



VİZE-2 ÖDEVİ RAPORU

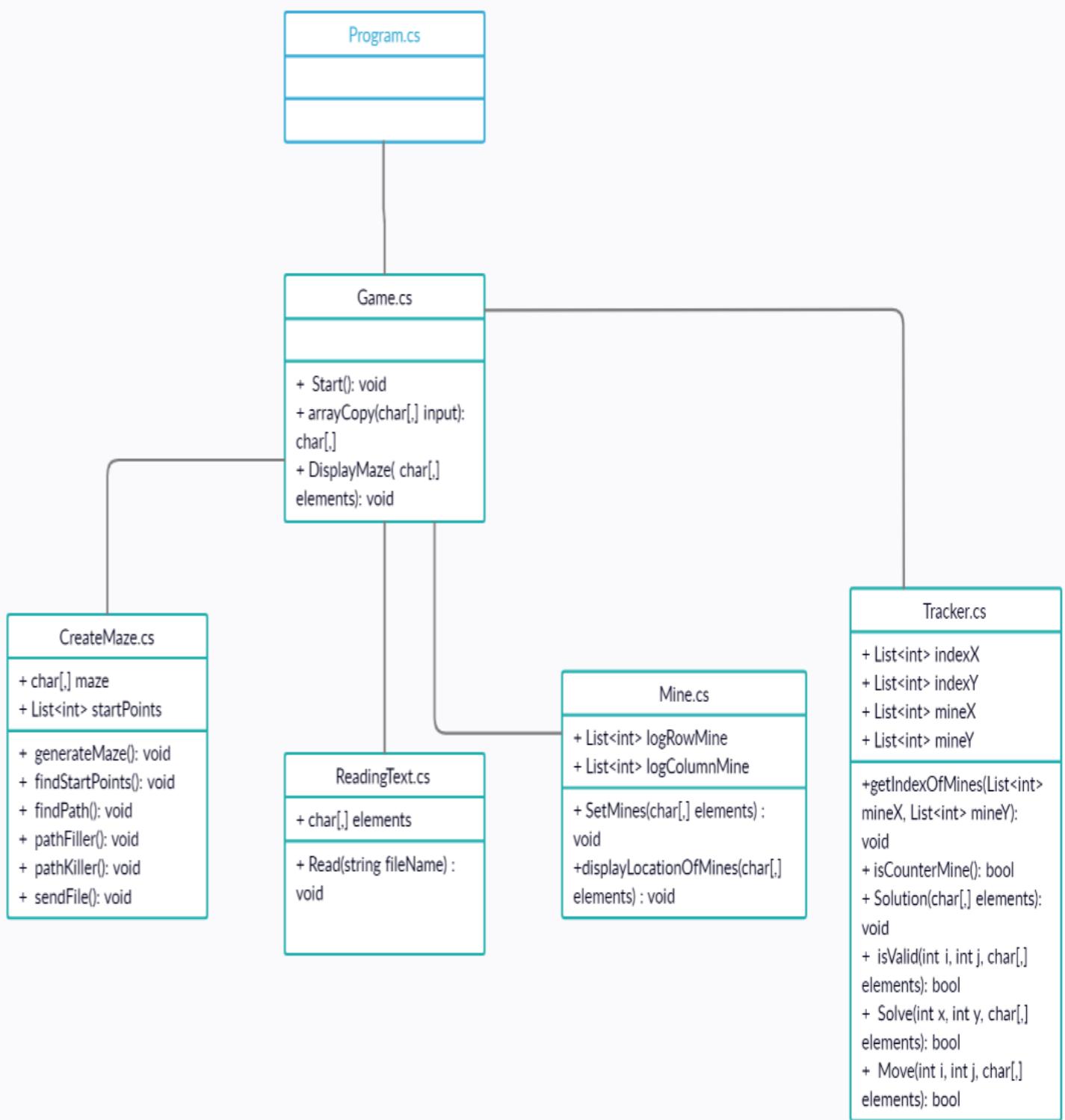


1030510326 BATUHAN EROL
(B GRUBU)

RAPOR İÇERİĞİ

- 1. Temel Sınıf Dizayn Diyagramları**
- 2. Algoritma Açıklamaları**
- 3. Backtracking Algoritması**
- 4. Kaynakça**

1. Temel Sınıf Dizayn Diyagramları



2. Algoritma Açıklamaları

Labirent ödevi 2 ana istere dayanmaktadır. Bunlardan ilki bir txt dosyası ile verilen labirentin okunup çözülmesi, ikincisi ise bir labirent üretip üretilen labirenti bir txt dosyasına uygun formatta yazmaktır.

1. Labirentin Okunması ve Çözülmesi

Bu bölümün isterlere uygun bir şekilde yapılabilmesi amacıyla 3 farklı sınıf kullanılmıştır. Bunlar, `ReadingText.cs`, `Mine.cs`, `Tracker.cs` 'dır.

ReadingText.cs

- Bu classın işlevi girilen dosyayı uygun formatta okumaktır. Txt dosyasındaki yolu ve duvarı temsil eden 0 ve 1 değerlerini 2 boyutlu char veri tipinde bir diziye atar.

Mine.cs

- İki properties ve iki methoddan oluşmaktadır.
- `SetMines()` methodu 3 tane biribirinden farklı x ve y ekseninde index üretir ve bu değerleri `logRowMine` ve `logColumnMine`'a atar. Böylece index değerleri kayıt altına alınmış olur.
- `displayLocationOfMines()` methodunda ise mayınların bulunduğu lokasyon değerini 'B' olarak değiştirir.

Tracker.cs

- Dört properties ve 6 methoddan oluşmaktadır.
- `getIndexOfMines()` methodu int veri tipinde iki tane List'i parametre olarak alır. Bu parametler `Game.cs` sınıfında `Mine.cs` sınıfının propertieslerini alacaktır. Böylece mayınların olduğu bölgelerin indexleri `mineX` ve `mineY` adlı propertieslerimizde tutulacaktır.
- `isCounterMine()` methodu, `indexX` ve `indexY` de kaydedilen indexleri `mineX` ve `mineY` ile kıyasarak true veya false döndürür. Burdan gelen boolean değere göre mayına basıp basılmadığını anlarız. Eğer mayına basılırsa `Solution()` methodu içerisindeki koda sayesinde sistem kendini kapatır. Yani çıkış yapar.
- `Solution()` methodu, ana hatlarıyla anlatmak gerekirse labirentin başlangıç noktalarından aldığı indexleri parametre olarak gönderek oradan gelen değere göre mayına çarpıp çarpmadığını, labirentin çözülmüş olduğunu kontrol eder. Kısacası `Tracker.cs` sınıfının ana hattını oluşturur.
- `isValid()` methodu, bu methodun aldığı parametler indexleri temsil eder ve bu indexleri kullanarak system boundarynin aşılıp aşılmadığını ve istenilen indexteki değerin yolu temsil edip etmediğini kontrol ederek bize true veya false değer döndürür.

- Solve() methodu, Move() methodundan gelen nihai sonuca göre bizlere true veya false değer döndürür.
- Move() methodu, bu method ana algoritmamızı oluşturur. Her defasında komşu elemanları kontrol eder ve gidilen yerleri işaretler. Eğer çıkmaz yola girerse önceki adımlara geri döner. İşlem sonucunda true değerler döndürür ise buradaki indexleri indexX ve indexY'e kaydedeler. Burada backtracking algoritması mantığı kullanılarak bu işlev kazandırılmıştır.

2. Labirent Üretimi

Bu bölümün isterle uygun bir şekilde yapılabilmesi amacıyla CreateMaze adında bir tane sınıf kullanılmıştır.

CreateMaze.cs

- İki properties ve 6 methoddan oluşmaktadır.
- generateMaze() methodu, veri tipi char olan iki boyutlu maze değişkenimize '#' değerini atayarak 30x30'luk bir matris oluşturur.
- findStartPoints() methodu, 4 ile 10 sayıları arasında olmak ile birlikte random başlangıç değerleri üretir ve bunu startPoints adındaki listeye ekler bunlar labirentin başlangıç noktaları olacaktır.
- findPath() methodu, başlangıç noktalarını kullanarak kendisine sağa,sola ve aşağı gitmek kaydıyla random bir şekilde yol çizer. Yol çizme işlemi matrisin 29. yani son satırına gelindiği zaman durur ve geçen kısımlara '0' değerini atar ve son olarak geriye kalan '#' değerini '1' e çevirir.
- pathFiller() methodu, art arda gelen her beş duvarın yani '1' değerinin üçüncü ve dördüncü değerini sıfır yaparak labirentin çözümünü zorlaştırma işlevi görür.
- pathKiller() methodu ise bir önceki bahsedilen methodla aynı amacı taşımaktadır, startPoints.Count/2 kadar son satırda rastgele indexlerdeki '0' yani çıkış değerlerini duvar ('1' değeri) yaparak olası çıkış yollarından birkaçını kapatır.
- sendFile() methodu ise üretilen labirentini gereklili dosya dizininde bulunan txt dosyasına yazmak ile sorumludur.

3. Parçaların Birleştirilmesi

İlk iki bölümde kullanılan sınıflar sayesinde istenilen görevler yerine getirilmiştir. En son olarak tüm bu parçaların birleştirilmesi ve bir menü oluşturmak için Game.cs adında bir sınıf tasarlanmıştır. Bu sınıf 3 tane method içermektedir ilk methodumuz olan Start() methodunda bizlere seçenekler sunulmuştur. Bu seçenekler, Kullanma Kılavuzu, Labirent Üretme, Labirent Çözme ve Çıkıştır.

Kullanıcıdan klavye yardımıyla bu özellikleri kullanması için belirtilen tuşlara basılması istenmiştir. Eğer kullanıcı Labirent Çözme seçeneğini seçeirse (tuş3) bize labirenti çözüp çözmediğine veya mayına basıp basmadığını dair bilgi verecektir. Eğer mayına basarsa beep sesi vasıtasyla uyarı verecek ve ekranı kapatacak, labirenti çözer ise bize çıkış yoluna götüren yolu verecektir ve karşımıza 3 tane farklı seçenek çıkaracaktır bu seçenekler vasıtasyla Labirenti Çözen Patikayı, Mayınların Lokasyonunu Labirent'in üzerinde görebilme ve labirentin orijinal halini görebilme olanağına erişmiş oluyoruz. `arrayCopy()` methodu, iki boyutlu olan char veri tipinde bir diziyi kopyalayabilmemizi sağlıyor. `DisplayMaze()` ise iki boyutlu char veri tipinde değeri uygun formatta ekrana basmamızı sağlıyor.

3. Backtracking Algoritması

Bu kısımda labirenti çözmek için kullandığım backtracking algoritmasının detaylarına deyineceğim. Backtracking algoritmasını kısaca tanımlamak gerekirse, “*Geri izleme, çözümlere aşamalı olarak adaylar oluşturan ve adayın geçerli bir çözüme tamamlanamayacağını belirlediği anda adayı terk eden, özellikle kısıt memnuniyet sorunları olmak üzere bazı hesaplama sorunlarına çözüm bulmak için genel bir algoritmadır.*”

Bu algoritma özellikle labirent çözme uygulamaları için biçilmiş bir kaftandır. Bizim kodumuzdaki implementasyonu ise, ilk etapta Tracker.cs sınıfındaki methodlarımızdan `Solution()` başlangıç noktalarının sütun indexlerini `Solve()` methoduna parametre olarak gönderir ve `Solve` methodunda aldığı bu parametleri `Move()` methoduna gönderir, `Move()` methodu ise sol, aşağı, sağ ve yukarı kısımlardaki yolları dener ve bu method `isValid()` methodunu kullanarak sınır kontrollerini ve index değerleri üzerindeki değerin yol olup olmadığını kontrol eder ve buna göre true veya false değer döndürür. Çıkmaz noktaya ulaşırsa geçtiği yollardan geri döner ve alternatif yolları dener. İşlem sonucunda çıkışa ulaşırsa true döndürür ulaşamazsa false döndürür. Böylece backtracking algoritması başarılı bir şekilde implemente etmiş olur.

Peki eğer bize yolu çözerse, patikanın satır ve sütun değerlerini nasıl veriyor? En son satır değerinin indexi olan 29'a ulaştığımız zaman ve buradaki değerimiz yol '0' değeri olduğu zaman bize true değerini döndürecek ve bu indexleri `indexX` ve `indexY`'a kaydedecekтир. Bu işlemden sonra önceki gidişatlara geri dönüp önceki işlemlerin girip indexlerini kayıt edecek ve true döndürecekтир. Bu işlemi başlangıç noktasına kadar yapacaktır. Buradan anlaşılacağı üzere tersten bir kayıt yapma söz konusudur.

4. Kaynakça

- <https://bilgisayarkavramlari.com/2009/11/01/geri-izleme-algoritmasi-backtracking-algorithm/>
- <https://www.youtube.com/watch?v=UKnNYDxoRjU&t=309s>
- <https://github.com/burakbozb1/C-backtracking-solve-maze>
- <https://www.geeksforgeeks.org/backtracking-introduction/>
- <https://www.youtube.com/watch?v=DKCbsiDBN6c&t=431s>
- https://www.youtube.com/watch?v=oF_mLSgseQ0&t=488s
- <https://www.kodlamamerkezi.com/c-net/c-ile-dosya-okuma-ve-yazma-islemleri/>