

# ALTIN TOPLAMA OYUNU

Yazılım Laboratuvarı I

Hilal Özkan 180201017  
Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi  
hilal\_ozkan1200@hotmail.com

Edmond Vujici 170201127  
Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi  
edmondvujic6@gmail.com

## ABSTRACT – ÖZET

**Altın Toplama Oyunu;** mxn boyutlu bir dikdörtgen tahta üzerinde farklı özelliklere sahip olan 4 tanen oyuncunun altın toplama yarışına dayanır.

## I. ANAHTAR KELİMELER

JGameGrid, Altın, Gümüş, Bronz, Elmas, Hedef, Hamle, Oyun Tahtası, Frame, Oyuncu, Sıra, En Yakın Yol, En Karlı Altın, Elenme, Mario, Green, Sonic, Ghost, ArrayList, Strateji, Sezme, Actor.

## II. Giriş

**JGameGrid** ile geliştirdiğimiz altın toplama oyununda ilk olarak bir **Frame** aracılığı ile kasadaki altın miktarı, oyun tahtasının boyutları, altın oranları ve adım sayısı istenir ve verilen verilere göre altın toplama oyunu başlar.

Kullanıcı oyunu oynarken verileri değiştirmek istemez ise oyun varsayılan veriler ile oynanır. İsteğe bağlı olarak oyunun hızını değiştirebilir, durdurabilir ya da oyunu adım adım oynayabilir.

Bu oyunu geliştirme amacımız farklı kısıtlara sahip arama algoritmalarının birbirlerine karşı etkinliklerini gözlemlemek, arama algoritmalarını bir uygulama içerisinde kullanma ve kodlama becerisini geliştirmek ve dinamik özelliklere sahip bir program geliştirmektir.

## III. TEMEL KAVRAMLAR

Proje gelişiminde: Programlama dili olarak JAVA dili kullanılmıştır. Geliştirme ortamı olarak **Apache Netbeans IDE 11.3** kullanılmıştır. Derleyici olarak **JDK 14** kullanılmıştır.

## IV. YALANCI KOD

**Oyun Oynama sınıfı Oyun Izgara sınıfıta kalıtım alır:**

- **global** olarak **4 oyuncu** tanımla
- **global** olarak **görünür altın sayısı** ve **listesi** tanımla
- **global** olarak **görünmez altın sayısı** ve **listesi** tanımla

- **Oyun Oynama sınıfının yapılandırıcısı** satır ve sütun sayısı alır:

- **Oyun Izgara** kalıtım aldığı için ızgara oluşturmak için parametre gönder
- pencerenin başlığı ata
- pencere görünür yap
- **birinci oyuncu** sağ-üst köşesine ekle, etkisiz yap ve gecikmesini belirle
- **ikinci oyuncu** sol-üst köşesine ekle, etkisiz yap ve gecikmesi belirle
- **üçüncü oyuncu** sağ-alt köşesine ekle, etkisiz yap ve gecikmesi belirle
- **dördüncü oyuncu** sol-alt köşesine ekle, etkisiz yap ve gecikmesi belirle
- her **altın** için:
  - rastgele değeri belirle ve sınıfı oluştur
  - rastgele konuma ızgaraya ekle
- her **gizli altın** için:
  - rastgele değeri belirle ve sınıfı oluştur
  - rastgele konuma ızgaraya ekle ve gizle
- **birinci oyuncuya** sıra ver

Her **Oyuncu** sınıfı **Aktör** sınıfından kalıtım alır:

- oyuncunun **başlangıç parası** ata
- **kullanacak para** miktarı için değişken tanımla
- sıra ona gelince varsayılan **adım atması** için değişken belirle
- **toplayacak para** miktarı için değişken tanımla
- **hedef belirlemede kullanacak para** için değişken tanımla
- hedeflediği paranın **XY koordinatı** için değişken tanımla
- hedeflenen paranın **listedeki indisi** bulmak için değişken tanımla
- geçici **XY koordinatı** için değişken tanımla
- parasının durumunu **kontrol** etmek için **ikili değişken** tanımla
- **toplam adım** sayısı için değişken tanımla
- **konum listesini** tanımla

- birinci oyuncunun **yapılandırıcısı** parametre almaz:
  - Aktör (kalıtım) sınıfına görselini gönder
  - **ilk konumunu** listeye ekle
- **act() prosedürünün üzerine yazılır:**
  - **oyuncunun parası** var mı diye kontrol et
  - **oyunun sonuna** geldi mi diye kontrol et
  - eğer oyuncunun parası yoksa:
    - oyuncu etkisiz yap
    - oyuncu ızgarada görünmez yap
    - oyuncunun görseli siyah-beyaz yap
    - sırayı ikinci oyuncuya ver
  - eğer oyuncunun parası varsa:
    - ızgaranın rengini **kırmızı yap**
    - eğer oyuncunun x konumu hedeflediği paranın x konumundan büyük veya küçük ise:
      - **oyuncunun istikametini hedeflediği paraya göre belirle** ve x-eksenine göre hareket ettir
      - oyuncunun her adımdaki **konumu listeye** ekle
      - **attığı adım sayısını** arttır
      - **toplam attığı adım sayısını** arttır
    - eğer oyuncunun x konumu hedeflediği paranın x konumuyla aynı ise:
      - **oyuncunun istikametini hedeflediği paraya göre belirle** ve y-eksenine göre hareket ettir
      - oyuncun her adımdaki **konumu listeye** ekle
      - **attığı adım sayısını** arttır
      - **toplam attığı adım sayısını** arttır
    - eğer oyuncunun x ve y konumu hedeflediği paranın XY konumuyla aynı ise:
      - oyuncuyu durdur
      - para alma **ses efekti** çal
      - paranın değeri **kasasına ekle** değer ekle
      - parayı **ızgaradan ve listeden sil**
      - **attığı adım sayısı** sıfır olarak belirle

- **bir sonraki oyuncunun** attığı adım sayısı sıfır olarak belirle
- hareket ettirmeden başka oyuncuya en **yakın parayı bul** ve hedef olarak belirle
- hamle için 5 para **maliyeti kasasından** çıkar
- eğer oyuncu varsayılan adım attıysa:
  - oyuncuyu durdur
  - **attığı adım sayısı** 0 olarak belirle
  - bir sonraki oyuncuya **sıra ver**
  - **oyunun sonuna** geldi mi diye **kontrol et**
  - her hamle için 5 para **maliyeti kasasından** çıkar
- oyuncunun **gizli bir altınla çarpıştı** mı diye kontrol et

#### koordinat belirleme prosedürü (A Oyuncu):

- **geçici** hedefin **XY değişkenlere** mevcut hedefin **XY değişkenleri** ata
- oyunun **sonuna mı geldi** diye kontrol et
- oyuncuya her paraya uzaklığı hesaplamak için **liste** tanımla
- her **altın** için:
  - oyuncudan altına **uzaklığı hesapla** ve listeye tut
- en küçük uzaklık olarak **listede 0. elemanı** olarak varsay
- **uzaklık listede** her **uzaklık** için:
  - eğer mevcut uzaklık varsayılan uzaklıktan daha küçük ise:
    - **en küçük uzaklık** olarak belirle ve **indisini tut**
- **indisi**, XY koordinatını değişkenlere ata
- eğer hedef varsa:
  - **oyuncuyu hareket ettir**
- oyuncunun parası varsa:
  - ve oyuncu daha önce gittiği hedefe tekrar gidiyorsa:
    - **oyuncunun parası aynı** kalır
  - eğer oyuncu başka bir hedefe gidiyorsa:
    - oyuncunun **kasasından para çıkar**
    - **geçici XY değişkenlere** mevcut olanları ata

- eğer oyuncunun parası yoksa:

- oyuncu elenmiştir.

**gizli altın oyuncu çarpışması kontrol eden prosedürü,** oyuncunun mevcut konumunu alır:

- her **gizli altın** için:
  - eğer oyuncunun mevcut konumu gizli altınla aynıysa:
    - gizli altının **değerini** tut
    - gizli olmayan bir **altın tanımla** ve gizli altının değerini ver
    - görünür altın **listesine gizli altını** ekle
    - görünür altını ızgaraya **ekle**
    - gizli altını **ızgaradan çıkar**
    - gizli altını **listeden çıkar**

**koordinat belirleme prosedürü (B Oyuncu):**

- **geçici** hedefin **XY değişkenlere** mevcut hedefin XY değişkenleri ata
- oyunun **sonuna mı geldi** diye kontrol et
- 2. oyuncuya her paraya uzaklığı hesaplamak için **liste** tanımla
- 2. oyuncuya **en karlı parayı** bulmak için **liste** tanımla
- her altın için:
  - oyuncudan altına **uzaklığı hesapla** ve uzaklıklar **listesine** tut
  - karı bulmak **için paranın değeri ile uzaklığı böl** ve **kar listesini** tut
- en karlı altın 0. indisteki altın olarak varsay
- karlar listesinde her kar için:
  - eğer mevcut kar varsayılan kardan daha büyük ise:
  - mevcut karı en büyük olarak varsay ve indisini tut
- en karlı altının **indisini, XY koordinatlarını** değişkenlere ata
- eğer hedef varsa:
  - oyuncuyu **hareket ettir**
- oyuncunun parası varsa:
  - ve oyuncu daha önce gittiği hedefe tekrar gidiyorsa:
    - oyuncunun parası aynı kalır
  - eğer oyuncu başka bir hedefe gidiyorsa:
    - **oyuncunun kasasından** para çıkar
    - **geçici XY değişkenlere** mevcut olanları ata
- eğer oyuncunun parası yoksa:
  - oyuncu elenmiştir.

**C oyuncunun koordinat belirleme prosedürü B oyuncununkiyle aynı.**

**C oyuncuna en yakın iki gizli altın açma prosedürü:**

- gizli altınlara uzaklıkları hesaplamak için **liste** tanımla
- her **gizli altın** için:
  - oyuncudan uzaklığı **hesapla** ve **listeye ata**
- en yakın gizli altını listede **0. indistekini** olarak varsay
- **gizli altın uzaklıkları** listesinde her uzaklık için:
  - mevcut uzaklık varsayılan uzaklıktan daha küçük ise:
    - mevcut uzaklığı **varsayılan** ata ve **indisini tut**
- gizli **altının koordinatlarını** ve **değerini** tut
- **görünür altın nesnesini** tanımla ve **gizli altının değerini** ver
- görünür altınlar listesine **gizli altını ekle**
- gizli altının koordinatlarına **görünür altını ekle**
- gizli altınlar uzaklıkları **listesini boşalt**

**D oyuncunun altın koordinatları B oyuncununkiyle aynı, fakat diğer oyuncuların hedeflerine ulaşamıyorsa o altına olan uzaklığı -sonsuz olarak yapar, ulaşabiliyorsa uzaklık olduğu gibi kalır.**

Lejant: **sınıf yapılandırıcı veya fonksiyon, tanımlama veya değişken atama, metot**

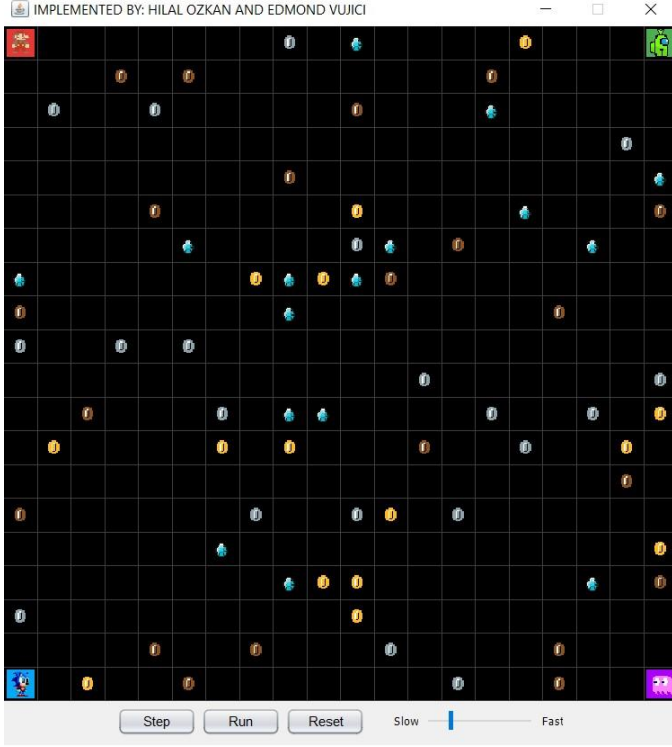
## V. YÖNTEM

Yazdığımız proje **inputScreen** ve **ResultScreen** ara yüzlerinden ve bir tane **gameGrid** ekranından oluşmaktadır. Diğer sınıflarımız da **PlayerA**, **PlayerB**, **PlayerC**, **PlayerD**, **PlayGame**, **Coin**, **GameClass** ve **WriteToFile**'dir.

Altın toplama oyununu **RUN** tuşuna basarak çalıştırdığımızda kullanıcıdan oyun için parametreler

istenir. Eğer bu parametrelerin hepsi veya herhangi biri boş bırakılırsa oyunun bilgileri varsayılan olarak kabul edilir.

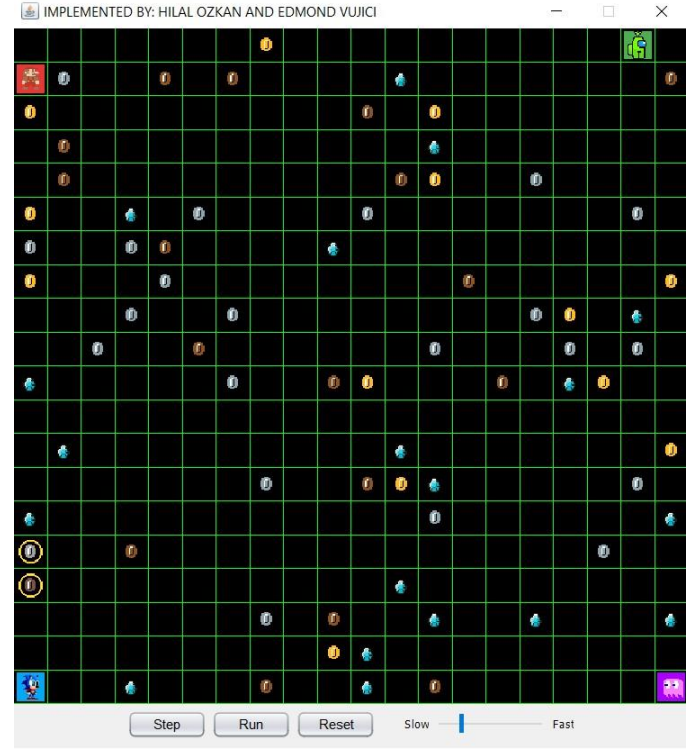
**OK** tuşuna bastığımızda **JGameGrid** kullanarak oluşturduğumuz üzerinde oyuncular ve altınlar olan oyun tahtası gelir. Oyuncularımızın isimleri sıra ile **Mario**, **Green**, **Sonic** ve **Ghost**'tur. Oyun başladığında ilk oynama sırası A oyuncusunda yani **Mario**'dadır.



**Mario**, oyuncular içerisinde en açgözlü olandır, en yakınındaki altını almayı hedefler. Her hamle yaptığında beş altın harcar ve her hedef belirlediğinde de beş altın harcar. Oyuncumuzu hareket ettirmek için **JGameGrid** içerisinde olan **act()** metodunu **override** ettik ve sağa sola aşağı yukarı hareket etmesini sağladık. **act()** metodu içerisinde **PlayerA** sınıfında olan **getCoinCordinate()** fonksiyonunu çağırdık. Bu metot A oyuncusu için en yakındaki altını hedefler ve bu altının koordinatlarına erişmemizi sağlar.

Oyuncumuz için hedef belirlendikten sonra bu hedefe gitmemiz için sıramız geldiğinde bir hamle yapmamız gerekir. Oyuncumuzu hareket ettirmek için **move()** komutunu kullandık ve bu komutu kullandığımız yerlerde **stepTaken** değişkeninin sayısını arttırdık böylece oyuncumuzun kaç adım attığını öğrendik ve adım sayısına eşit olduğunda sırayı B oyuncusuna verdik. Eğer A oyuncusu adım sayısını tamamlamadan hedefine ulaşırsa da sırayı B oyuncusunun **getCoinCoordinate()** fonksiyonunu çağırarak sıranın B oyuncusuna geçmesini

sağladık. Sıra değiştiğinde oyun tahtamızdaki karelerin çerçeve renkleri de oyuncuların renklerini alır bu sayede ıranın kimde olduğunu anlayabiliriz.



B oyuncumuzun rengi yeşil olduğu için karelerin çerçeveleri de yeşil oldu.

B oyuncusu yani **Green** en kârlı olan altını seçer. A oyuncusuna göre daha akıllıdır. Bu yüzden de hedef belirleme maliyeti on altın değerindedir. Her hamle maliyeti beş altın değerindedir.

B oyuncusu **getCoinCoordinate()** metodunda tüm altınların kârlarını hesaplar ve en yüksek kâra sahip olan altını hedef altın olarak belirleyerek bu altının koordinatlarına ulaşır. B oyuncusu hamlesini tamamladıktan sonra sıra C oyuncusuna yani **Sonic** oyuncumuza geçer.

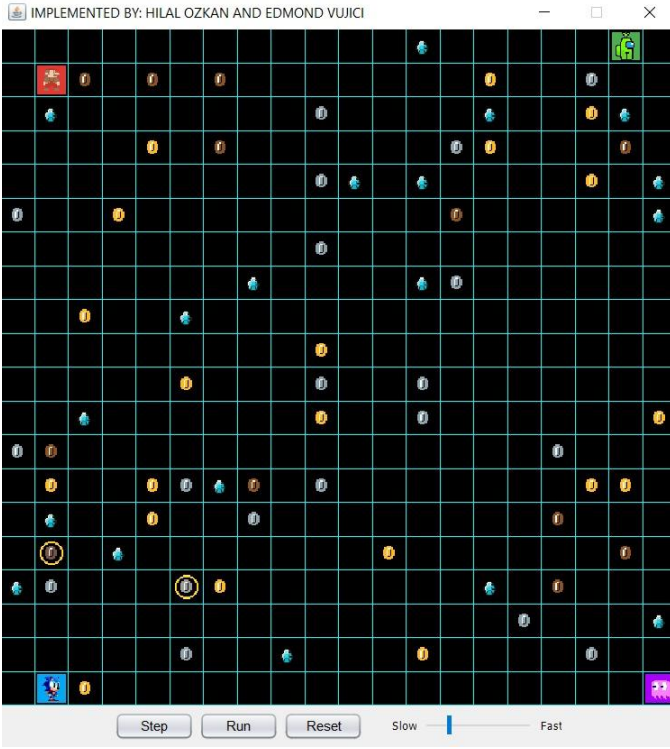
$$en\ karlı = \frac{altın\ değeri}{uzaklık}$$

C oyuncusu da aynı B oyuncusu gibi en kârlı olan altını kendine hedef olarak belirler fakat B oyuncusundan farklı olarak C oyuncusu kendine en yakın olan iki gizli altını **openInvCoin()** fonksiyonu ile açar.

Gizli altınlar açıldıktan sonra görünür altınların olduğu **vMoney ArrayList**'ine eklenir ve görünür hale getirilir. Gizli altınlar, görünür altınlardan ayırt edilebilmeleri için sarı bir halkaya sahipler.







C oyuncusu tarafından açılan gizli altınlar diğer oyuncular tarafından da alınabilir. Gizli altınları bir tek C oyuncusu açmaz. Eğer oyuncular gizli bir altının üzerinden geçerse gizli altın açılır ve görünür olur fakat oyuncu ancak ikinci defa bu altın üzerinden geçtiğinde bu altını alabilir.

**Green** oyuncusu gizli bir altın üzerinden geçmiş ve altın açılmış.

C oyuncusunun bir hamle yapması için altın maliyeti 5 ve hedef belirlemesi için altın maliyeti 15'dir. C oyuncusu hamlesini yaptıktan sonra sıra D oyuncusuna geçer.



D oyuncusu yani **Ghost** en akıllı oyuncudur. Hedef belirlemeden önce diğer oyuncuların hedeflerine olan uzaklıklarına bakar ve eğer kendisi bu hedef altınlarına diğer oyuncularından önce erişemeyecekse o altınları hedef olarak belirlemez yani hedefleyeceği altınlar arasında diğer oyuncuların kendisinden daha önce alabileceği altınlar yoktur.

D oyuncusu `getCoinCoordinate()` fonksiyonunda önce sezgilime işlemini yapar ve daha sonra en kârlı olan altını hedef olarak belirler. Hedef belirleme maliyeti yirmi altın değerindedir. Her hamle için maliyeti beş altın değerindedir. D oyuncusu hamlesini yaptıktan sonra sıra A oyuncusuna geçer.

Eğer bir oyuncunun kasasındaki altın biterse oyuncu elenir ve artık sıra o oyuncuya geçmez. **Green** ve **Sonic** oyuncuları elenmiş.



Oyunumuzda 4 farklı değerde kazanç vardır. Bunlar 20 değerindeki elmas, 15 değerindeki altın, 10 değerindeki gümüş ve 5 değerindeki bronzdur. Gizli altınların normal altınlardan farkı sarı bir haklaya sahip olmaları.



## VI. ZORLANDIĞIMIZ NOKTALAR

Dünyanın mevcut durumu (COVID-19 krizi) yüz yüze takım olarak geliştirmemize engel oldu. Bundan dolayı başka yöntemlere başvurduk. **Google Meet** uygulamasını kullanmaya başladık ve bağlantı sıkıntılarının olmasına rağmen başarılı bir şekilde projemizi geliştirdik.

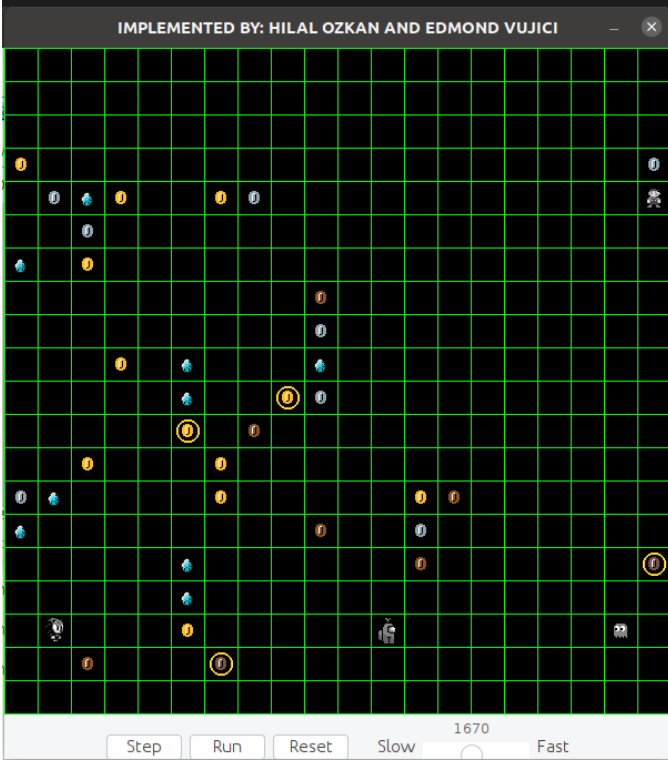
Oyunumuz da oyuncuları hareket ettirmek de zorlandık. **JGameGrid**'in `act()` metodunu doğru kullanmadığımız için oyuncularımızı istediğimiz şekilde hareket ettiremedik. `act()` metodu `gameGrid` tarafından her adımda tetiklenen bir metod olduğu için oyuncularımızın hareketleri fazla hızlıydı ve attığı adımları göremiyorduk. Yaptığımız araştırmalardan `setSlowDown()` metodunu kullanmamız gerektiğini öğrendik ve bu sorunu çözdük.

Görev dağılımı ise, her gün beraber **Google Meet** üzerinden konuştuğumuzdan dolayı eşittir.

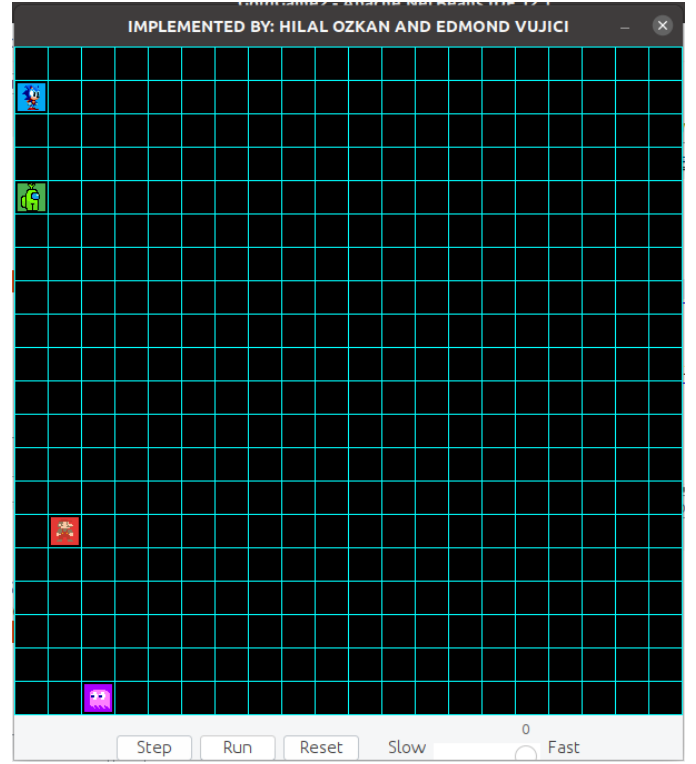
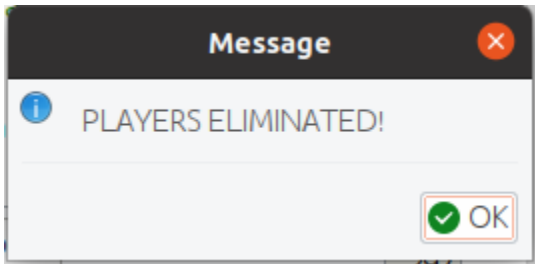
## VII. SONUÇ

Oyunun bitmesi için ya oyun tahtasında hiç altın kalmaması gerekir ya da oyundaki 4 oyuncunun da kasalarındaki altınların bitmesi gerekir.

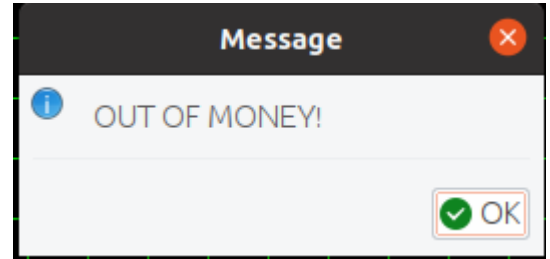
Her oyuncunun başlangıçta kasasında eşit miktarda ve kullanıcı isteğine göre değiştirebilen bir miktar altın vardır. Altını biten oyuncu oyundan elenir.



Eğer bütün oyuncuların altınları bitmiş ise oyun tahtasında alınacak altınlar olsa bile oyun sona biter.





Eğer oyun tahtasında görünür veya görünmez hiç altın kalmamış ise oyun biter.



Oyun sona bittiğinde **ifGameOver()** metodu sayesinde bütün oyuncular durdurulur ve oyuncuların kasada kalan altın miktarları, kazandığı altın miktarları, harcadığı altın miktarları, attığı adım sayısı ve gittiği yollar **ifGameOver()** metodu içerisinde oyuncuların sınıflarından çekilir.

Çekilen veriler **ResultScreen** sınıfından bir nesne oluşturularak **result** penceresinde kullanıcıya gösterilir. Bu sayede hangi oyuncunun ne kadar adım attığını, kasasında kalan altın miktarını, topladığı altın miktarını ve harcadığı altın miktarını öğrenebilirsiniz.

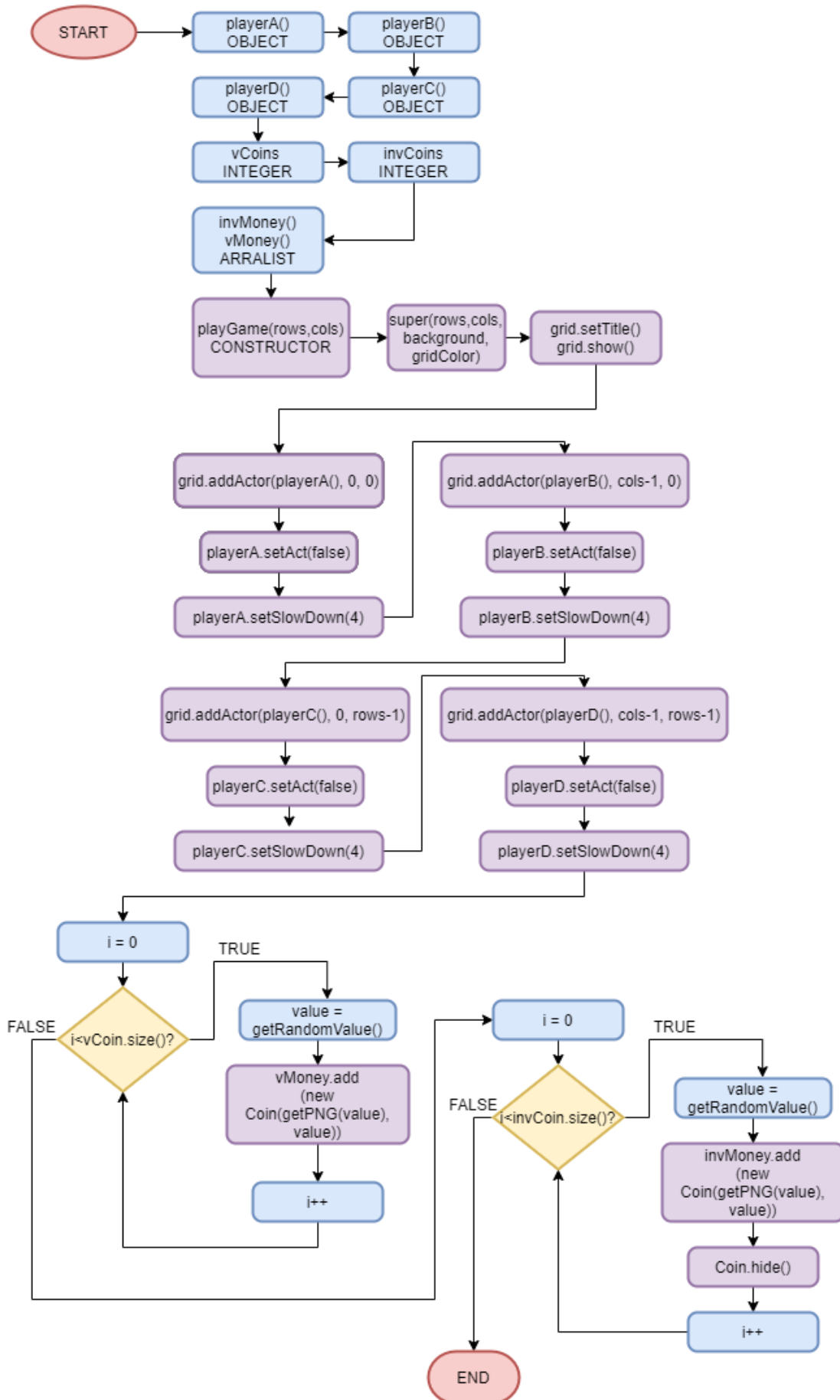
RESULT				
RESULT				
Player	Step	Cur...	Colle...	Spent
	96	140	205	265
	76	110	295	250
	71	0	320	250
	43	0	210	160
<button>WRITE TO FILE</button>				

Eğer oyuncularını hedeflerine ulaşmak için oyun boyunca izlediği yolların koordinatlarını görmek isterseniz **WRITE TO FILE** butonuna tıklayıp ve output.txt dosyasından gidilen yolları görebilirsiniz.

### VIII. KAZANIMLAR

- JGameGrid'in kullanımı ve uygulanmasını,
  - Kalıtımın özelliklerini kullanmayı,
  - JGameGrid'deki fonksiyonlarının üzerine yazmayı,
  - JSwing kullanmayı,
  - 8-bit resim çizmeyi ve düzeltmeyi,
  - Sözde kodu daha anlaşılır şekilde yazmayı,
  - Başka bir sınıftan başka sınıfa parametre import etmeyi,
  - Farklı kütüphaneleri deneyip en uygun olanı seçmeyi,
- başarılı bir şekilde öğrenmiş bulunmaktayız.

# OYUN OYNAMAMA SINIFI





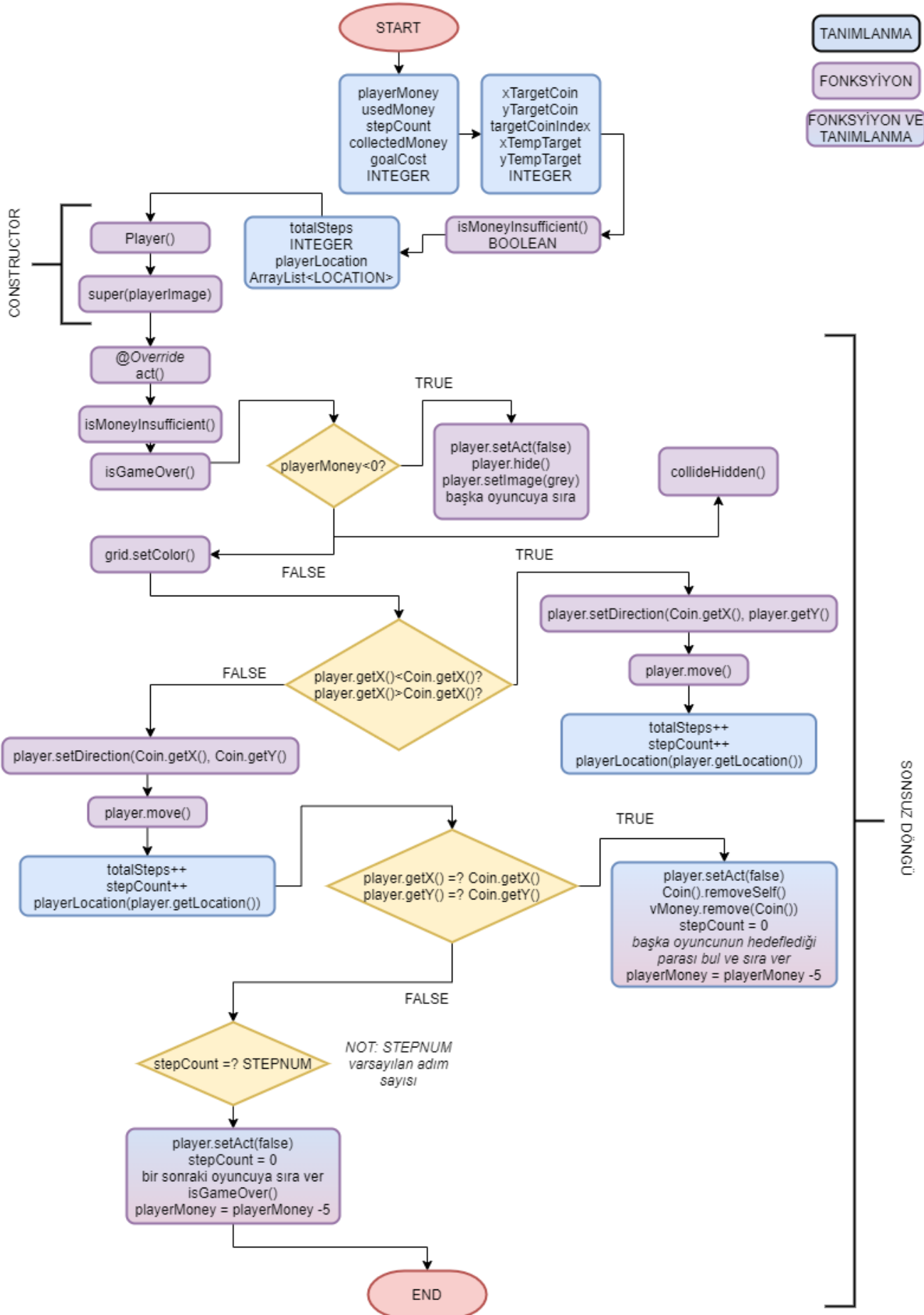
# OYUNCU SINIFI

LEJANT

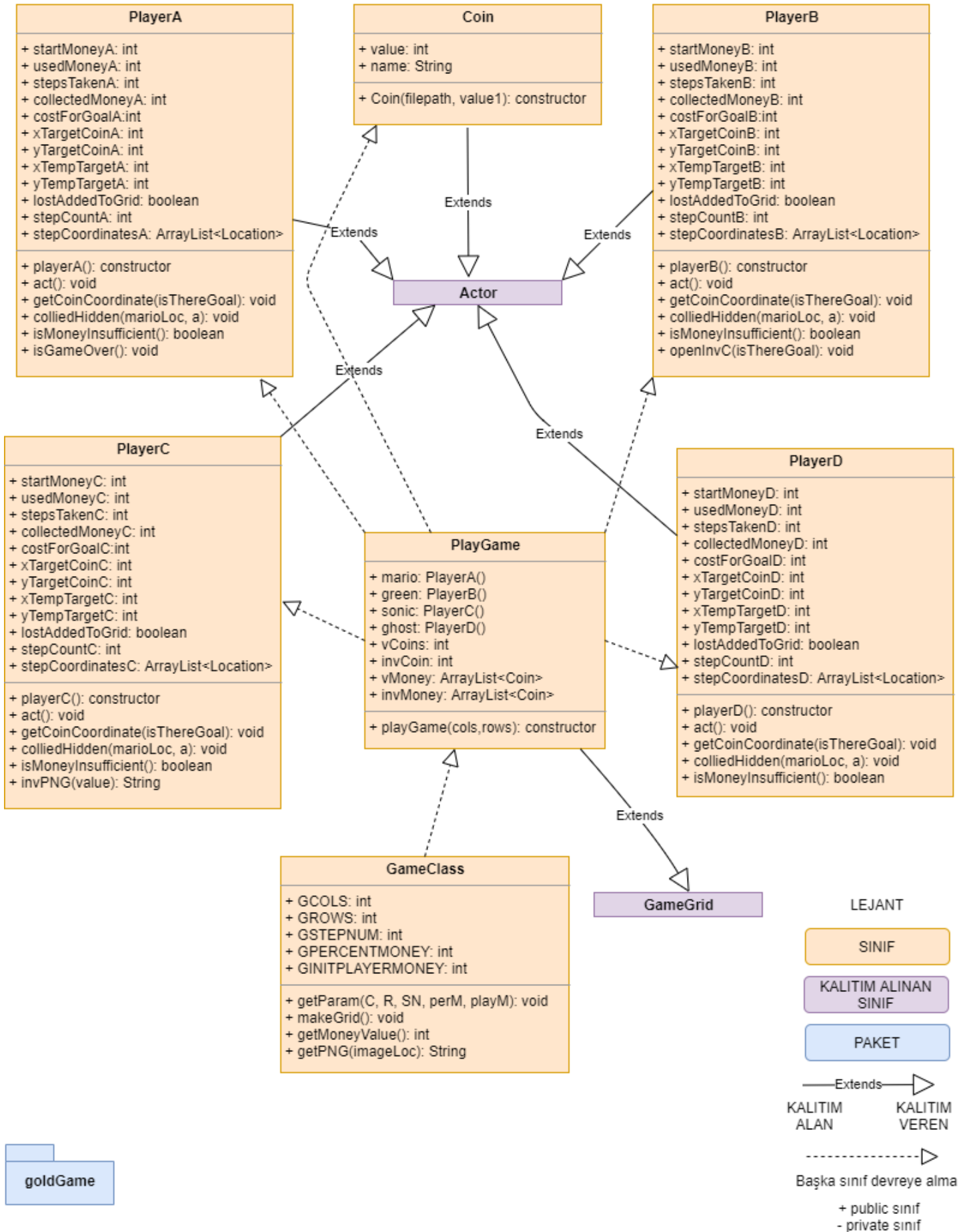
TANIMLANMA

FONKSİYON

FONKSİYON VE  
TANIMLANMA



# UML DİYAGRAMI



## IX. KAYNAKÇA

1. <http://www.aplu.ch/home/apluhomex.jsp?site=46>

(Eriřim Tarihi: 02-11-2020)

2. <http://www.aplu.ch/home/apluhomex.jsp?site=47>

(Eriřim Tarihi: 05-11-2020)

3. <http://www.aplu.ch/home/apluhomex.jsp?site=48>

(Eriřim Tarihi: 09-11-2020)

4. <http://www.aplu.ch/classdoc/jgamegrid/index.html>

(Eriřim Tarihi: 03-11-2020)

5. <https://www.javatpoint.com/java-jframe>

(Eriřim Tarihi: 19-11-2020)

6. [https://www.w3schools.com/java/java\\_arraylist.asp](https://www.w3schools.com/java/java_arraylist.asp)

(Eriřim Tarihi: 07-11-2020)