ise whichEnemy 1 değerine eşitlenir ve azman nesnesinin DoorX ve DoorY değerleri 10 0 olmak üzere atanır. Case "C" ise ve if koşulunda Gargamel'e eşitlik çıkmış ise whichEnemy 0 değerine eşitlenir ve gargamel nesnesinin DoorX ve DoorY değerleri 0 5 olmak üzere atanır. Case "C" ise ve if koşulunda Azman'a eşitlik çıkmış ise whichEnemy 1 değerine eşitlenir ve azman nesnesinin DoorX ve DoorY değerleri 0 5 olmak üzere atanır. Case "D" ise ve if koşulunda Gargamel'e eşitlik çıkmış ise whichEnemy 0 değerine eşitlenir ve gargamel nesnesinin DoorX ve DoorY değerleri 3 10 olmak üzere atanır. Case "D" ise ve if koşulunda Azman'a eşitlik çıkmış ise whichEnemy 1 değerine eşitlenir ve azman nesnesinin DoorX ve DoorY değerleri 3 10 olmak üzere atanır. Son olarak bir if koşulu açılır ve ENEMY dizisinin uzunluğunun 2 olması durumunda whichEnemy özelliğine 2 değeri atanır. drawMap() fonksiyonunda iç içe iki for döngüsü açılır ilk döngü y değerlerim ve ikinci döngü x değerime denk gelir. İçteki döngünün içerisinde bir if koşulu yazılır m.getMap(x,y).equals("1") şartı ile haritadan okunan string satırının(y) istenilen konumunda(x) "1" değeri olup olmadığı kontrol edilir. Koşul sağlanıyor ise bir if daha açılır ve x=3 y=0 koşulu kontrol edilir doğru ise drawImage ile A kapısının fotoğrafı ekrana çizdirilir. Else if açılır ve x=10 y=0 koşulu kontrol edilir doğru ise drawImage ile B kapısının fotoğrafı ekrana çizdirilir. Else if açılır ve x=0 y=5 koşulu kontrol edilir doğru ise drawImage ile C kapısının fotoğrafı ekrana çizdirilir. Else if açılır ve x=3 y=10 koşulu kontrol edilir doğru ise drawImage ile D kapısının fotoğrafı ekrana çizdirilir. Bu koşullar doğru değil ise drawImage ile yol fotoğrafı çizdirilir. if koşulu yazılır m.getMap(x,y).equals("0") şartı ile haritadan okunan string satırının(y) istenilen konumunda(x) "0" değeri olup olmadığı kontrol edilir. Koşul sağlanıyor ise drawImage ile duvar fotoğrafı çizdirilir. DrawImage ile şirine çizdirilir. scr.showScore() ile skor ekrana yazdırılır. drawShortestRoad() metodunda if koşulu ile hangi düşmanın oyunda olduğuna göre for döngüsü açılır. Gargamel oyunda ise for döngüsü gargamelin yolunun x değerlerini tutan dizinin boyutu kadar döner ve drawImage ile gargameli ve dijkstra yolunu ekrana çizer. . Azman oyunda ise for döngüsü azmanın yolunun x değerlerini tutan dizinin boyutu kadar döner ve drawImage ile azmanı ve dijkstra yolunu ekrana çizer. . Gargamel ve Azman oyunda ise yukarıdaki işlemlerin ikisi de çalışır. lostScreen() metodunda drawImage ile m nesnesinin LostScreen fotoğrafı ekrana çizdirilir. winScreen() metodunda drawImage ile m nesnesinin WinScreen fotoğrafı ekrana çizdirilir ve drawString() metodu ile oyuncunun puanı ekrana yazdırılır. startSscreen() drawImage ile m nesnesinin StartBackground, but1 but2 ve nameTag fotoğrafı ekrana çizdirilir. KeyAdapter extend eden Al sınıfında keyPressed() metodu tanımlanır ve bu metodda keycode adı int bir değer tanımlanır. getKeyCode() metoduyla bastığımız tuş int değere dönüşür ve keycode değişkenine atanır. If koşulu yazılır ve bu koşulda startScreen değişkeni doğru olduğunda bir if else if tanımlanır. B tuşuna bastığımızda brainy değişkeninin true olmasını ve startScreen değişkeninin false olmasını sağlar veya L tuşuna bastığımızda brainy değişkeninin ve startScreen değişkeninin false olmasını sağlar. If koşulu açılarak işlemler karakterin türüne göre ayrılır. Basılan tuş yukarı ok tuşu ise oyuncunun bir üst karesi duvar değil ise gargamel ve azmanın shortestRoad() metodları çağırılır. lySmurf.move() metoduna sırasıyla 0 ve -1 değerleri gönderilir. Koşulun dışında checkCollision() metodu ve azman ve gargamel için boldijkstrapath() metodu çağırılır. Basılan tuş aşağı ok tuşu ise oyuncunun bir alt karesi

duvar değil ise gargamel ve azmanın shortestRoad() metodları çağırılır. lySmurf.move() metoduna sırasıyla 0 ve 1 değerleri gönderilir. Koşulun dışında checkCollision() metodu ve azman ve gargamel için boldijkstrapath() metodu çağırılır. Basılan tuş sol ok tuşu ise oyuncunun bir sol karesi duvar değil ise gargamel ve azmanın shortestRoad() metodları çağırılır. lySmurf.move() metoduna sırasıyla -1 ve 0 değerleri gönderilir. Koşulun dışında checkCollision() metodu ve azman ve gargamel için boldijkstrapath() metodu çağırılır. Basılan tuş sağ ok tuşu ise bulunan karenin 12,7 olup olmadığı kontrol edilir. 12,7 değil ise ve oyuncunun bir sağ karesi duvar değil ise gargamel ve azmanın shortestRoad() metodları çağırılır. Oyuncunun türüne göre move metoduna sırasıyla 1 ve 0 değerleri gönderilir. Koşulun dışında checkCollision() metodu ve azman ve gargamel için boldijkstrapath() metodu çağırılır. 12,7 ise gargamel ve azmanın shortestRoad() metodları çağırılır. lySmurf.move() metoduna sırasıyla 1 ve 0 değerleri gönderilir. Koşulun dışında checkCollision() metodu ve azman ve gargamel için boldijkstrapath() metodu çağırılır. Karakter gözlüklü şirin ise Basılan tuş yukarı ok tuşu ise oyuncunun bir üst karesi ve iki üst duvar değil ise gargamel ve azmanın shortestRoad() metodları çağırılır. brSmurf.move() metoduna sırasıyla 0 ve -2 değerleri gönderilir. Koşulun dışında checkCollision() metodu ve azman ve gargamel için boldijkstrapath() metodu çağırılır. Basılan tuş aşağı ok tuşu ise oyuncunun bir alt ve iki alt karesi duvar değil ise gargamel ve azmanın shortestRoad() metodları çağırılır. brSmurf.move() metoduna sırasıyla 0 ve 2 değerleri gönderilir. Koşulun dışında checkCollision() metodu ve azman ve gargamel için boldijkstrapath() metodu çağırılır. Basılan tuş sol ok tuşu ise oyuncunun bir sol karesi ve iki sol karesi duvar değil ise gargamel ve azmanın shortestRoad() metodları çağırılır. brSmurf.move() metoduna sırasıyla -2 ve 0 değerleri gönderilir. Koşulun dışında checkCollision() metodu ve azman ve gargamel için boldijkstrapath() metodu çağırılır. Basılan tuş sağ ok tuşu ise bulunan karenin 12,7 olup olmadığı kontrol edilir. 12,7 değil ise ve oyuncunun bir sağ karesi duvar değil ise gargamel ve azmanın shortestRoad() metodları çağırılır. Oyuncunun türüne göre move metoduna sırasıyla 2 ve 0 değerleri gönderilir. Koşulun dışında checkCollision() metodu ve azman ve gargamel için boldijkstrapath() metodu çağırılır. 12,7 ise gargamel ve azmanın shortestRoad() metodları çağırılır. brSmurf.move metoduna sırasıyla 2 ve 0 değerleri gönderilir. Koşulun dışında checkCollision() metodu ve azman ve gargamel için boldijkstrapath() metodu çağırılır.

Sonuçlar

Düşman karakterin üstünden geçse bile karakter puan kaybetmeliydi fakat dijkstrada karaktere olan en yakın yol çizdiğinden ve gargamel ona göre hareket ettiğinden oyununcunun ilerisinde bir yol çizilemeyeceğinden hata alınıyor. Bu nedenle düşman karakter üstünden geçeceği zaman 1 adım atsın şeklinde karar verildi. Harita.txt dosyasından harita okunurken boşluklar olmadan okuma yapılıyor.

Yalancı Kod:

Maze Sınıfı

- Ara yüzde kullanılacak kütüphaneleri import et
- frame adında JFrame nesnesi oluştur.
- Static main metodunda Maze adlı Constructorı çağır
- Consturctor içinde frame'in başlığını ayarla.
- Frame'in arka plan rengini ayarla.
- Frame'e Board nesnesi ekle.
- Frame'in yüksekliğini ve genişliğini ayarla.
- Frame'in ekranın ortada açılmasını ayarla.
- Frame'i görünür kıl.
- Frame'in boyutunu değiştirilemez olarak ayarla.
- Çıkış butonuna basıldığında Frame'in kapanmasını ayarla.

Map Sınıfı

- Ara yüzde kullanılacak kütüphaneleri import et.
- s1 ve s olmak üzere Scanner nesnesi oluştur.
- String dizisi tipinde boyutu 11 olan Map özelliği oluştur.
- String dizisi tipinde boyutu 2 olan Character adında özelliği oluştur.
- Image tipinde road, djG, djA, sirine, but1, but2, wall adında özelliği oluştur.
- Image tipinde entryA, entryB, entryC, entryD, startBackground adında özelliği oluştur.
- Image tipinde winScreen, lostScreen, nameTag adında özelliği oluştur.

- String dizisi tipinde ENEMY, DOOR adında özelliği oluştur.
- Constructorda oluşturulan Image tipindeki özelliklere ImageIcon kullanarak fotoğraf ata.
- openfile(), readFile(), closeFile() metodlarını çağır.
- Özellikler için getter ve setter tanımla.
- openFile() metotunda try içerisinde harita.txt dosyasını oku ve catch içinde hata mesajını yaz.
- readFile() içinde count değişkeni tanımla ve 0 değerini ata.
- Döngü içinde haritayı oku ve her satır okumasında count değişkenini bir artır.
- Character, ENEMY, DOOR dizilerinin boyutunu count-11 yap.
- For döngüsü içerisinde Character dizisine harita.txt dosyasının count -11'inci satırına kadar ata.
- Okunan string'i ":" ve "," karakterlerine göre böl ve splitted dizisine ekle.
- Splitted dizisinin 1. elemanını ENEMY dizisine, 3. elemanını DOOR dizisine ekle
- while döngüsünde harita.txt dosyasının count-11'inci satırından sonrasını Map dizisinde ata.

Character Sınıfı

- int tipinde ID adında özellik tanımla.
- String tipinde name ve characterKind adında özellik tanımla.
- Constructor yaz.
- shortestRoad metodunu yaz.
- Özelliklerin getter ve setterlarını yaz.

Enemy Sınıfı

- int tipinde enemyID adında özellik tanımla.
- String tipinde enemyName ve enemyKind adında özellik tanımla.
- Constructor yaz.
- shortestRoad metodunu yaz.
- Özelliklerin getter ve setterlarını yaz.

Azman Sınıfı

- Ara yüzde kullanılacak kütüphaneleri import et.
- azLoc adında Location nesnesi oluştur.
- doorX, doorY adında int tipinde özellik oluştur.
- Azman adında Image nesnesi oluştur.
- intDivideX ve intDivideY adında int dizisi oluştur.
- Constructorda azLoc nesnesini initialize et.
- Oluşturulan Image tipindeki özelliklere ImageIcon kullanarak fotoğraf ata.
- azLoc.setX ve azLoc.setY metotlarını kullanarak azman karakterine bir başlangıç değeri ata.
- shortestRoad() metotunda tempX ve tempY değişkeni tanımla.
- tempX değişkenini intDivideX dizisinin 0 elemanı eksi 1. elemanına eşitle.
- tempY değişkenini intDivideY dizisinin 0 elemanı eksi 1. elemanına eşitle.
- Azman'ın sahip olduğuğu X ve Y değerlerine tempX ve tempY değerlerini ekle.
- BÖLDISKTRAPATH
- Özellikler için getter ve setter tanımla.

Gargamel Sınıfı

- Ara yüzde kullanılacak kütüphaneleri import et.
- grLoc adında Location nesnesi oluştur.
- doorX, doorY adında int tipinde özellik oluştur.
- Gargamel adında Image nesnesi oluştur.
- intDivideX ve intDivideY adında int dizisi oluştur.
- Constructorda grLoc nesnesini initialize et.
- Oluşturulan Image tipindeki özelliklere ImageIcon kullanarak fotoğraf ata.
- grLoc.setX ve grLoc.setY metotlarını kullanarak gargamel karakterine bir başlangıç değeri ata.
- shortestRoad() metotunda tempX ve tempY değişkeni tanımla.
- If koşulunda karakterin X'i eksi düşmanın X'i 1 veya -1'e eşit ise ve Y'leri eşitse veya karakterin Y'si eksi düşmanın Y'si 1 veya -1'e eşitse ve X'leri eşitse tempX değişkenini intDivideX dizisinin 0 elemanı eksi 1. elemanına eşitle ve tempY değişkenini intDivideY dizisinin 0 elemanı eksi 1. elemanına eşitle.

- If koşulu yanlış ise tempX değişkenini intDivideX dizisinin 0 elemanı eksi 2. elemanına eşitle ve tempY değişkenini

intDivideY dizisinin 0 elemanı eksi 2. elemanına eşitle.
- Gargamel'in sahip olduğuğu X ve Y değerlerine tempX ve tempY değerlerini ekle.
- BÖLDISKTRAPATH
- Özellikler için getter ve setter tanımla.

Player Sınıfı

- int tipinde playerID adında özellik tanımla.
- String tipinde playerName ve playerKind adında özellik tanımla.
- Constructor yaz.
- showScore metodunu yaz.
- Özelliklerin getter ve setterlarını yaz.

BrainySmurf Sınıfı

- Ara yüzde kullanılacak kütüphaneleri import et.
- bsLoc adında Location nesnesi oluştur.
- BrainySmurf adında Image nesnesi oluştur.
- score adında int dizisi oluştur 20 değerine eşitle.
- Constructorda brLoc nesnesini initialize et.
- Oluşturulan Image tipindeki özelliğe ImageIcon kullanarak fotoğraf ata.
- brLoc.setX ve brLoc.setY metotlarını kullanarak gözlüklü şirin karakterine bir başlangıç değeri ata.
- move() metotunda temp değişkeni tanımla.
- temp değişkenini fonksiyona gelen tx artı BrainySmurf'un X değerine eşitle.
- BrainySmurf'un sahip olduğuğu X değerine tempX değerini ata.
- temp değişkenini fonksiyona gelen ty artı BrainySmurf'un Y değerine eşitle.
- Özellikler için getter ve setter tanımla.

LazySmurf Sınıfı

- Ara yüzde kullanılacak kütüphaneleri import et.
- lsLoc adında Location nesnesi oluştur.
- LazySmurf adında Image nesnesi oluştur.
- score adında int dizisi oluştur 20 değerine eşitle.
- Constructorda lsLoc nesnesini initialize et.
- Oluşturulan Image tipindeki özelliğe ImageIcon kullanarak fotoğraf ata.
- lsLoc.setX ve lsLoc.setY metotlarını kullanarak gözlüklü şirin karakterine bir başlangıç değeri ata.
- move() metotunda temp değişkeni tanımla.
- temp değişkenini fonksiyona gelen tx artı LazySmurf'un X değerine eşitle.
- LazySmurf'un sahip olduğuğu X değerine tempX değerini ata.
- temp değişkenini fonksiyona gelen ty artı LazySmurf'un Y değerine eşitle.
- Özellikler için getter ve setter tanımla.

Location Sınıfı

- int tipinde X ve Y adında özellikler tanımla.
- Constructor'da X ve Y özelliklerine 0 başlangıç değerini ata.
- Özelliklerin getter ve setterlarını yaz.

Object Sınıfı

- Constructor'da oluştur.
- Abstract rand() metodunu yaz.

Gold Sınıfı

- Ara yüzde kullanılacak kütüphaneleri import et.
- glLoc adında Location nesnesi oluştur.
- Gold adında Image nesnesi oluştur.
- value adında int özellik oluştur 15 değerine eşitle.
- M adlı Map nesnesi oluştur.
- Gold nesnesi tutan boyutu 5 olan goldL adında ArrayList oluştur.
- Constructorda glLoc ve m nesnesini initialize et.
- Oluşturulan Image tipindeki özelliğe ImageIcon kullanarak fotoğraf ata.
- Özellikler için getter ve setter tanımla.
- rand() metodunda int tipinde temp adında değişken oluştur 0 değerini ata.
- Koşulu temp==0 olan while döngüsü oluştur.
- Döngüde rnd adlı Random nesnesi değişken oluştur.
- int tipinde x1 adında değişkene 0 ile 13 arasında random bir tam sayı ata.
- rndy adlı Random nesnesi değişken oluştur.
- int tipinde y1 adında değişkene 0 ile 11 arasında random bir tam sayı ata.

- m.getMap() metoduyla haritada x1 y1 değerlerindeki yerlerin yola denk gelip gelmediğine bak.
- Geliyorsa x1 ve y1 değerlerini Gold nesnesinin x ve y değeri yap ve temp değerini 1 artır.
- Gelmiyorsa temp değerini 0'a eşitle.

Mushroom Sınıfı

- Ara yüzde kullanılacak kütüphaneleri import et.
- mshrmLoc adında Location nesnesi oluştur.
- Mushroom adında Image nesnesi oluştur.
- value adında int özellik oluştur 50 değerine eşitle.
- M adlı Map nesnesi oluştur.
- Constructorda mshrmLoc ve m nesnesini initialize et.
- Oluşturulan Image tipindeki özelliğe ImageIcon kullanarak fotoğraf ata.
- Özellikler için getter ve setter tanımla.
- rand() metodunda int tipinde temp adında değişken oluştur 0 değerini ata.
- Koşulu temp==0 olan while döngüsü oluştur.
- Döngüde rnd adlı Random nesnesi değişken oluştur.
- int tipinde x1 adında değişkene 0 ile 13 arasında random bir tam sayı ata.
- rndy adlı Random nesnesi değişken oluştur.
- int tipinde y1 adında değişkene 0 ile 11 arasında random bir tam sayı ata.
- m.getMap() metoduyla haritada x1 y1 değerlerindeki yerlerin yola denk gelip gelmediğine bak.
- Geliyorsa x1 ve y1 değerlerini Mushroom nesnesinin x ve y değeri yap ve temp değerini 1 artır.
- Gelmiyorsa temp değerini 0'a eşitle.

Score Sınıfı

- Ara yüzde kullanılacak kütüphaneleri import et.
- String tipinde Score adında özellik tanımla.
- Constructorda Score değişkenine "SCORE: 20" değerini ata.
- showScore() metodunda drawString kullanarak Score değişkenini ekrana yazdır.
- Özelliklerin getter ve setterlarını yaz.

Board Sınıfı

- Ara yüzde kullanılacak kütüphaneleri tanımla.
- Map, LazySmurf, BrainySmurf, Gold, Mushroom, Gargamel, Azman, Score, Random nesnesi tanımla.
- win, lost, startScreen, brainy, isDrawObjectG, isDrawObjectM boolean değişkenlerini tanımla.
- whichEnemy, time, duration, random, randomMantar int değişkenlerini tanımla
- Constructor içinde nesneleri initialize et.
- KeyListener ekle.
- actionPerformed() metodunda kazanma ve kaybetme durumlarını tanıt ve time değişkenine 1 ekle.
- repaint() metodunu çağır.
- initGame() metodunu çağır.
- initGame() metodunda readEnemyAndDoor() metodunu çağır.
- 5 adet Gold nesnesi oluştur ArrayList'e ekle.
- 5 adet Gold nesnesine random x ve y değerleri ata.
- Mantar nesnesine random x ve y değeri ata.
- Hangi düşman var ise o düşmanın boldijkstrapath() metodunu çağır.
- Timer nesnesi oluştur ve delay 25 listener this olsun.
- paint() metodunda super.paint(g) metodu yazılır.
- Başlama ekranında olup olmama kontrol edilir. Başlamada ise startSscreen() metodu çağırılır.
- Kazanma kaybetme ve başlama durumları yanlış ise harita karakterler, objeler, en kısa yol çizilir ve çarpışmalar kontrol edilir.
- Kazanma durumunda olup olunmadığı kontrol edilir doğru ise winScreen() çağırılır.
- Kaybetme durumunda olup olunmadığı kontrol edilir doğru ise lostScreen() çağırılır.
- drawObjects() metodunda süre koşuluyla mantar ve 5 altın nesnesi çizdirilir.
- drawCharacter() metodunda hangi karakter seçilmiş ise o karakter çizdirilir.
- checkCollision() metodunda düşmanlar ve karakter arasında ve objeler ve karakter arasında çarpışmayı kontrol et.
- Varsa puan azalt veya artır.

- readEnemyAndDoor() fonskyonunda Map sınıfında okunan kapı ve karakter bilgilerine göre kapı ve karakter ataması yap.
- drawMap() fonksiyonunda m.getMap() fonksiyonunu kullanarak “0” yazan yerlere duvar “1” yazan ve kapı olmayan yerlere yol çiz.
- Kapı olan yerlere kapı çiz.
- Şirineyi ve skoru çiz.
- drawShortestRoad() fonksiyonunda düşmana göre belirlenen dijkstra yolunu ekrana çiz.
- lostScreen() metodunda kaybetme fotoğrafını ekrana çiz.
- winScreen() metodunda kazanma fotoğrafını ve skoru ekrana çiz.
- startSscreen() metounda başlangıç ekranını ve seçim butonlarının fotoğrafını çiz.
- Al sınıfında startScreen değerini kontrol et.
- Doğruysa L veya B tuşuna basılınca startScreen değerini false yap karakter seçimini tamamla.
- Yukarı ok tuşuna bastıysa üst kısmı duvar mı kontrol et değilse ilerlet.
- Düşmanlar için kısa yol hesapla.
- Çarpışmayı kontrol et
- Aşağı ok tuşuna bastıysa alt kısmı duvar mı kontrol et değilse ilerlet.
- Düşmanlar için kısa yol hesapla.
- Çarpışmayı kontrol et
- Sol ok tuşuna bastıysa sol kısmı duvar mı kontrol et değilse ilerlet.
- Düşmanlar için kısa yol hesapla.
- Çarpışmayı kontrol et
- Sağ ok tuşuna bastıysa 12,7 noktasında olup olmadığını kontrol et ve sağ kısmı duvar mı kontrol et değilse ilerlet.
- Düşmanlar için kısa yol hesapla.
- Çarpışmayı kontrol et.

ShowGraph Sınıfı

- enemyRoad() metodunu yaz.
- graph adında Graph nesnesi oluştur.
- Tablo adlı iki boyutlu dizi oluştur ve bu diziye harita koordinatlarını string olarak ata duvar olan yerlere “d” yaz.
- dugumler adında Node[11][13]nesnesi tutacak iki boyutlu dizi oluştur.
- İç içe for döngüsü aç ve x y değerlerini gez.
- Try catch içerisinde düğümler dizisine tablo dizisinin elemanlarını ata.
- İç içe for döngüsü aç ve x y değerlerini gez.
- If koşuluyla haritanın içinde olduğundan emin ol.
- Bir if koşuluyla daha istenilen tablo değerlerim “d” olmadığında addEdge ile kenar ekle.
- String yol değişkeni tanımla.
- Bu değişkeni graph nesnesinin DijkstraShortestPath() metoduna eşitle.
- Yol değişkenini return et.

Edge Sınıfı

- Source ve destination adında 2 adet Node nesnesi ve double tipinde weight değişkeni oluştur.
- Constructor ile atama yap.
- public String toString() fonksiyonu ile nodeların adlarını ve weight’i Stirng tipinde return et.
-

Graph Sınıfı

- Set<Node> nodes ve boolean directed özelliği tanımla.
- Constructor içinde nodes’u initialize et.
- addEdge metodu ile parametre olarak gelen Node nesnelerini ve weight’i nodes Set listesine ekle.
- addEdgeHelper metodunda for döngüsü ile bütün kenarları gez.
- If ile parametre olarak gelen Node nesneleri kaynak ve hedef düğümse;kenar ağırlığına parametre ağırlığını ata.
- Edge nesnelerinin listesine bu parametreleri ekle
- DijkstraShortestPath metodunda parametre olarak başlangıç ve son düğüm nesnelerini al.
- Hashmap sınfından changedAt nesnesi oluştur ve bu nesneye başlangıç key’ine null
-
- En yakın uzaklığı a değer ata.
- Hashmap sınıfından shortestPathMap nesnesi oluştur.
- for (Node node : nodes) ile tüm nodeları gez.
- Node başlangıç düğümüyse; shortestPathMap nesnesine başlangıç düğümünü ve 0.0 ağırlığını, ekle.
- Başlangıç düğümü değilse; artı sonsuz ata.

- for (Edge edge : start.edges) döngüsü ile başlangıçtan gidilebilecek düğümleri gez.
- shortestPathMap'e hedef düğümü ve ağırlığını ekle.
- changedAt'e hedef düğümü ve başlangıç düğümünü ekle.
- start.visit() ile başlangıç düğümümnün isVisited değerini true et.
- While döngüsü aç.closestReachableUnvisited() metoduna shortestPathMap'i gönder.Gelen değeri currentNode'a ata.
- currentNode null ise yol olmadığını yazdır.
- currentNode son düğüme eşitse;child = end atmasını yap.path'e son düğümün adını ata.
- While döngüsü içinde parent = changedAt.get(child) atamasını yap.Eğer parent null' a eşitse döngüyü kır.
- path = parent.getName() + "," + path ile güncel yolu yazdır.child = parent ataması yap.
- path'i return et.
- Anlık node'un isVisited değerini true et.
- Gezilmemiş tüm nodeları gez.Alternatif yol varsa güncelle.
- Artı sonsuza eşitle.
- Ulaşılabilir en yakın düğümü null yap.
- Nodes nesnesinde baştan gezin.
- Node ziyaret edildiyse devam et.
- Edilmediyse şimdiki uzaklık değişkenini shortestPathMap.get(node) eşitle
- Eğer şimdiki uzaklık artı sonsuza eşitse devam et.
- Şimdiki uzaklık en kısa uzaklıktan küçükse en küçük uzaklığa şimdiki uzaklığı ata.
- Ulaşılabilir en yakın düğümü kontrol ettiğim düğüme eşitle.
- En yakın düğümü return et

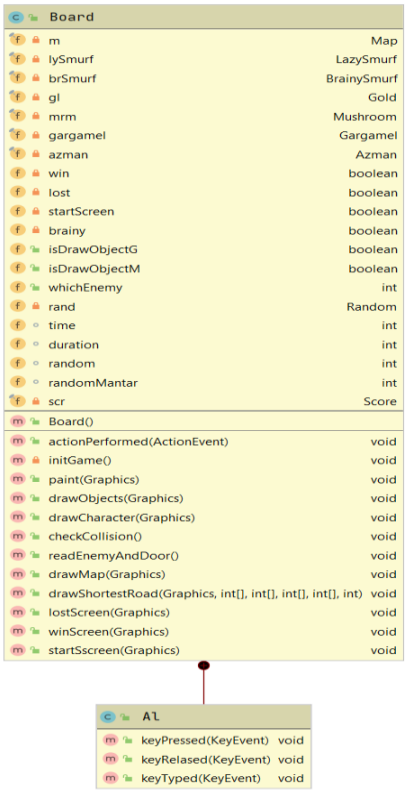
Node Sınıfı

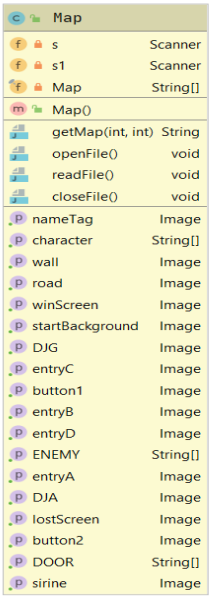
- LinkedListi import et.
- String tipinde name boolean tipinde visited Edge nesnesinin listelerini tutan edges tanımla..
- Constructor içinde name'e atama yap edges'i initialize et.
- visit() unvisit() metodları ile visited'a boolean değerini ata.isVisited() metodu ile visited değerini return et.

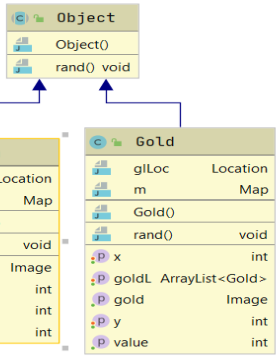
Zaman Karmaşıklığı ve Big O Notasyonu:

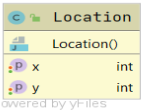
Dijikstra algortimasınd karmaşıklığı düşürmek için Java'nın yığın veri yapısını kullandık.Edge sınıfımıza bir CompareTo () yöntemi eklememizin sebebine bir bakalım.Öncelikle kaç adet kenara sahip olduğumuzu bulmalıyız.Bizim 143 adet node'umuz var her node arasında çift yönlü bağlantı kurulduğundan 2 adet kenar mevcut.Fakat bu node'lardan duvar olanlarla bağlantı yapılmadı.Toplamda 480 adet kenarımız var bunu kenar eklediğimiz metodda bir değişken tutarak hesapladık.Bu yapıda kullanılan kenarları kaldırmak ve yenilerini eklemek o(log(kenar sayısı) kadardır.yani ->O(log(480)) Eğer yığın yapısı kullanmasaydık; ->O(480) olacaktı. Ayrıca, O (sayıOfNodes) yinelememiz var ve bu nedenle PriorityQueue'dan silinme sayısı (O (log (sayıOfEdges)) zamanını alıyor. Tüm kenarlarımızı eklemek de O (log (sayıOfEdges)) süresini alıyor.Bu PriorityQueue kullanırken bize toplam O ((numberOfEdges + numberOfNodes) * log (numberOfEdges)) karmaşıklığını verir.Yani -> O((480+143)x*log(480).=0((623x*log(480) .Eğer yığın yapısı kullanmasaydık; -> O ((numberOfEdges + numberOfNodes) * numberOfEdges) olacaktı.->O((480+143)*480)=O((623x*480).

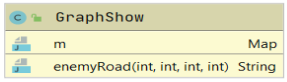
UML Diagramı:

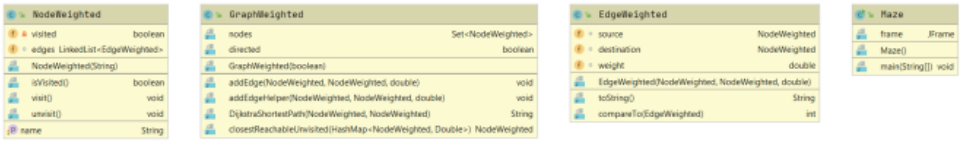


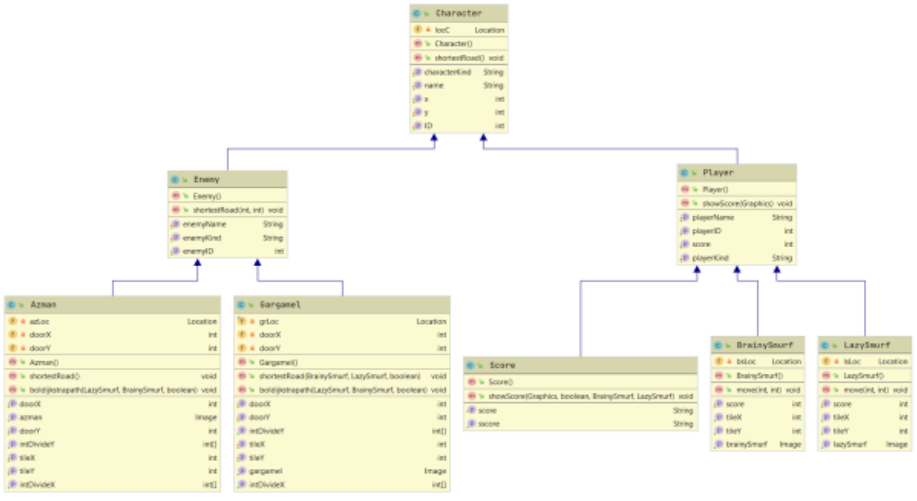

Map
s Scanner
s1 Scanner
Map String[]
Map()
getMap(int, int) String
openFile() void
readFile() void
closeFile() void
nameTag Image
character String[]
wall Image
road Image
winScreen Image
startBackground Image
DJG Image
entryC Image
button1 Image
entryB Image
entryD Image
ENEMY String[]
entryA Image
DJA Image
lostScreen Image
button2 Image
DOOR String[]
sirine Image


Object
Object()
rand() void
Mushroom
mshrmLoc Location
m Map
Mushroom()
rand() void
mushroom Image
x int
y int
value int
Gold
glLoc Location
m Map
Gold()
rand() void
x int
goldL ArrayList<Gold>
gold Image
y int
value int

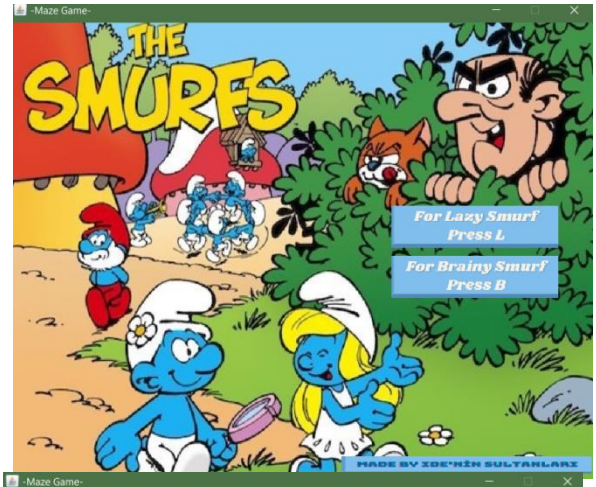

Location
Location()
x int
y int

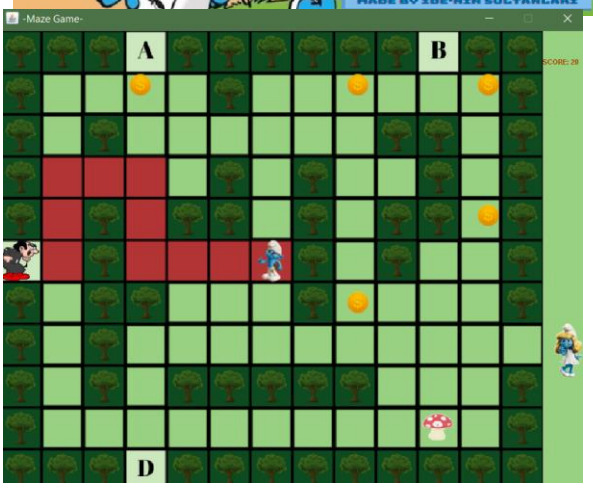

GraphShow
m Map
enemyRoad(int, int, int, int) String

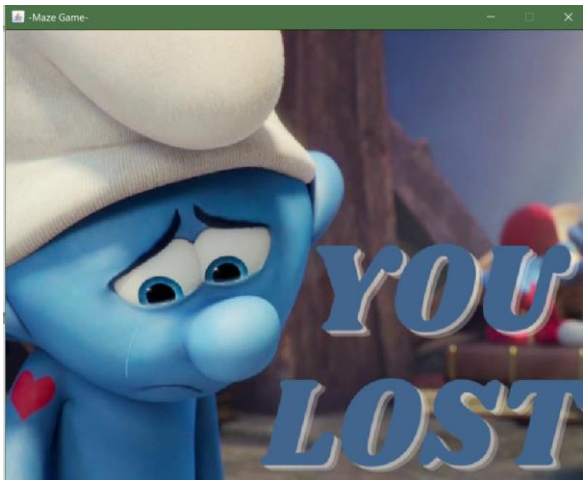

NodeWeighted
GraphWeighted
EdgeWeighted
Maze

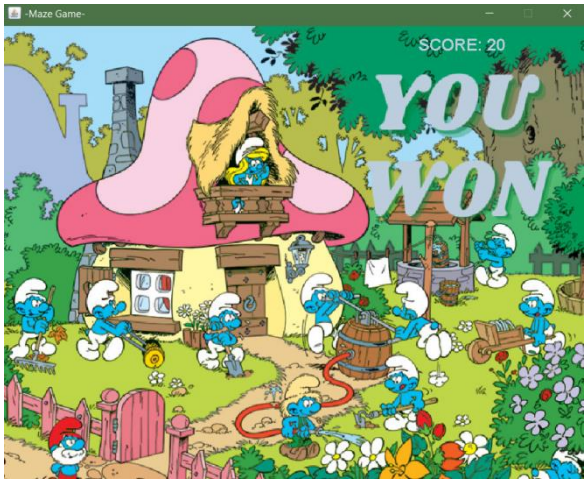

Character
Enemy
Player
Azman
Gargamel
Scarw
BrainySmurf
LazySmurf

Çıktılar:









Kaynakça:

https://zetcode.com/javagames/

https://www.youtube.com/watch?v=64V8CC7nSok

https://stackoverflow.com/questions/18061100/splitting-a-string-in-java-on-more-than-one-symbol/18061175

https://docs.oracle.com/javase/tutorial/2d/images/drawimage.html

https://stackoverflow.com/questions/18249592/how-to-change-font-size-in-drawstring-java