1.1. Problem Tanımı: Bu proje bizden nesneye yönelik programlamanın mantığını daha iyi anlamamız için Encapsulation, Inheritance, Polymorphism, Abstraction yapılarını kullanarak java diline hakim olmamız istenmiştir. Ayrıca Yapıcı (constuctor) metotları (parametreli ve parametresiz olarak en az iki) ve tüm özellikler için get, set metotları kullanılmıştır.

Projede asıl olarak istenen Dijkstra Algoritmasını kullanarak düşman karakterlerle dost karakterler arasındaki dinamik en kısa yolun her hamlede harita üzerinde gösterilmesidir. Ayrıca bizden bir arayüz tasarlamamız istenmektedir. İstenilen haritayı oluşturmak için de graphics.h veya allegro.h gibi kütüphanelerden faydalanılmıştır.

Projede oyuncular ve düşman karakterin ortak özellikleri bulunmakta burada bizden beklenen hiyerarşik bir sınıf mimarisi kurmamızdır. Karakter sınıfı en temel sınıfımız olup, oyuncular ve düşman karakter bu sınıflardan türemiştir.

1.2. Yapılan Araştırmalar: İlk olarak karşılaştığımız problem Dijkstra Algoritmasının matığını anlamaktı. Oyunun akışı gereği her hamlede hem düşman karakter hem de dost karakterin yerleri değiştiği için bu algoritmayı kendi kodumuzca uyarladık. Daha sonra swing kısmı için bizden istenilen matrisi (11-13) panellerden oluşturduk. Gridlayout yapısını araştırarak paneller üzerinde yapmamız gereken işlemleri yaptık (Background rengi, paneller arsındaki çizgiler için kullandığımız border fonksiyonu gib) ve bu panellere karakterlerin resimlerini ekleyip J labellere atarak (Image ıcon fonksiyonu)swing kısmını tamamladık. En son olarak da Keylistenner yapısını kullanarak dost karakterlerimizin hareketlerini klavyemizden hareket ettirdik.

1.3.Tasarım:

1.3.1. Yazılım mimarisi:

1-Bize verilmiş olan harita.txt nin içindeki verileri okut.

2-Panellerin içine attığımız labellerin boyutunu ayarlama(x=76,y=63).

3-Dost karakterin puanın (x=12,y=0). matrsite Font ile yazdır.

4-JFrame tanımla.

5-JPanel tanımlayıp matrisin içini panellerle doldur (11 e 13 lük matris).

6-Bu matrisin içini harita.txtde olduğu şekilde duvar ve yollarla doldur.

7-Duvarların backround’u gray, yolların background’u white.

8-KeyListenner yapısı ile dost karakterlerin hareketini yönet.

9- jFrame.setSize(1000,700) (matris ekran boyutu).

10-Lokasyon sınfıyla matrsite bütün karakterlerin yerlerini belirle.

11-Dizi tanımla ve Collections.shuffle(dizi) ile düşmanların rastgele hangi kapılardan oyuna başlayacağını belirle.

12-ShortestPath sınıfıyla en kısa yol algoritmasını çağır.

13-Operasyon sınıfını çağır ve hangi karakterle oynacağını seç.

14-Random ile altınların ve mantarın nerede çıkacağını belirler.

15-Timer sınıfıyla altınların 5sn de bir, mantarın da 7 sn de bir yerlerini değiştir.

16-Düşman karakterlerden Gargamelin dost karaktere olan en kısa yolun rengi green, Azmanın ise dost karaktere olan en kısa yolun rengi blue.

17- If else yapıları ile dost karakterlerin puanını düşmanlara yakalandıklarında azaltarak, altın veya mantar toplayınca da yükselterek belirle.

1.4.Genel Yapı: Prjemizi kodlarken ilk olarak bizden istenen sınıfları (Dost,Düşman vs) oluşturup sınıfların içlerini gerekli şekilde doldurduk. Daha sonra Inheritance yapısıyla birbirleriyle ilişkili olan sınıflar arasında gerekli kodlamaları yaptık. Bize verilmiş olan harita.txt nin içindekileri try catch ile verileri okuttuk. Sonra bir tane JFrame tanımlayarak içersinde JPanel oluşturup bu JPanelin içini 143 (11\*13) tane JPanel ile doldurduk. Herbirisinin boyut aynı olacak şekilde ayarladık (x=76, y=63). GridLayout yapısı ile her bir panelin içine istediğimiz şekilde ulaşabildik (bazı panellerde işlem yapmamız gerektiği için). Border ile panellerin arasını siyah çizgilerle bölüp genel bir matris yapısı oluşturduk. Sonra KeyListenner fonksiyonu ile dost karakterleri dilediğimiz şekilde (sağ için 6, sol için 4, yukari için 8, asagi için 2) hareket ettirdik. JFrame.setSize ile matrisin boyutunu ayarlayarak jFrame.setDefaultCloseOperation(jFrame.EXIT\_ON\_CLOSE) yapısı ile bu boyutun sabit kalmasını sağladık. -Lokasyon sınfıyla matrsite bütün karakterlerin yerlerini belirledik. Lokasyon sınıfı x ve y ile koordinatları tutan sınıftır. Dizi tanımladık ve Collections.shuffle(dizi) ile düşmanların rastgele hangi kapılardan oyuna başlayacağını belirledik. Daha sonra azman sınıfında ve gargamel sınıfındaki HareketEt fonksiyonu ile hareket etmelerini sağladık (Bu fonksiyon aşağıda açıkalanacaktır). ShortestPath sınıfı yani Dijkstra Algoritması **ağırlıklı bir graf üzerinde** yani kenarları(edge) belli bir metrik değere göre değerleri olan herhangi iki düğüm arası en kısa mesafeyi bulmamızı sağlayan bir algoritmadır. Sonra operasyonlar sınıfı ile ilk başta bir sonraki adımda duvar olup olmadığını kontrol ettirdik (boolean). If else yapısı ile dost karakter prenses şirineye ulaştığı zaman veya dost karakterin puanı<=0 olduğu taktirde oyunun bitip bitmediğini kontrol ettirdik. Lokasyon çevirici ile matris sınıfında oluşturduğumuz 78\*78 lik matrisi Düsman sınıfındaki matrise uyarladık. YolÇiz fonksiyonu ile JPaneller arasında düşman karakterlerin dost karaktere olan en kısa yolu çizdirdik. BufferedImage yapısı ile bütün karakterlerin, altının ve mantarın resimlerini JLabellara uygun boyutlar (x=76,y=63) olacak şekilede ekledik. YolTemizle fonksiyonu ile her adımda en kısa yol temzilenip tekrar en kısa yolu gösterdik. haritaciz fonksiyonu ile her adımda yerleri değişen karakterlerin yerlerini güncelledik. Daha sonra konsolda oyuncuya hangi karakterle oynamak istediğini sorduk. Random fonksiyonu ile altınların ve mantarın nerede çıkacağını belirledik.

Timer sınıfıyla altınların 5 saniyede bir, mantarın da 7 saniyede bir yerlerinin değişmesini sağladık. If else yapıları ile oyuncu hangi dost karakteri seçerse o karakter (gözlüklü şirin istenildiği gibi her adımda 2 adım, tembel şirin ise her adımda 1 adım ilerler) ile istediği şekilde oynaması için if else yapılarının içini gerekli şekilde doldurduk. Oyuncu dost karakteri prenses şirine ulaştırmaya çalışırken başlangıç puanını istenildiği gibi 20 olarak belirledik. Bu süreçte dost karakter düşman karakterlerden gargamele (her adımda 2 adım ilerler) yakalanırsa 15 puan, azmana (her adımda 1 adım ilerler) yakalnırsa da 5 puan kaybederler. Aynı zamanda dost karakter topladığı her altın için 5 puan, her mantar için ise 50 puan kazanır. Dost karakterin puanı<=0 olduğu taktirde oyun bitmektedir. Yani dost karakter puanı 0 olmadan prenses şirine ulaşması gerekmektedir.

1.5.Referanslar:

<https://youtu.be/jT3c45XkPTg>

<https://youtu.be/eVV6p0axFYc>

<https://stackoverflow.com/questions/299495/how-to-add-an-image-to-a-jpanel>

<https://docs.oracle.com/javase/tutorial/uiswing/components/border.html>

<https://stackoverflow.com/questions/7702697/how-to-get-x-and-y-index-of-element-inside-gridlayout>

<https://forum.java.com.tr/yazim-alani-awt-textarea-ve-swing-jtextarea-siniflari/>

<https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

<https://youtu.be/SZXXnB7vSm4>

Algoritma Karmaşıklığı:

