



**KOCAELİ ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ**

**YAZILIM LABORATUVARI-1 PROJE -1  
ALTIN TOPLAMA OYUNU**

**ENGİN YENİCE  
190201133**

**CEMRE CAN KAYA  
190201137**

# Altın Toplama Projesi

Cemre Can Kaya  
Bilgisayar Mühendisliği  
Kocaeli Üniversitesi  
190201137  
[cemre\\_cankaya@hotmail.com](mailto:cemre_cankaya@hotmail.com)

Engin Yenice  
Bilgisayar Mühendisliği  
Kocaeli Üniversitesi  
190201133  
[enginyenice2626@gmail.com](mailto:enginyenice2626@gmail.com)

**Özet—** Altın Toplama Oyunu;  $m \times n$  boyutlu bir dikdörtgen tahta üzerinde farklı özelliklere sahip olan oyuncuların altın toplama yarışına dayanır. Bu yarış bir simülasyon şeklinde çalışır.

**Keywords—** *A oyuncusu, B oyuncusu, C oyuncusu, D oyuncusu, Oyun alanı, Hedef belirleme*

## I. GİRİŞ

Oyun C# Programlama dili kullanılarak nesne tabanlı programlama mantığı ile geliştirildi. Oyun kullanıcının belirlediği  $m \times n$  karelik bir oyun alanında ve belirlenen kurallara göre bir simülasyon şeklinde çalışmaktadır. Kullanıcı oyun alanının boyutunu ve oyuncuların özelliklerini belirledikten sonra oyun başlatılır oyun sonunda oyuncuların tüm hareketleri ve tüm bilgileri ekrana tablo olarak yansıtılır. Bu bilgiler her kullanıcı için ayrı ayrı dosyalara yazdırılır.

Oyuncuların her biri farklı şekilde çalışmaktadır;

**A oyuncusu:** Altın toplamak için kendisine en yakın olan altını seçer.

**B oyuncusu:** A oyuncusu gibi kendine en yakın altını seçer ama kendine en yakın ve en kârlı olan altını seçer.

**C oyuncusu:** B oyuncusu ile aynı şekilde en yakın ve en kârlı hedefi seçer. C'nin bir yeteneği bulunmakta. C sıra her kendisine geldiğinde haritada bulunan gizli altınlardan varsayılan olarak 2 adet gizli altını görünür hale getir. Hedefini bu işlemten sonra belirler.

**D oyuncusu:** D oyuncusu hedef belirlerken ilk olarak diğer oyuncuların hedeflerine bakar. Eğer rakiplerinin hedeflerine onlardan önce gidebiliyorsa en yakın veya en kârlı olmasına bakmaksızın rakibinin hedefine gider. Rakiplerinin hedeflerine onlardan önce varamıyorsa B oyuncusu gibi en yakın ve en kârlı altını seçer.

Oyun başlatıldığında oyun alanında altın kalmayınca veya tüm oyuncular elenene kadar simülasyon devam eder. Oyun bittiğinde oyunun sonuçları, oyuncuların oyun sonu bilgileri kullanıcıya oyun bitiminde gösterilir aynı zamanda oyunun bulunduğu konumda her kullanıcı için farklı bir dosyada tüm bilgileri kaydedilir.

## II. YÖNTEM

### A. Başlangıç Ekranı

Başlangıç ekranında kullanıcı oyunun ayarlarını belirler. Başlangıç ekranı oyun ayarları ve oyuncu ayarları bölümlerinden oluşur.

Oyun Ayarları	
Oyun alanı genişliği	20
Oyun alanı yüksekliği	20
Altın oranı	% 20
Gizli altın oranı	% 10
Başlangıç Altını	200
Maksimum Adım	3

Şekil 1 : Oyun ayarları

Oyun ayarları bölümünde kullanıcı, oyun alanının genişlik ve yüksekliğinin kaç kareden oluşacağını belirler. Oyun alanında bulunan karelerin yüzde kaçında altın bulunacağını ve bu altınların yüzde kaçının gizli olacağını belirler.

Tüm oyuncuların başlangıçta kasasında kaç altın ile başlayacağını belirler. Bir hamlede yani bir turda maksimum yapabileceği hareket sayısını belirler.

Eğer oyun ayarları değiştirilmez ise oyun varsayılan olarak belirlenen değerler ile başlatılır. Şekil 1' de belirtilen değerler varsayılan değerlerdir.

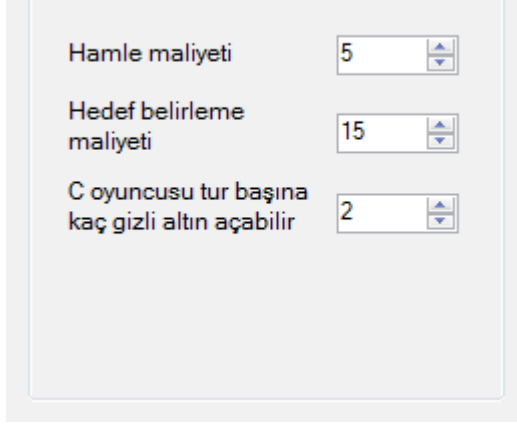
A Oyuncusu özellikleri	
Hamle maliyeti	5
Hedef belirleme maliyeti	5

Şekil 2: A oyuncusu ayarları

Oyuncu ayarları bölümünde ise oyuncuların bir hamle yaptıklarında harcamaları gereken hamle maliyeti ve bir hedef belirlerken harcamaları gereken hedef bulma maliyeti belirlenir. Bu değerler her oyuncu için ayrı ayrı belirlenir. Şekil 2: A oyuncusu ayarları 'de A oyuncusunun değerleri gösterilmiştir.

Her oyuncu varsayılan olarak hamle maliyeti için 5 altın harcar. Ama varsayılan hedef belirleme maliyeti her oyuncu için farklıdır bu değerler;

- A hedef belirleme maliyeti: 5
- B hedef belirleme maliyeti: 10
- C hedef belirleme maliyeti: 15
- D hedef belirleme maliyeti: 20

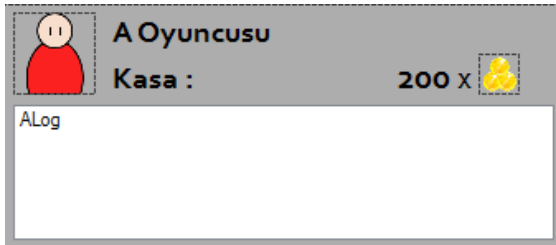


Şekil 3: C oyuncusu ayarları

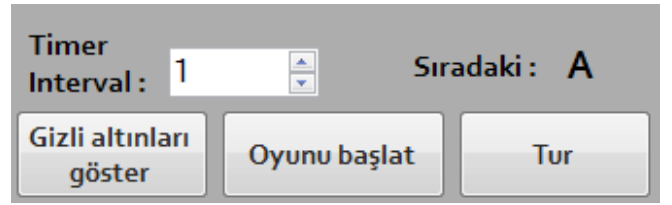
C oyuncusunun kendi özelliği olan gizli altın açma yeteneğinin, diğer oyunculara ek olarak kaç altın açacağı kullanıcı tarafından belirlenir. Eğer kullanıcı belirlemez ise **Şekil 3'** te de belirtildiği gibi C oyuncusu varsayılan olarak her tur en yakın 2 gizli altını görünür hale getirir.

#### B. Oyun ekranı

Oyun ekranının sol tarafındaki panelde oyuncuların bilgileri gösterilir. Oyunu başlat, oyun hızı, gizli altınları göster, tur gibi butonlar bulunur oyun bu butonlar aracılığıyla kontrol edilir. Ekranın geri kalan kısmındaki panel oyun tahtasının görüntülendiği, oyunun görsel olarak oynandığı bölümdür.



Oyuncunun kasasında bulunan altın her tur güncellenir. Oyuncunun yaptığı her hamle, hedef belirleme gibi bilgilerin tüm kayıtlarını alta bulunan listbox' ta son yapılan hamle en yukarıda olacak şekilde tersten listelenir. Listede altlara gidildikçe geçmiş hamleler görülebilir.



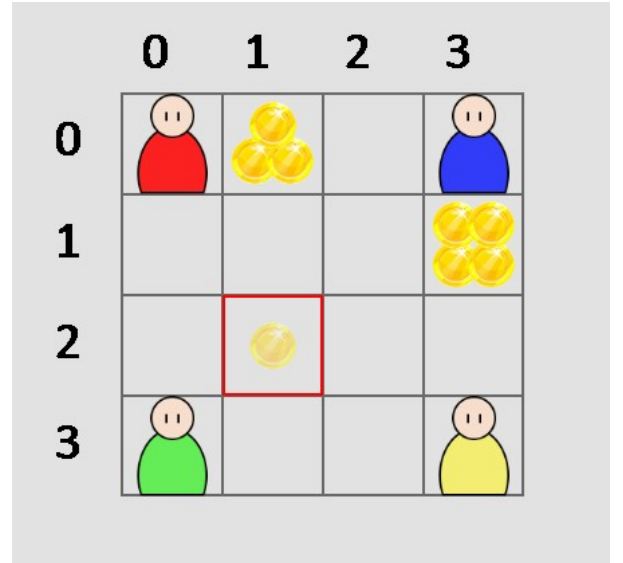
**Timer Interval:** Oyun başlatıldığında oyunun her hamle arasında ne kadar zaman aralığı olacağını belirler. Varsayılan değer 400 milisaniyedir.

**Gizli altınları göster:** Oyun alanı ekranında gizli altınları görünür hale getirir ya da gizler. Sadece kullanıcının gizli altının yerini öğrenmesi için, oyun içerisinde gizli altın oyuncuların görebileceği hale gelmez.

**Oyunu başlat:** Oyunu başlatır ya da durdurur.

**Tur:** Tur butonu her tıklandığında sıradaki oyuncunun hamlesini yapmasını sağlar. Her tıklama tek bir hamle yapmayı sağlar oyun oynamaya devam etmez.

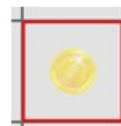
**Sıradaki:** Oyun sırasının kimde olduğunu gösterir. Bir sonraki hamleyi hangi oyuncunun yapacağını gösterir.



Oyun ekranı dinamik bir biçimde oyun ayarlarında girilen genişlik, yükseklik değerlerine göre oluşturulur ve panelde ortalanan, çizdirilir.

Altın resimlerindeki 1 altın aslında 5 değerinde altını temsil eder.

 = 5 Altın  = 10 Altın

 = Gizli altını ifade eder.

ScoreBoard	A Oyuncusu	B Oyuncusu	C Oyuncusu	D Oyuncusu
<b>Toplam Adım Sayısı</b>	2	2	1	1
<b>Kazanan Altın Miktarı</b>	10	20	10	10
<b>Harcanan Altın Miktarı</b>	20	30	20	25
<b>Kasadaki Altın Miktarı</b>	190	190	190	185
<div> <div>A Hareket Kayıtları</div> <div>B Hareket Kayıtları</div> <div>C Hareket Kayıtları</div> <div>D Hareket Kayıtları</div> </div>				

Oyun bittikten sonra ekrana skor tablosu getirilir. Skor tablosunda her oyuncunun oyun boyunca yaptığı;

- Toplam hamle,
- Toplanan tüm altınlardan kazandığı altın değeri,
- Hedef belirleme ve hamleler için harcadığı toplam altın değeri,
- Elde ettiği kâr ya da zarara göre kasasında kalan altın değeri,

Gibi bilgiler listelenir.

Alt bölümde oyuncuların oyun boyunca yaptığı tüm hareketleri listelenir.

Oluşturulan tüm oyun çıktıları oyunun bulunduğu konumda “GameLog” adında bir klasör içerisinde her oyuncu için ayrı ayrı .txt dosyası olarak yazdırılır.

Çıktı dosyası formatı;

```

| 0 | 1 | 2 | 3 | 4 |
| 0 | A |   |   | B |
| 1 |   |   |   |   |
| 2 |   |   |   |   |
| 3 |   | 10 |   |   |
| 4 | C |   |   | D |

#### Oyuncu Bilgileri ####
Oyuncu adı : A
Oyuncu hedef belirleme maliyeti : 5
Oyuncu adım maliyeti : 5
Oyuncu başlangıç altını : 200

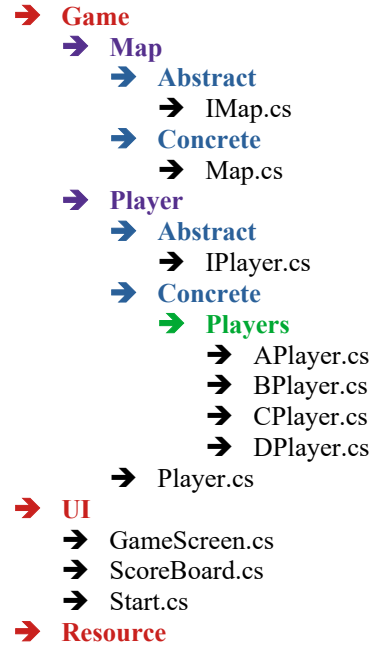
#### Oyun Sonucu ####
Toplam adım sayısı : 2
Toplam kazanan altın : 10
Toplam harcanan altın : 20
Kasada bulunan altın : 190

#### Oyun Bilgisi ####
1) X: 0, Y: 0 koordinatından 200 altın ile oyuna katıldı.
2) Hedef belirlemek için 5 altın harcadı.
3) Hedef: X:0 Y:3 olarak belirlendi.
4) Tahmini Kazanç: 0 Altının Değeri: 10
5) Hedefine ulaştı.
6) Hedef belirlemek için 5 altın harcadı.
7) Hedef: X:4 Y:2 olarak belirlendi.
8) Tahmini Kazanç: -5 Altının Değeri: 10
9) Hedefine ulaşması için 1 adım kaldı.
10) Oyun alanında altın kalmadı.
11) Oyun Bitti

```

Projenin yönetilmesi açısından proje temel olarak **Tasarım ve Yönetim** olmak üzere 2 yapıya bölünmüştür.

Projenin tüm yönetsel işlemleri **Game** klasörü içerisinde ve tasarımsal yapısı **UI** klasörü içerisinde gerçekleştirilmektedir. Bunun yanında projede bulunan bütün görsel dosyalar **Resource** klasörü altında bulunmaktadır.



Klasörler kendilerine bulundukları katmanlara göre renklendirilmişlerdir. Klasör isimleri kalın yazı biçimi ile belirtilmiştir.

#### IV. PROJEDE BULUNAN SINIFLAR VE FONKSİYONLARI

Proje içerisinde class, interface tipinde sınıflar ve abstract metotlar kullanılmıştır.

##### A. Interface

Nesne yönelimli programlama dillerinde arayüz, değişik sınıflardan nesnelerin kategorize edilmesini sağlayan bir soyut tür çeşididir.

##### B. Abstract

Abstract Metot, sadece Abstract Class'ların içerisinde tanımlanabilen, tanımlanırken gövde bulundurmayan, daha sonra içerisinde bulunduğu Abstract Class kalıtılarak override edilen Metotlardır. Abstract Metotlar, Private olarak tanımlanamaz.

##### C. Class

Sınıf, nesne yönelimli programlama dillerinde nesnelerin özelliklerini, davranışlarını ve başlangıç durumlarını tanımlamak için kullanılan şablonlara verilen addır. Bir sınıftan türetilmiş bir nesne ise o sınıfın örneği olarak tanımlanır. Sınıflar genelde şahıs, yer ya da bir nesnenin ismini temsil ederler.

#### D. Map/Abstract/IMap.cs (Interface)

Map sınıfını kategorize etmek için kullanılan bir sınıftır. Map sınıfı içerisinde bulunması gerek metotları belirtmektedir.

#### E. Player/Abstract/IPlayer.cs (Interface)

Player sınıfını kategorize etmek için kullanılan bir sınıftır. Player sınıfı içerisinde bulunması gerek metotları belirtmektedir.

#### F. Map/Concrete/Map.cs (Class)

IMap interface inden kalıtım almaktadır.

##### 1) Map.cs Metotları

###### a) GET Metotları

**string[,] GetMap():** Oyun haritasını 2 boyutlu string tipinde dizi olarak geri döndürür.

**int[,] GetGoldMap():** Oyunda bulunan altınların haritasını 2 boyutlu int tipinde dizi olarak geri döndürür.

**int[,] GetPrivateGoldMap():** Oyunda bulunan gizli altınların haritasını 2 boyutlu int tipinde dizi olarak geri döndürür.

**int[] GetPlayerTarget(string playerName):** Parametre olarak gönderilen oyuncunun hedeflediği altının koordinat bilgilerini dizi olarak geriye döndürür.(0 indexi Y düzlemindeki koordinatı, 1 indexi X düzlemindeki koordinatı temsil eder.)

**int GetPlayerRemainingSteps(string playerName):** Parametre olarak gönderilen oyuncunun hedeflediği altına kalan adım sayısını geriye döndürür.

**int GetGoldCount():** Oyun alanında bulunan görünür altınların sayısını geriye döndürür.

**int GetPrivateGoldCount():** Oyun alanında bulunan gizli altınların sayısını geriye döndürür.

**string GetMapPointValue(int CordY, int CordX):** 2 boyutlu Harita dizisi üzerinde girilen koordinat parametrelerinde bulunan değeri geri döndürür.

**int GetGoldPointValue(int CordY, int CordX):** 2 boyutlu Altın Haritası dizisi üzerinde girilen koordinat parametrelerinde bulunan altın değerini geri döndürür.

**int GetPrivateGoldPointValue(int CordY, int CordX):** 2 boyutlu Gizli Altın Haritası dizisi üzerinde girilen koordinat parametrelerinde bulunan gizli altın değerini geri döndürür.

**int GetGameOrder():** Oyun sırasının hangi oyuncuda olduğunu geriye döndürür.

0. A
1. B
2. C
3. D

**bool IsFull():** Oyun haritası tamamen dolu olup olmadığını geriye döndürür.

False = Haritada boş yer var.

True = Harita tamamen dolu.

**bool GetgameOver():** Oyunun bitip bitmediğini geriye döndürür.

False = Oyun bitmedi.

True = Oyun bitti.

**string GetgameOverReason():** Oyunun bitiş sebebini metin olarak geriye döndürür.

**string GetMapString():** Oyun haritasını string tipine dönüştürerek geriye döndürür.

##### b) SET Metotları

**void SetPlayerTarget(int targetY, int targetX, string playerName):** Parametre olarak gelen koordinatları parametre olarak gelen oyuncuya hedef olarak belirler.

**void SetPlayerRemainingSteps(int steps, string playerName):** Parametre olarak gelen adım sayısı ve oyuncu adına ait bilgiler ile oyuncunun hedefe kalan adım sayısını değiştirir.

**void SetGameOrder():** Oyun sırasını değiştirir.

##### c) REMOVE Metotları

**void RemovePlayersIsDeath(int gameOrder):** Parametre olarak gelen sırayı kontrol eder ve altını bitip oyunu kaybeden oyuncuyu sıradan çıkarır.

**void RemoveGoldPoint(int CordY, int CordX):** Parametre olarak girilen koordinatları Altın Haritasında 0 olarak değiştirir.

**void RemovePrivateGoldPoint(int yCord, int xCord):** Parametre olarak girilen koordinatları Gizli Altın Haritasında 0 olarak değiştirir.

##### d) UPDATE Metotları

**void UpdateGoldMapPoint(int CordY, int CordX, int data):** Altın haritasında parametre olarak gelen koordinatlardaki değeri parametre olarak gelen data değeri ile değiştirir.

**void UpdateMapPointData(int CordY, int CordX, string data):**Tüm ögelerin bulunduğu haritayı parametre olarak gelen koordinatlardaki değeri parametre olarak gelen data değeri ile değiştirir.

**void UpdatePrivateGoldMapPoint(int CordY, int CordX, int data):** Gizli Altın haritasında parametre olarak gelen koordinatlardaki değeri parametre olarak gelen data değeri ile değiştirir.

**void AddPlayer(int CordY, int CordX, string PlayerCode):** Parametre olarak gelen koordinatlara parametre olarak gelen oyuncuyu ekler.

**void AddAllGold(int GoldRate, int PrivateGoldRate):** Oyuna parametre olarak %? altın ve bu altınların %? kaçının gizli olacağı parametre olarak gelmektedir. Bu hesaplamaların ardından altınlar haritalarına eklenecektir.

#### G. Player/Concrete/Player.cs (Class)

IPlayer interface inden kalıtım almaktadır.

##### 1) Player.cs Metotları

###### a) GET Metotları

**String GetName():** Oyuncunun adını geriye döndürür.

**int[,] GetPlayerMatris():** Oyuncunun konumunun bulunduğu int tipinde 2 boyutlu diziyi geriye döndürür.

**string GetPlayerMapString():** Oyuncunun konumunun bulunduğu diziyi string bir formata çevirerek geriye döndürür. Oyuncu loglarını oluşturmak için kullanılacaktır.

**int GetPlayerGold():** Oyuncunun kasasında bulunan altın miktarını geriye döndürür.

**int GetGoldEarnedOnReachTarget():** Oyuncunun hedeflediği altına ulaştığında kazanacağı altın miktarını geriye döndürür.

**int[] GetLastCord():** Oyuncunun oyun alanında bulunduğu son koordinat bilgisini dizi olarak geriye döndürür.

0. Y Koordinatı

1. X Koordinatı

olarak belirlenmiştir.

**int GetSearchCost():** Oyuncunun hedef belirleme maliyetini geriye döndürür.

**int GetRemainingSteps():** Oyuncunun hedefine kalan hamle sayısını geriye döndürür.

**List<string> GetLog():** Oyuncunun oyun sırasında yaptığı tüm işlemlerin kaydını geriye döndürür.

**int[] GetTargetedGoldCord():** Oyuncunun hedeflediği altının bulunduğu koordinat bilgisini geriye döndürür.

0. Y Koordinatı

1. X Koordinatı

olarak belirlenmiştir.

**int GetTargetedGoldValue():** Hedeflenen altının değerini geriye döndürür.

**int GetTotalNumberOfSteps():** Oyuncunun oyun boyunca toplam hamle sayısı

**int GetTotalAmountOfGoldSpent():** Oyuncunun oyun boyunca harcadığı toplam altın

**int GetTotalAmountOfGoldEarned():** Oyuncunun oyun boyunca kazandığı toplam altın

##### b) SET Metotları

**void SetPlayerGold(int gold):** Oyuncunun kasasında bulunacak altın değerini belirler. Parametre olarak gelen değeri atama yapar.

**void SetLog(string log):** Parametre olarak gelen hareket bilgisini, oyuncunun işlem geçmişine ekler.

**void SetGoldEarnedOnReachTarget(int gold):** Hedeflenen altına ulaştığında elde edeceği altın miktarını belirler.

**void SetRemainingSteps(int remainingSteps):** Hedeflenen altına ulaşabilmesi için kalan hamle sayısını atama yapar.

**void SetTargetedGoldCord(int CordY, int CordX):** Hedeflenen altının koordinat bilgilerine atama yapar.

**void SetTargetedGoldValue(int goldValue):** Hedeflenen altının değerini atama yapar.

**void SetPlayerMapValue(int CordY, int CordX, int data):** Oyuncu haritasında gelen parametrelerdeki koordinatlara değer ataması yapar.

**void SetTotalNumberOfSteps(int value):** Oyuncunun oyun boyunca yaptığı toplam hamle değerini artırır.

**void SetTotalAmountOfGoldSpent(int value):** Oyuncunun oyun boyunca harcadığı toplam altın değerini artırır.

**void SetTotalAmountOfGoldEarned(int value):** Oyuncunun oyun boyunca kazandığı toplam altın değerini arttırır.

*c) UPDATE Metotları*

**void UpdatePlayerGoldValue(int gold):** Oyuncunun kasasındaki altına ekleme veya çıkartma işlemi yapar.

**void UpdateCord(int yCord, int xCord):** Oyuncunun koordinatını parametre olarak gelen koordinatlar ile değiştirir.

*d) GAME FUNCTION Metotları*

**abstract void SearchForGold(IMap map):** Her oyuncu için özel olarak yazılacak bir metottur. Oyuncunun hedefini belirler.

**void PrivateGoldShow(char duzlem, int hareket, IMap map):** Parametre olarak gelen düzlem üzerinde, parametre olarak gelen hareket değeri kadar ilerler ve o aralık içerisinde gizli altın varsa görünür yapar.

**void Move(IMap map):** Oyuncuyu hedefine doğru hareket ettirir.

**bool IsDeath():** Oyuncunun kasasında bulunan altın durumuna göre oyunda kalıp kalamayacağını belirler.

- Altın değeri sıfıra küçükse veya sıfıra eşitse TRUE
- Altın değeri sıfırdan büyükse FALSE

*e) WRITE TXT Metotları*

**void CreateFolder():** Oyun kayıtlarının çıktı olarak tutulacağı klasör yok ise oluşturur.

**void WriteToFile(string maptext):** Oyunun başlangıç haritasını ve oyuncunun işlem geçmişini txt dosyasına yazar.

*2) Player/Concrete/Players/APlayer.cs (Class)*

Player.cs sınıfından kalıtım almaktadır. Player sınıfında bulunan **SearchForGold(IMap map)** abstract metodunu bulundurmak zorundadır.

**public override void SearchForGold(IMap map):** Kendine en yakın uzaklıktaki altını hedefler.

*3) Player/Concrete/Players/BPlayer.cs (Class)*

Player.cs sınıfından kalıtım almaktadır. Player sınıfında bulunan **SearchForGold(IMap map)** abstract metodunu bulundurmak zorundadır.

**override void SearchForGold(IMap map):** En karlı olan altın kareyi hedefler. Altının uzaklığı ve altın miktarını dikkate alarak belirler.

*4) Player/Concrete/Players/CPlayer.cs (Class)*

Player.cs sınıfından kalıtım almaktadır. Player sınıfında bulunan **SearchForGold(IMap map)** abstract metodunu bulundurmak zorundadır.

**public void PrivateGoldShow(IMap map):** Kendisine en yakın belli sayıdaki gizlenmiş altınların olduğu kutuları açar.

**override void SearchForGold(IMap map):** İlk olarak **PrivateGoldShow** metodunu çağırır ardından, En karlı olan altın kareyi hedefler. Altının uzaklığı ve altın miktarını dikkate alarak belirler.

*5) Player/Concrete/Players/DPlayer.cs (Class)*

Player.cs sınıfından kalıtım almaktadır. Player sınıfında bulunan **SearchForGold(IMap map)** abstract metodunu bulundurmak zorundadır.

**override void SearchForGold(IMap map):** Diğer oyuncuların yapacağı hamleleri önceden sezme yeteneği bulunur. Diğer oyuncuların hedeflediği altınlara onlardan önce erişemiyorsa bu altınları hariç tutar ve hedef olarak diğer altın kareler içerisinde en karlı olanı seçer.

V. KABA KOD

190201137-190201133-Kaba-Kod.pdf olarak teslim dosyaları içerisine eklenmiştir.

VI. KAYNAKÇA

- <https://docs.microsoft.com/tr-tr/dotnet/csharp/how-to/parse-strings-using-split>
- <https://www.yazilimkodlama.com/etiket/c-matris-dizi-olusturma/>
- <https://ferhatkortak.wordpress.com/2013/11/10/c-abstract-nedir-nasil-kullanilir/#:~:text=Abstract%20Method%2C%20sadece%20Abstract%20Class'lar%C4%B1n%20i%C3%A7erisinde%20tan%C4%B1mlanabilen%2C%20tan%C4%B1mlan%C4%B1rken,'lar%2C%20Private%20olarak%20tan%C4%B1mlanamaz.>
- <https://medium.com/codable/interfacelerin-mantigi-nedir-1-hikayeli-f9b960228328>
- <https://www.beyaz.net/tr/ipucu/entry/394/class-nedir>