Savaş Kart Oyunu Raporu

Ceyda YURDASUCU
Kocaeli Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği
230201085

Sümeyye Nur ALTUN
Kocaeli Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği
230201032

1.Özet

Bu rapor, ’Savas Kart Oyunu’ isimli Programlama Laboratuvarı 1 dersinin 1. projesini açıklamak ve sunumunu gercekleştirmek amacıyla oluşturulmuştur. Raporda projenin özeti, giriş kısmı, yöntemler, deneysel sonuçlar,sonuç,yazar katkıları ve kaynakça bulunmaktadır. Proje yapımında Java programlama dili, Eclipse ide'si ve Java'nın swing paketi kullandık. Projeye dahil edilmesi gereken sınıflar import işlemi ile dahil edilmiştir.

2.Giriş

Bu projede yapılması gereken 4 adet problemimiz var. 1. problemimiz Java'nın nesne yönelimli programlama mantığını kullanarak sınıf yapılarını oluşturmak. 2. problemimiz oyun mekaniğini oluşturmak. Savaş mekaniğinde oyunculara 6 kart dağıtılması ve bu 6 karttan her hamlede 3 kart seçmemiz gerekiyor. Daha önce kullanmadığımız kartları kullanmak öncelikli olmalı ve her adımdan sonra oyuna yeni kart dahil olmalı. Tarafların birbirine karşı olan vuruş güçlerine göre dayanıklılıkları azalmaktadır. Oyun mekaniğini hazırlarken bunlara dikkat ederek hazırlamak gerekmektedir. 3. problemimiz oyunu görselleştirmek. Oyun görselleştirmesinde kullanıcı kartlarını seçerken bilgisayarın elindeki kartları görmemeli, kart seçimi bittikten sonra bilgisayarın kartları da ara yüzde görülebilmelidir. 4. problemde kartların yaptığı hamlelerin bir dosyaya yazılmasıdır.

3.Yöntem

A. 1. problem için çözüm yöntemi

Oluşturulan sınıflar hiyerarşik bir yapıdadır.Öncelikle savasAraci sınıfı oluşturulmuştur.Bu sınıfın temel özelliği oyundaki tüm savaş kartlarının temel özelliklerini

ve davranışlarını tanımlayan soyut (abstract) bir sınıftır. Bu sınıf, tüm kart tiplerinin uyması gereken temel şablonu oluşturur.bu sınıfta normal özellikler ve soyut özellikler tutulur.Soyut özellikler alt sınıflar tarafından tanımlanması gereken özelliklerdir.Başlangıç seviye puanı özelliği için yapıcı metod ,seviye puanını almak ve ayarlamak için getter() ve setter() metodları kullanılmıştır.Daha sonra savasAraci sınıfından türetilen Deniz,Hava,Kara sınıfları oluşturulmuştur.Her sınıfta araç türlerinin önce özellikleri tanımlanmış daha sonra yapıcı metod ile nesne özelliklerine başlangıç değerleri atanmıştır.Getter() ve setter() metodları ile bu nesne özelliklerinin alınması ve değiştirilmesi sağlanmıştır.Daha sonra hava sınıfından türeyen siha,ucak;kara sınıfından türeyen obus ve kfs;deniz sınıfından türeyen firkateyn ve sida alt sıfları oluşturulmuştur.Her alt sınıfta üst sınıfının yapıcı metodu super() metodu ile çağrılır ve nesnelerin doğru bir şekilde başlatılması sağlanır.Yapıcı metod içinde dayanıklılık,sinif,vurus ve vurus avantaj özellikleri başlatılır.Getter() metodu ile bu özellikler ayrı ayrı alınır .Daha sonra kartPuaniGoster() metodu ile savasa araclarının özelliklerinin detaylıca gösterilmesi sağlanır son olarak durumGuncelle() metodu ile saldiriDegeri ve kazanılanSeviyePuani değerleri parametre alınarak kartların dayanıklılık ve seviye puanları set() edilirek güncellenir.Her altsınıf için aynı yaklaşım kullanılmış ve altsınıfların oluşturulması tamamlanmıştır.Bu sınıfları oluştururken kullanılan yaklaşımlar şunlardır:

3.2A YAKLAŞIMLAR

Abstraction(Soyutlama) Soyutlama, nesneyi bazı karakteristik özellikleri olan ve bazı fonksiyonları gerçekleştirebilen bir veri tipi olarak genelleştirmektir. Class yapısı ile gerçekleştirilir. Soyut sınıflardan nesne oluşturulamaz. Soyut

yönteme sahip bir sınıfın kendisi de otomatik olarak soyuttur. Static,final,private olarak tanımlı yöntemler soyut olarak tanımlanamazlar.

Encapsulation(Paketleme) Kapsülleme, bir nesnenin iç yapısını, özelliklerini bozulmalara ve hatalara karşı korumaktır. Bununla birlikte başka sınıflar tarafından korunan özelliklere erişilmemesi demektir. Bu işlem erişim belirleyiciler sayesinde gerçekleşmektedir. Bir sınıfın özelliklerini dışardan erişime kapatabilmek için sınıf private olarak tanımlanır. Tanımlanan bu sınıfın özelliklerine doğrudan ulaşmak mümkün değildir. Erişebilmek için getter ve setter metotlarını ve Constructorlar kullanılır.

Inheritance(Kalıtım alma) Bir sınıfın kendisine ait özelliklerini ve methodlarını bir başka sınıfa aynen aktarması ya da diğer sınıflarda kullanılması kalıtım almadır. Üst sınıf özelliklerini alt sınıflarda kullanmak için extends kullanılır. Bir alt sınıf üst sınıfa erişmek istediğinde super() kullanmalıdır. Super(), üst sınıftaki öğelere erişmek ve üst sınıfa ait nesne yaratmak için kullanılır.

Polymorphism(Çok biçimlilik) Farklı şekilde çalışan nesnelere aynı şekilde erişmek şeklinde açıklanabilir.

Projemizde bu metodları kullandığımız yerler şunlardır:

ABSTRACTION KULLANIMI

SavasAraci Sınıfında:Abstract sınıf olarak tanımlanmıştır
getDayaniklilik(), getVurus(), getAltSinif() gibi soyut metodlar kullanılmıştır
Temel kart özelliklerinin soyutlanması sağlanmıştır

Kategori Sınıflarında.:
Hava, Kara ve Deniz sınıfları abstract olarak oluşturulmuştur
Her kategoriye özel avantaj sistemleri soyutlanmıştır.

II. ENCAPSULATION KULLANIMI

Alt araç Sınıflarında:

altSinif ve ...VurusAvantaji özellikleri private olarak tanımlanmıştır

Bu özelliklere erişim getter metodlarıyla sağlanmıştır

Kara,Hava ve Deniz Sınıfında:

dayaniklilik, vurus ve ...VurusAvantaji özellikleri private yapılmıştır

Kontrollü erişim için getter/setter metodları kullanılmıştır

Veri bütünlüğü korunmuştur

II. INHERITANCE (KALITIM) KULLANIMI

Temel Kalıtım Yapısında:

Altaraç sınıfları sınıfı Deniz,Hava ve Kara sınıflarından türetilmiştir
Her alt sınıf üst sınıfın özelliklerini miras almıştır

Constructor Kullanımında:
super() metodu ile üst sınıf constructor'ları çağrılmıştır.
Kalıtım zinciri boyunca özellikler aktarılmıştır
Hiyerarşik yapı kurulmuştur

IV. POLYMORPHISM KULLANIMI

Metod Ezme (Override) İle:kartPuaniGoster() metodu her kartta özelleştirilmiştir

durumGuncelle() metodu farklı davranışlar sergilemektedir

getAltSinif() her kartta kendi tipini döndürmektedir

Avantaj Sisteminde:

Vuruş avantajları her kart tipinde farklı hesaplanmaktadır

Aynı metod isimleri farklı işlevler görmektedir

Bu sayede dinamik davranış sağlanmıştır.

2. Problem İçin Çözüm Yöntemi

İlk olarak Oyuncu sınıfından yardım alarak insan ve bilgisayar adında nesneler oluşturduk. Bu oyunculara ait kullanıcı adı , seviye puanı gibi değerleri oyuncudan isteyerek bu istenen değerleri insan ve bilgisayar nesnelerinin parametrelerine atadık. Daha sonra

-kartlariDagit() metodu

Bu metotta oyunculara kartları dağıttık. kartlariDagit adli metotta Ucak, Obus, Firkateyn, Sida, Siha, KFS adlı sınıflara ait nesneler oluşturuldu. Kartların kendilerine ait seviye puanları olmalı ve bu seviye puanları oyun başında girilen seviye puanlarına eşit oluyor. kartlariDagit metodunda oluşturduğumuz kart nesnelerine bu seviye puanlarını atadık. Kart dağıtım işleminden önce boolean tipli gelistirilmisKartlarAcik adlı değişken ile oyuncuların skorları kontrol ediliyor. Eğer skorları 20'den büyük ise KFS, Sida, Siha gibi kartlar dağıtılınabiliyor.Bu adımdan sonra kart seçim işlemi geliyor.

-kartSec() metodu

Kart seçim işlemi bilgisayar için Oyuncu sınıfında kartSec isimli metotta Random sınıfı import edilerek random nesnesi oluşturulup rastgele olması sağlanıyor. İnsanın kartlarını seçmesi kartSecimDialogGoster metodunda oyunun ara yüzünde gerçekleşiyor. kartSec() metodu insanın kart seçimlerini yapması için kartSecimDialogGoster metoduna yönlendiriyor.

-kartSecimDialogGoster()

kullanilmamisKartSayisi isimli değişkenle bu kartların sayısı for döngüsü yardımıyla bulunuyor. Bu değişken ile eğer kullanılmamamış yeterli sayıda kart kalmazsa o kartlar tekrar seçilebilir haline getiriliyor ve aynı işlem oyun bitene kadar devam ediyor. Bir sonraki işlem kartların karşılaştırılması işlemidir. Bu işlem karsilastir metodunda yapılmaktadır.

-karsilastir() metodu

karsilatir metodu parametre olarak SavasAraci sınıfından oluşturulmuş insanKartini ve bilgisayarKartini almaktadır. Bu metotta ilk olarak oyuncuların birbirine karşı avantaj puanı var mı kontrol ediliyor. Bu kontrol instanceof operatörü ile sağlanmaktadır. Kartın bir avantaj puanı olur ise avantajPuani adlı değişkene atanmaktadır. Daha sonra saldiriHesapla() metodu ile oyuncuların toplam vuruş gücü bulunur ve bilgisayarVurus ve insanVurus adlı değişkenlere atanır. saldiriHesapla() metodu parametre olarak kartın vuruşunu ve avantajPuani'ni almaktadır. Bu adımdan sonra kartların kalan dayanıklıkları için insanKartDayaniklilik ve bilgisayarKartDayaniklilik adında iki değişken tanımlanmaktadır. Kalan dayanıklılığı bulmak için getter metoduyla kartın dayanıklılığı çağrılır ve dayanıklılıktan karşı tarafın vuruş gücü(rakip taraf insan ise insanVurus bilgisayar ise bilgisayarVurus) çıkartılır. Bulunan sonuçlar insanKartDayaniklilik ve bilgisayarKartDayaniklilik'a atanır. Bu adımdan sonra kartların dayanıklılıklarına göre işlem yapılacaktır. Eğer bilgisayarKartDayaniklilik 0'dan küçükse bilgisayarın kartı elenir ve insanın kartının seviye puanı insanKarti.setSeviyePuani metoduyla değiştirilir. İnsanın skoru da elenen kartın seviye puanı kadar artar ve insan.setSkor ile güncellenir. Eğer insanKartDayaniklilik 0'dan küçükse insanın kartı elenir ve bilgisayarın kartının seviye puanı bilgisayarKarti.setSeviyePuani metoduyla değiştirilir. Bilgisayarın skoru da elenen kartın seviye puanı kadar artar ve bilgisayar.setSkor ile güncellenir. Hem insanKartDayaniklilik hem de bilgisayarKartDayaniklilik 0'ın altına düşerse iki kart da elenir. İki oyuncunun da skorunda elenen kartların seviyesi kadar artış olur.

-hamleyap() metodu

Bu metotta olacak hamle sayısına göre işlemler yapılır. Önce kart seçimi yapacak oyuncunun 3 kart seçip seçmediği kontrol edilir ve hamle başında yeni eklenen kartı seçme zorunluluğu getirilir. Hamleler sayılır ve her hamlede seçilen kartlar işlenir. İşlenen kartlar karsilastir() metodu çağırılarak karşılaştırılır. Hamle sayisi suankiHamle değişkeni ile sayılır. Eğer suankiHamle oyuncudan istenen hamleSayisi'na eşit olursa oyun sonlanır.

-oyunSonucuGoster()

Bu metotta oyun sonlandığında kontroller yapılır ve kazanan oyuncunun adı ekrana yazılır. İlk olarak oyuncuların elinde olan kartlar tek tek for döngüsü yardımıyla dolaşılır ve ellerinde olan kart sayıları bilgisayarKartSayisi ve insanKartSayisi adlıdeğişkenlere atanır. Yine aynı döngü yardımıyla ellerinde olan kartların toplam dayanıklılığı hesaplanır ve hesaplama sonucu insanToplamDayaniklilik ve bilgisayarToplamDayaniklilik adlı değişkenlere

atanır. Kazanan oyuncunun belirlenmesi işlemi için kontroller yapılır. Oyunun sonlanması ya herhangi bir oyuncunun kartlarının tükenmesi ile ya da oyun başında belirlenen hamle sayısına ulaşması ile olur. if else kontrolleri ile bilgisayarKartSayisi 0'a eşitse kazanan değişkeni insan nesnesinin adına , insanKartSayisi 0'a eşitse kazanan değişkeni bilgisayara eşit olur. Eğer iki oyuncunun da kartları tükenmediyse ve oyun , oyun başında belirtilen hamle sayısına ulaşılarak bittiyse oyuncuların skorları karşılaştırılır.Skor karşılaştırılırması insan.getSkor ve bilgisayar.getSkor metodları çağırılarak yapılır. Skorlar da eşit olursa oyuncuların insanToplamDayaniklilik ve bilgisayarToplamDayaniklilik adlı değişkenleri karşılaştırılır. Hangi dayanıklılık değişkeni daha fazlaysa kazanan o oyuncu olur. Eğer onlarda eşit olursa kazanan değişkenine "Berabere" kelimesi atanır. Ekrana oyun bittiğinde ulaştığımız bilgiler yazılır.

C. 3. problem için Çözüm Önerisi

Bu problem oyunun görselleştirilmesi problemidir. Oyunun görselleştirmesini yaparken Java'nın Swing paketini kullandık. Görselleştirmeyi yaparken yazdığımız kodlarda import edilmesi gereken Swing paketinden sınıflar var ise bunları import ettik.

-Ana ekranın açılması

Ana ekranın açılması için Jframe sınıfını kullandık. Jframe sınıfından oyunAnaEKrani adında değişken tanımladık. oyunAnaEkrani.setSize(1200,800) komutu ile oyun ekranını 1200*800 pixel boyutunda ayarladık.

-Oyun Alanı(Paneli)

Jpanel sınıfından yeni bir nesne oluşturduk ve GridLayout(2, 1, 10, 10) ile 2 satır ve 1 tane de sütun olacak şekilde oyun alanını ızgaralara ayırdık. Arka plan rengini kartPanel.setBackground() ile yeşil rengi yaptık.

-Bilgi Paneli

Yeni bir panel oluşturulur. setBackground(new Color()) ile panelin rengi belirlenir.

-Bilgi Alanı

Yazıların yazılacağı bilgiAlani adında bir JTextArea nesnesi oluşturduk. bilgiAlani.setBackground(new Color()) ile arka planı sarı olarak ayarladık. bilgiAlani.setFont(new Font()) ile yazıların fontunu ayarladık. JScrollPane sınıfından bir nesne oluşturarak bilgiAlanını kaydırmaklı ekran şeklinde yaptık.

-Hamle Yap butonu

Oyunda seçtiğimiz kartların hamle yapması için JButton sınıfından hamleButton isimli buton oluşturduk. hamleButton.setFont(new Font()), hamleButton.setBackground(new Color()), hamleButton.setForeground() gibi metotlar ile hamle butonunun özelliklerini belirledik.

-Yeni Kart Paneli, Skor Paneli

Jpanel sınıfından yeni bir nesne oluşturduk. setLayout(new FlowLayout(FlowLayout.CENTER, 10, 5)) ile içeriklerin ortalanması sağlanır. oyunAnaEkrani.add(yeniKartPanel, BorderLayout.NORTH) ile ana pencerenin üst kısmına yeniKartPanel eklenmiş oldu.

-oyunAnaEkrani.setVisible(true)

Bu sayade oyun ekranı kapatılmadığı sürece açık kalır.

-JButton grupKartButtonOlustur() metodu

Bu metod bir kart çeşidini alarak bunu bir button olarak oluşturur ve aynı tipteki kartları gruplar. JButton nesnesi oluşturulur ve boyutu 100x150 piksel olarak ayarlanır. Bu butonlara kartResimleri() metodu çağrılarak kartların resimleri eklenir. StringBuilder sınıfından bir nesne oluşturularak kart hakkında bilgiler(kart çeşidi, kart adeti, dayanıklıkları vb.) verilir. Butona tıklama olayı (kartSecimDialogGoster) eklenir. infoPanel adında JPanel nesnesi oluşturulur. infoPanel için BoxLayout.Y_AXIS paremetresi ile dikey bileşenler ayarlanır. Arka planı da yarı saydam bir renk olarak ayarlanır. Kart sayısını gösteren sayiLabel oluşturulur. Oluşturulan buton return ile geri döndürülür.

-kartSecimDialogGoster() metodu

Bu metot insan nesnesinin kart seçimini yapabilmesi için görsel oluşturur. Parametre olarak oyuncunun seçeceği kartları ve kart tipini alır. Yeni bir dialog penceresi oluşturulur ve FlowLayout düzeni kullanılarak kartların yan yana dizilmesi sağlanır. Daha sonra her kart için bir buton oluşturulur ve kartın görseli ve boyutları ayarlanır.StringBuilder sınıfından cardInfo ismli bir nesne oluşturulur. Bu nesne her kartınbilgilerini gösterir.Kartlar kullanılmış ise kartButton.setBorder(BorderFactory.createLineBorder(Color.RED, 3)) ile kenarları kırmızı işaretlenir. Daha sonra kartın o hamlede seçili olup olmadığı kontrol edilir ve eğer seçili ise kartButton.setBorder(BorderFactory.createLineBorder(Color.GREEN, 3)) ile kenarları yeşil işaretlenir.

-JButton kartButtonOlustur() metodu

Bu metot yeni bir buton oluşturur ve 100*150 boyutunda kart boyutu ayarlar. kartButton.setIcon(kartResimleri(kart)) ile kartın görseli butona eklenir. Kart tooltip bilgisi oluşturularak kart üstüne gelindiğinde kartın bilgileri verilir. Kartın kullanılıp kullanılmadığı kontrolü if ile yapılır ve eğer kart kullanıldıysa kart üstüne gelindiğinde kırmızı renk ile 'Daha önce kullanıldı' bilgisi verilir. Eğer kart yeni ise mavi renk ile 'yeni kart' bilgisi verilir.

-kartlariGuncelle() metodu

Bu metot oyunların kartlarını güncelleyip, ara yüzde görünmesini sağlar. kartPanel.removeAll() ile kartların hepsini siler. jPanel sınıfından yeni bir oyuncuPanel nesnesi oluşturulur. Kartların tekrar görselleşmesi için daha önce açıkladığımız metodlardan kartlariGrupla() ve grupKartButton() metodu kullanılır. Yeni oluşturulan kartlar panele eklenir.

-ImageIcon kartResimleri() metodu

Bu metot önce bir String dizisi oluşturur ve bu diziye '.png' ve '.webp' uzantıları atılır. Bu uzantıların atılma sebebi yüklenecek olan görsellerin uzantılarının .png veya .webp olmasıdır. Uzantıları kontrol edilerek resimler kartlar için oluşturulan butonlara yüklenecektir. Eğer dosya bulunamazsa oyuncuların kartlarına varsayılan bir görsel oluşturulacaktır.

-ImageIcon getKapaliKart() metodu

Bu metot bilgisayarın kartlarını insan oyuncusu göremeyeceği için kartların resimleri gözükmez ve bordo renkte varsayılan görsel oluşturulur.

-karsilastirmaEkraniGoster() metodu

Oyuncu 3 kartını seçtikten sonra hamle yapılır ve bu metot hamle yapıldıktan sonra bilgisayar ve insan oyuncularının kartlarının karşılaştırmasının ara yüzünü oluşturur. Bunun için anaPanel adında Jpanel sınıfından bir panel oluşturulur. İnsan oyuncusunun ve bilgisayarın kartlarını karşılaştırmak için bir for döngüsü başlatılır. İnsan ve bilgisayar kartları için paneller oluşturulur, kartların görselleri yerleştirilir ve kartların arasına 'VS' labeli konulur. Karşılaştırma işlemi bittikten sonra hamle yapmaya devam etmek için JButton sınıfından 'Devam Et' butonu oluşturulur.

-oyunSonucunuGoster()

Bu metodu daha önce açıklamıştık, şimdi ise bu metottaki görselleştirilme kısmını açıklayacağız. Bu metodun görselleştirme kısmında oyunun sonunda sonuçları görüntülemek için bir JDialog penceresi oluşturuluyor. Bu pencereye kazanan taraf ya da berabere bittiyse 'berabere bitti' bilgisi, oyuncuların skorları, kalan kartları yazılır. Jbutton sınıfından bitirButton ve yenidenOynaButton nesneleri oluşturulur. Bu butonlar panelin en altına yerleştilir ve eğer oyun bitirilmek isteniyorsa bitirilir, tekrar oynanmak isterse tekrar oynanır.

C. 4. Problem İçin Çözüm Önerisi

Oyun bilgilerinin dosyaya yazılması istenmektedir. Bunun için PrinterWriter sınıfında dosyaYazma adında bir nesne oluşturduk. oyun_kayit.txt adında bir dosya oluşturduk. flush() ile bilgiler hemen dosyaya yazılır.

Kaynakça

- https://youtu.be/RPe4fil45Mo?si=HiaLVD2s7sbt-vFi (kodlama vakti youtube kanalı java kursu playlisti)

- https://youtu.be/zIitDyuUrmQ?si=LaAumKu0GrOTmust (Ufuk Çelik youtube kanalı adım adım java playlisti)

-https://muhammedtalhacevik.medium.com/java-swing-ve-gui-programlama-kullan%C4%B1c%C4%B1-aray%C3%BCzleri-olu%C5%9Fturman%C4%B1n-temel-rehberi-by-muhammed-talha-7ec9fd9992cb

Katkılar

Projenin tüm kısımlarını ekip çalışması halinde yürüttük. Projede yaptığımız her adımda birbirimizi bilgilendirdik. Yaptığımız kısımlarda eğer hatalarla karşılaşırsak bu hataları çözmekte birbirimize yardım ettik.

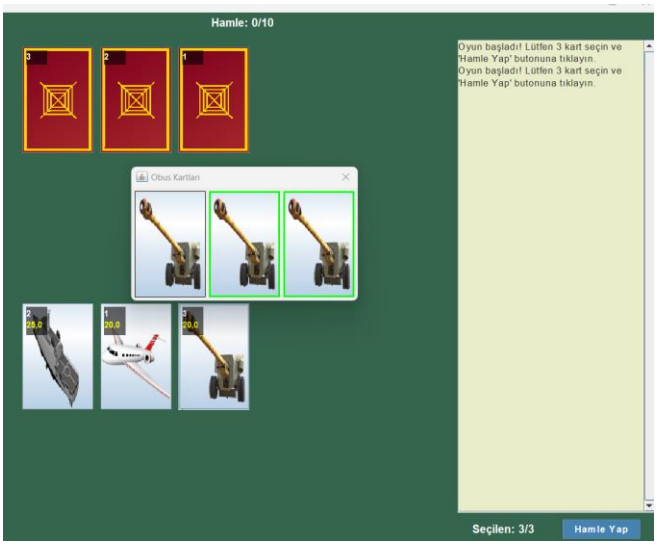
Sümeyye Nur Altun katkıları

Oyundaki sınıflarını oluşturmak için gerekli araştırmaları yaparak bu sınıfları oluşturmaya katkı sağladı. Oyun mekaniğinin mantığını kurmaya katkı sağlayarak bunları koda dökmeye katkı sağladı. Görselleştirme için gerekli olan araştırmaları yaptıktan sonra görsel için kullanılan metotları oyun sınıfı içinde başarılı bir şekilde implementasyonunu gerçekleştirdi. Oyundaki hataları görmek için test sürecini gerçekleştirdi.

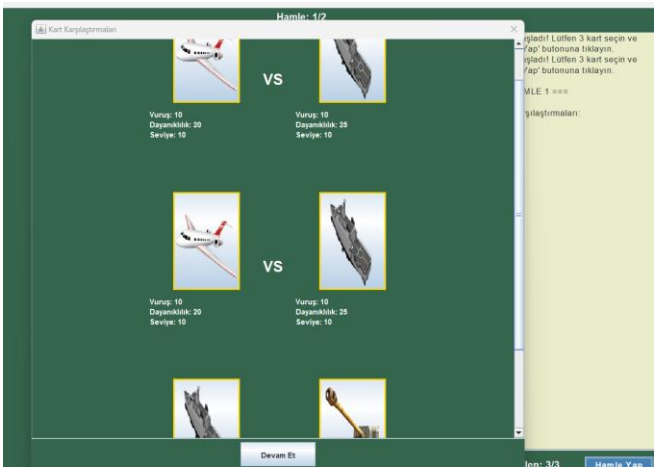
Ceyda Yurdasucu katkıları

Oyun için gerekli sınıfları oluşturmada rol aldı.Oyun mekaniği için gerekli metodların ve özelliklerin oluşturulmasında katkı sağladı.Görselleştirme için gerekli olan araştırmaları yaptıktan sonra görsel için kullanılan metodları oyun sınıfı içinde başarılı bir şekilde implementasyonunu gerçekleştirdi.Oyun çalıştırıldıktan sonraki test sürecinde hata tespiti yaparak oyunun geliştirilmesinde ve doğru çalışmasında önemli bir rol oynadı.

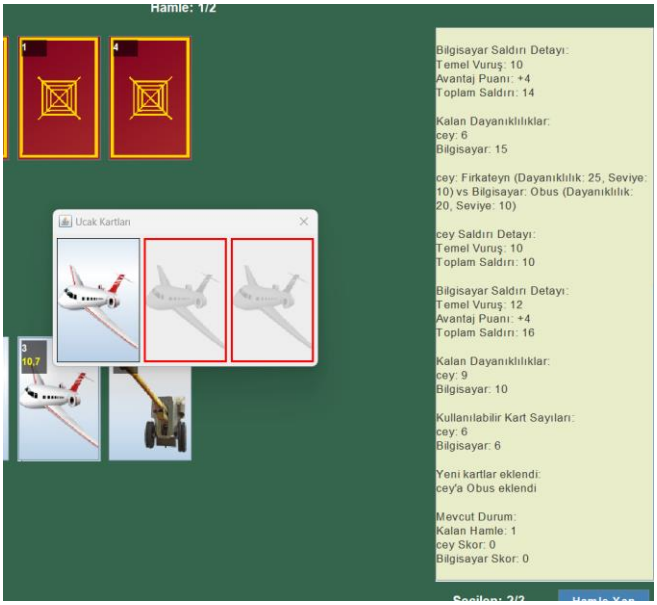
SONUÇLAR



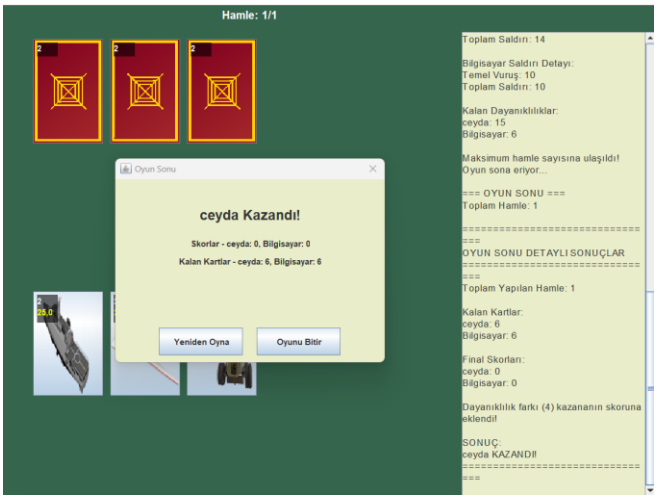
Burada kart seçim örneği gösterilmiştir.Kartlar gruplanıyor ve üzerine tıklandığında adedi kadar kart açılıyor.Seçilen kartlar yeşil olarak çerçeveleniyor ve seçimde karmaşıklık bu sayede gideriliyor.Seçilen kartın üzerine tekrar tıklandığında bu kartın bırakılması sağlanıyor ve seçilen kart sayısı azaltılıyor.



Burada kart karşılaştırmaları örneklendirilmiştir.Her seçilen 3 kart sonrası karşılıklı olarak seçim sırasına uygun olacak şekilde kartların karşılaştırılması seviye,vuruş,dayanıklılık özellikleriyle birlikte gösteriliyor.Devam et butonu ile karşılaştırma ekranı kapatılıyor ve kapatıldıktan sonra karşılaştırma sonucu bilgileri yan taraftaki bilgi paneline ekleniyor.



Burada önceki tur seçilen kartların belirtilmesi kırmızı çerçeve ile yapılması ve bilgi panelinde önceki hamlenin sonuçlarının detaylarının işlenişi gösterilmektedir.



Oyun sonu iki şekilde bitiyordu ilki belirlenen hamle sayısına ulaşılınca ikincisi ise bir tarafın elindeki kart sayısı 1 olduğundaydı.Eklenen görsel hamle sayısına ulaşıldığı durumu göstermektedir.Hamle sonunda berabere kalınmış ve dayanıklılık farkı ile kazanan belirlenmiştir.Oyun sonunda bizlere kazanan bilgisi ve oyunu bitir,yeniden oyna butonu ile seçenek sunulmuştur.Oyunu bitir butonu programı sonlandırır.Yeniden oyna butonu ile sıfırdan bir oyun başlar.

```
ceyda: Ucak (Dayanıklılık: 8, Seviye: 10) vs Bilgisayar: Firkateyn (Dayanıklılık: 15, Seviye: 10)

ceyda Saldırı Detayı:
Temel Vuruş: 10
Toplam Saldırı: 10

Bilgisayar Saldırı Detayı:
Temel Vuruş: 10
Avantaj Puanı: +4
Toplam Saldırı: 14

Kalan Dayanıklılıklar:
ceyda: -6
Bilgisayar: 5

ceyda'nın kartı elendi!
Bilgisayarın kartı 10 seviye kazandı! Yeni seviye: 20
Bilgisayarın skoru 10 arttı! Yeni skor: 20

Maksimum hamle sayısına ulaşıldı! Oyun sona eriyor...

=== OYUN SONU ===
Toplam Hamle: 4

================================
OYUN SONU DETAYLI SONUÇLAR
================================
Toplam Yapılan Hamle: 4

Kalan Kartlar:
ceyda: 7
Bilgisayar: 5

Final Skorları:
ceyda: 50
Bilgisayar: 20

SONUÇ:
ceyda KAZANDI!
================================
```

Burada başlatılan bir oyunun .txt uzantılı bir dosyaya detaylı şekilde hamlelerin, karşılaştırmaların ve sonuçlarının son olarakta final durumunun yazdırıldığı örnek senaryo üzerinden gösterilmektedir.

PROJE KABAKODU

• Başla
• Temel Sınıfları Oluştur
- SavasAraci (abstract) sınıfını oluştur (temel özellikler
- Alt sınıfları oluştur (Ucak, Obus, Firkateyn, Siha, KFS, Sida)
- Oyuncu sınıfını oluştur
• Oyun Başlangıç Ayarları
- Oyuncu adını al -
Maksimum hamle sayısını al
- Başlangıç seviye puanını al
- İnsan ve bilgisayar oyuncularını oluştur
- Kart havuzunu oluştur ve kartları dağıt
- Log dosyasını başlat
• GUI Bileşenlerini Oluştur
- Ana pencereyi oluştur
- Kart panelini oluştur
- Bilgi panelini oluştur
- Hamle sayacını oluştur
- Skor göstergelerini oluştur
- Yeni kart panelini oluştur
• Kart Dağıtım Sistemi
- Her oyuncuya 6'şar kart dağıt
- Kartları görsel olarak yerleştir
- Kart görsellerini cache'le

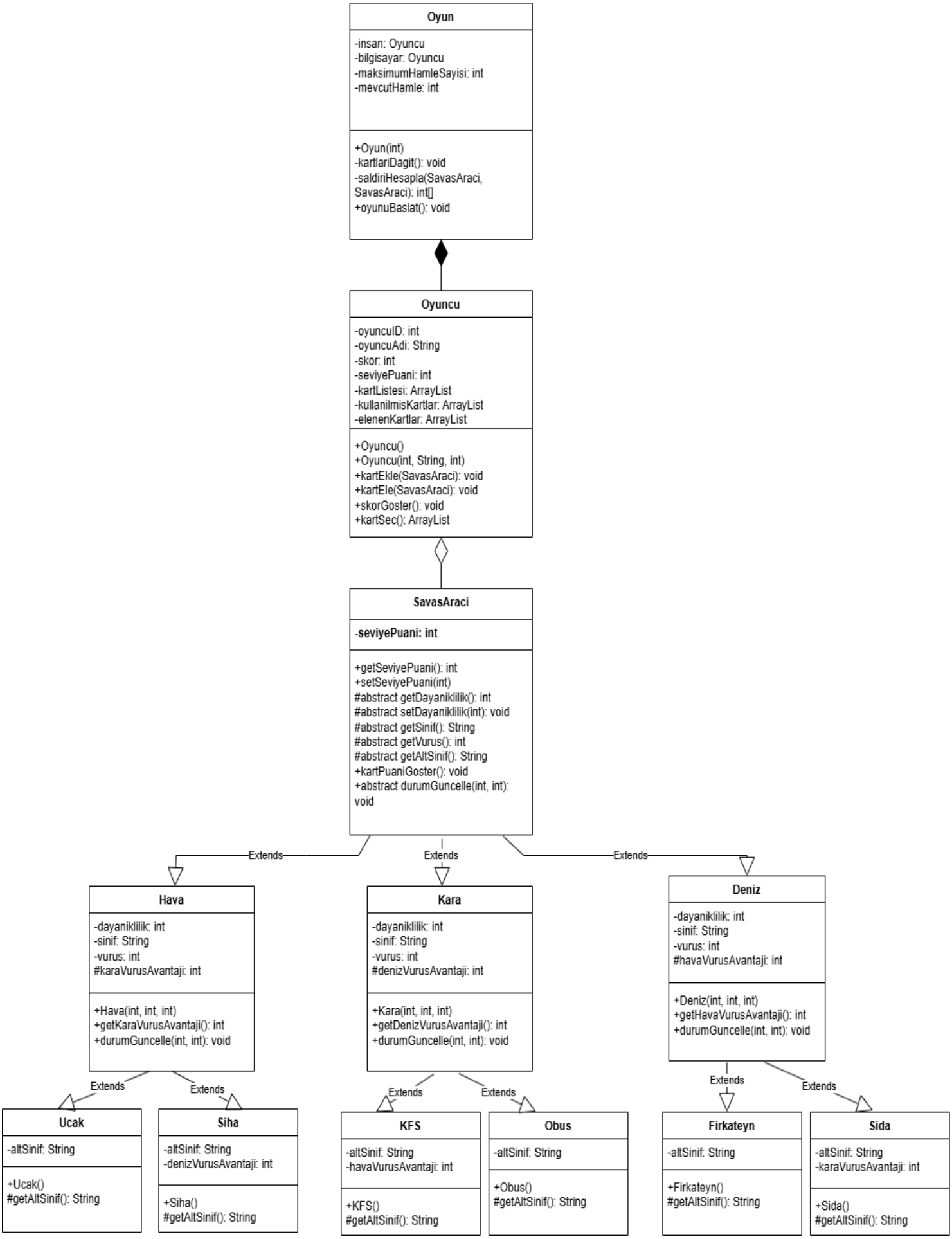
• Her Hamle İçin
- Kart seçim kontrolü yap (3 kart)
- Yeni kart seçim kontrolü yap -
Hamle sayısını güncelle -
Son hamle veya maksimum hamle kontrolü yap
EĞER son hamle veya maksimum hamle ise
- Karşılaştırma ekranını göster
- Sonuçları hesapla
- Oyunu bitir
DEĞİLSE -
Normal hamleye devam et
- Seçilen kartları işle
- İnsan kartlarını al
- Bilgisayar kartlarını seç
- Karşılaştırma ekranını göster
- Sonuçları hesapla
- Kart tükenmesi kontrolü yap
EĞER bir oyuncunun kartı tükeniyorsa
- Ekstra kartlar ekle
- Son hamle modunu aktifleştir
DEĞİLSE
- Normal oyuna devam et
- Yeni kart ekleme
EĞER son hamle değilse
- Her iki oyuncuya yeni kart ekle
- Yeni kart seçim zorunluluğunu aktifleştir
- Arayüzü güncelle
- Kartları güncelle
- Yeni kart panelini güncelle
- Bilgi panelini güncelle -
Hamle sayacını güncelle

- Skor kontrolü yap
EĞER skor >= 20 ise
- Gelişmiş kartları aktifleştir

• Log Sistemini aç
- Her hamleyi kaydet
- Karşılaştırma sonuçlarını kaydet
- Oyun sonucunu kaydet

• Oyun Sonu İşlemlerini tamamla
- Final skorlaranı hesapla
- Kazananı belirle
- Sonuç ekranını göster
- Log dosyasını kapat

• Bitir


Oyun
-insan: Oyuncu
-bilgisayar: Oyuncu
-maksimumHamleSayisi: int
-mevcutHamle: int
+Oyun(int)
-kartlariDagit(): void
-saldiriHesapla(SavasAraci, SavasAraci): int[]
+oyunuBaslat(): void
Oyuncu
-oyuncuID: int
-oyuncuAdi: String
-skor: int
-seviyePuani: int
-kartListesi: ArrayList
-kullanilmisKartlar: ArrayList
-elenenKartlar: ArrayList
+Oyuncu()
+Oyuncu(int, String, int)
+kartEkle(SavasAraci): void
+kartEle(SavasAraci): void
+skorGoster(): void
+kartSec(): ArrayList
SavasAraci
-seviyePuani: int
+getSeviyePuani(): int
+setSeviyePuani(int)
#abstract getDayaniklilik(): int
#abstract setDayaniklilik(int): void
#abstract getSinif(): String
#abstract getVurus(): int
#abstract getAltSinif(): String
+kartPuaniGoster(): void
+abstract durumGuncelle(int, int): void
Extends
Extends
Extends
Hava
-dayaniklilik: int
-sinif: String
-vurus: int
#karaVurusAvantaji: int
+Hava(int, int, int)
+getKaraVurusAvantaji(): int
+durumGuncelle(int, int): void
Kara
-dayaniklilik: int
-sinif: String
-vurus: int
#denizVurusAvantaji: int
+Kara(int, int, int)
+getDenizVurusAvantaji(): int
+durumGuncelle(int, int): void
Deniz
-dayaniklilik: int
-sinif: String
-vurus: int
#havaVurusAvantaji: int
+Deniz(int, int, int)
+getHavaVurusAvantaji(): int
+durumGuncelle(int, int): void
Extends
Extends
Extends
Extends
Extends
Extends
Ucak
-altSinif: String
+Ucak()
#getAltSinif(): String
Siha
-altSinif: String
-denizVurusAvantaji: int
+Siha()
#getAltSinif(): String
KFS
-altSinif: String
-havaVurusAvantaji: int
+KFS()
#getAltSinif(): String
Obus
-altSinif: String
+Obus()
#getAltSinif(): String
Firkateyn
-altSinif: String
+Firkateyn()
#getAltSinif(): String
Sida
-altSinif: String
-karaVurusAvantaji: int
+Sida()
#getAltSinif(): String