PROGRAMLAMA LABORATUVARI I
2. PROJE RAPORU

Bedirhan EREN
230202081

Omar ATTAL
230202001

1. ÖZET

Bu rapor Programlama Laboratuvarı I Dersinin **2. Projesini** açıklamak ve sunumunu gerçekleştirmek amacıyla oluşturulmuştur.

Proje nesneye yönelik programlama standartları çerçevesinde oluşturulmuştur.

Bu projenin backend bölümü **Java** dilinde **Visual Studio Code** ortamında geliştirilmiştir. Frontend kısmı ise **JavaFX** kütüphanesi kullanılarak geliştirilmiştir. Raporda projenin tanıtımı özet, giriş, yöntem, deneysel sonuçlar, sonuç, yazar katkıları ve kaynakça bölümlerinden oluşmaktadır.Proje aşamasında yararlanılan kaynaklar raporun son bölümünde bulunmaktadır.

2. Projenin Tanıtımı

Bu proje nesneye yönelik programlama kuralları çerçevesinde geliştirilmiştir. Projede kalıtım, polimorfizm, getter-setter metodları, constructor (yapıcı) metodlar oldukça sık kullanılmıştır. Projede bir kart oyunu simule edilmektedir. Kullanıcı interaktifliğinde bir oyundur ve kullanıcı kendi kartlarını istediği gibi seçebilir. Kullanıcıdan önce tur sayısı girmesi istenir. Oyun maksimum 5 tur sürebilmektedir. Uygun tur sayısını girdikten sonra oyun başlar. İki tarafa da rastgele 6 adet kart atılır. Elinde olan oyun başı rasgele 6 karttan her tur 3 kart seçerek karşısında onun rakibi olan bilgisayarın seçtiği kartlarla savaşır. Her turun sonucunda kartların dayanıklılığı azalır ve eğer kartın dayanıklılığı sıfırın altına düşerse kart destesinden çıkartılır. Kartların her biri üst sınıfından kalıtım almaktadır ve sınıfların kendileri arasında vuruş avantajları bulunmaktadır. Belli kart sınıfları belli kart sınıflarına daha fazla bonus hasar verir. Buna göre dayanıklılık hesaplaması yapılır. Her tur sonu iki tarafa da +1 kart eklenir. Oyun bittikten sonra gerekli hesaplamalar yapılıp kazanan taraf ekrana kazandı olarak gösterilir.

3. YÖNTEM

Bu projede bir interaktif kart oyunu simüle edilmektedir. Oyunda 2 taraf bulunmaktadır biri bilgisayar diğeri ise kullanıcı tarafıdır. Her iki tarafında kendine ait kartları vardır. Kullanıcı ve bilgisayar bu kartlardan her tur 3 adet seçerek kartlarını savaştırır. Her bir kartın kendine ait dayanıklılık, seviye puanı , vuruş avantajı gibi özellikleri mevcuttur. Proje nesneye yönelik programlama bilgisini geliştirmek için geliştirilmiştir ve buna göre yapılar kullanılmıştır.

İlk adım olarak class yapıları tanımlandı. Oyunda bulunan kartların tamamının kalıtım alması için SavasAraclari.java class dosyası oluşturuldu. Bu clas yapısının içine seviye puanı değişkeni tanımlandı. Ardından abstract fonksiyonlar tanımlandı. Bu fonksiyonlar Dayaniklilik, Vurus, Sinif değişkenlerini alt classlarda değiştirilmesi için abstract yapısı kullanıldı. SavasAraclari classına KartPuaniGoster fonksiyonu yazılarak ilgili kartın kart özelliklerine erişilmesi sağlandı. Ayrıca seviye puanını ayarlamak ve görmek için getter setter seviye puanı metodları kullanıldı. Daha sonra her biri SavasAraclari classindan kalıtım alan Hava, Kara, Deniz classları tanımlandı. Hava classında public abstract int karavurusAvantaji fonksiyonu tanımlandı böylece her bir hava kartının alt classlarda kendine özel kara vuruş avantajı değeri verilmesine olanak sağlandı. Ayrıca classın constructor (yapıcı metodu) da yazıldı ve seviye puanını super kullanımı ile SavasAraclari sınıfından aldı. Ve bu classda Ovveride edilerek DurumGuncelle ve KartPuaniGoster fonksiyonu kullanıldı. Override edildiği için super.KartPuaniGoster() kullanıldı. Sinif fonksiyonu da SavasAraclarindan override edildi ve "Hava" sınıfını döndürmesi sağlandı. Daha sonra Kara.java class yapısı tanımlandı. Kara kartlarının denizVurusAvantaji() fonksiyonu abstract olarak tanımlandı. Burada Havayla aynı şekilde KartPuaniGoster ve DurumGuncelle fonksiyonları override edildi. Ve Sınıf fonksiyonu da override edilerek "Kara" sınıfını döndürmesi sağlandı. Deniz.java classında ise havaVurusAvantaji fonksiyonu abstract olarak tanımlandı. DurumGuncelle, KartPuaniGoster ve Sinif fonksiyonları Override edildi.

Ardından her biri Savaş araçları sınıfımıza bağlı olan bu 3 class için (Hava, Kara, Deniz) özel kartlar tanımlandı. Bu kartlar Hava class'ı için : Ucak ve Siha , Kara classı için : KFS (Kara Füze Sistemi) ve Obüs, Deniz classı için Firkateyn ve Sida

Hava.java classının alt classları (uçak ve siha) için açıklamaya başlayalım. **Ucak.java** oluşturuldu. Bu classın extends ifadesi ile Hava classından kalıtım alması sağlandı. Ucak constuctoru oluşturularak sınıf ve name değişkenleri atandı. GetName ve SetName getter setter metodları oluşturuldu. Aynı şekilde getSeviyePuani ve setSeviyePuani getter setter metodları oluşturuldu. Dayanıklılık , Vurus , KaraVurusAvantaji,KartPuanıGöster, DurumGuncelle ve sinif ve alt

sinif metodları bir üst classdan override edilerek dökümanda yazan değerler return edildi. Dayanıklılık değişkenininin getter ve setter metodları yazıldı.

Aynı şekilde **Siha.java** oluşturuldu. Bu classın extends ifadesi ile Hava classından kalıtım alması sağlandı. Siha constuctoru oluşturularak sınıf ve name değişkenleri atandı. GetName ve SetName getter setter metodları oluşturuldu. Aynı şekilde getSeviyePuani ve setSeviyePuani getter setter metodları oluşturuldu. Dayanıklılık , Vurus , KaraVurusAvantaji,KartPuanıGöster, DurumGuncelle ve sinif ve alt sinif metodları bir üst classdan override edilerek dökümanda yazan değerler return edildi. Dayanıklılık değişkenininin getter ve setter metodları yazıldı.

Kara.java classının alt classları için **KFS.java** ve Obus.java tanımlandı. Bu iki classın Kara.java classından extends ifadesi kullanılarak kalıtım alması sağlandı. Burada hava classlarındaki kartlardan farklı olarak KaraVuruşAvantajı yerine **HavaVurusAvantaji** değeri girildi. Böylece kara kartları hava kartlarına karşı bir vuruş avantajına sahip olacaktı. Aynı şekilde gerekli **getter-setter** metodları , **constructorlar** ve ilgili metodlar Ucak.java veya Siha.java da olduğı gibi override edilerek karta özel değerleri atandı.

Deniz.java için Firkateyn.java ve Sida.java oluşturuldu. Extends ifadesi ile bu classların deniz classından kalıtım alması sağlandı. Bu classlara diğer classların avantajından farklı olarak karaVurusAvantaji eklendi. Benzer şekilde getter setter metodları , constructorlar ve ilgili metodlar override edilerek kartlara ait özelliklerin tanımlanması sağlandı.

Böylece kartlar için gerekli java class dosyaları tanımlanmış ve içerisine gerekli değerleri almaları sağlanmış oldu.

Ardından **Oyuncu** class yapısı oluşturuldu.

Constructor yapısı oluşturularak oyuncu bilgileri atandı. Skor değerleri için getter-setter metodları yazıldı.

Ardından kullanıcı ve **bilgisayar**.java class yapıları oluşturuldu ve bu classların oyuncu classından extends ifadesi ile kalıtım alması sağlandı. Kullanıcı için kart seç fonksiyonu Oyuncu classından **override** edildi ve yazıldı. Kullanıcı constructor fonksiyonu yazıldı. Aynı işlem bilgisayar classı için de yapıldı.

GameUI.java class yapısı oluşturuldu. Burada oyunun arayüz ve hesaplama işlemleri yapıldı. JavaFX kütüphanesi VSCode'a kuruldu.

JavaFX kütüphanesi kullandığımız için gerekli importlar yapıldı ardından public class GameUI extends Application ifadesiyle JavaFX kütüphanesini doğru bir şekilde entegre edildi. Daha sonra turSayisi, mevcuttur gibi int değerler tanımlandı. Kullanıcı ve bilgisayarın kartlarını tutmak için kullanıcı ve bilgisayar seçilen kartlar adlarında yeni arraylistler tanımlandı. Tur geçmişi listesi ve bir önceki seçilen kartlar için arraylistler tanımlandı.

@Override public void start(Stage primaryStage) girişEkranı(primaryStage);

JavaFX'den override edilerek giriş ekranımızı başlatması için yazıldı. private static String rastgeleKartTuru() Ve private static Object ***rastgeleKartEkle(List<Object> kartListesi, String kartTuru)*** tanımlandı ve bu fonksiyonlar içerisinde Kullanıcı ve bilgisayara başlangıçta atanacak olan 6 adet rasgele kart ve bu kartların uygun numaralarla isimlendirilmesi için (ucak1,Ucak2 gibi) gerekli algoritmalar yazıldı. Böylece bu fonksiyonlar sayesinde Kullanıcı ve bilgisayara rasgele 6 adet kart atılmış oldu. Bu kartlar en başta sadece Ucak, Firkateyn ve Obus olabilir çünkü Sida,Siha,KFS belirli bir seviye puanından sonra erişime açılıyor. Daha sonra private void **girişEkranı(Stage primaryStage)** tanımlandı. Bu fonksiyon bizim giriş ekranımızı oluşturacak ve oyuna geçişimizi sağlayacaktı. **BackgroundImage backgroundImage = new BackgroundImage** ifadesi ile oyunumuzun arkaplanı yüklendi. Ardından arka plan ölçeklendirmeleri yapıldı. Daha sonra giriş ekranında oyuna başla butonu yer alması için giriş ekranına bir PNG dosyası koyuldu ve bu tuşa basıldığında oyunun başlaması için **oyunaBaslaButton.setOnAction(e -> oyunBaslangicEkrani(primaryStage));** ifadesi yazıldı. Daha sonra giriş ekranında bir de Card Infos butonu eklendi kullanıcı giriş ekranında Kart bilgilerini görüntülemek isterse bu tuşa basacak ve kartların isimlerini görüntüleyebilecekti. **kartBilgileriButton.setOnAction(e -> kartBilgileriEkrani(primaryStage));**

Oyundan çıkılması için bir de EXIT butonu yapıldı ve bu butona basıldığında program durduruldu. Daha sonra yukarıda bahsedilen **private void oyunBaslangicEkrani(Stage primaryStage)** tanımlandı. Burada kullanıcı Start Butonuna bastıktan sonra gelecek ekran yazıldı. Arkaplan belirlendi. Kullanıcıdan bir tur sayısı girmesi istendi. Gerekli tur sayısına göre oyunun ilerleyen safhalarında işlem yapılacaktı. Tur sayısı girildikten sonra başla tuşuna basılması için baslaButton.setOnAction(e -> ifadesi tanımlandı ve içerisine try-catch ifadeleri ile uygun tamsayı bir tur değeri girilmesi eğer girilmediyse hata vermesi sağlandı. **primaryStage.setScene(scene);** ile ekran güncellendi. Başla tuşundan sonra ana oyunumuzun döneceği olan private void turEkraniniGoster(Stage primaryStage) fonksiyonu tanımlandı. Arkaplan yüklendi ve yukarı tarafa bilgisayarın kartlarınının konulması için HBox **bilgisayarKartPane = new HBox(10);** oluşturuldu. Böylece bilgisayarın kartları için ekranın üst tarafında yerleşmesi sağlandı aynı şekilde kullanıcı kartlarının alt tarafa yerleşmesini sağlamak için **VBox kullaniciKartBox = new VBox(10);** ifadesi tanımlandı ve konumu ayarlandı. for (Object kart : kullaniciKartlari) String imagePath = getImagePath(kart); ImageView imageView = new ImageView(new Image(imagePath)); imageView.setFitWidth(150); imageView.setFitHeight(200);

ifadesi ile kullanıcının elinde bulunan kartlar ekranda görselleriyle beraber gösterildi.

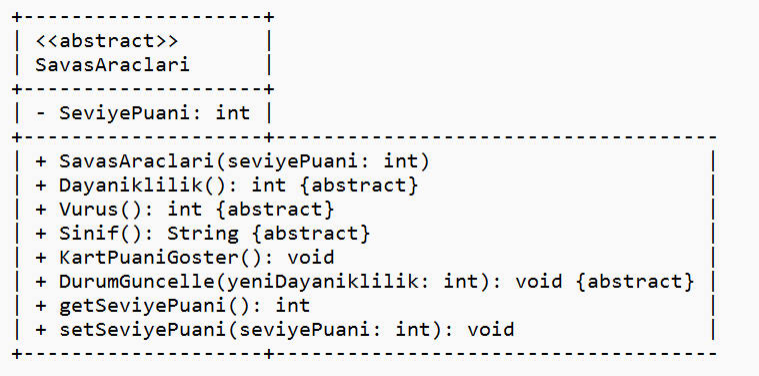
kartBox.setOnMouseClicked(e -> ifadesi ile kartların üzerine tıklandığında o kartı seçilen kart olarak belirlenmesi sağlandı. Böylece kartın üzerine basıldığında o kart seçilecek ve seçimi tamamla butonuna basıldığında o kartlar ile kullanıcı savaşa girecek. Ayrıca bir önceki turda seçilen kartlar bir ArrayListe kaydedilerek bir dahaki tur aynı 3 kartın seçilmemesi sağlandı.Her tur bu arraylist clear atılarak sıfırlandı ve böylece ard arda turlar aynı kartlar seçilemedi. Button tamamlaButton = new **Button("Seçimi Tamamla");** butonu ile seçimi tamamla butonu tanımlandı. Böylece kullanıcı kendi 6 kartı arasından 3 adet kart seçti eğer 3 karttan fazla kart seçmesine izin verilmedi. Bilgisayarın kendi 6

kartı arasından rasgele 3 kart seçmesi için gerekli algoritma yazıldı.

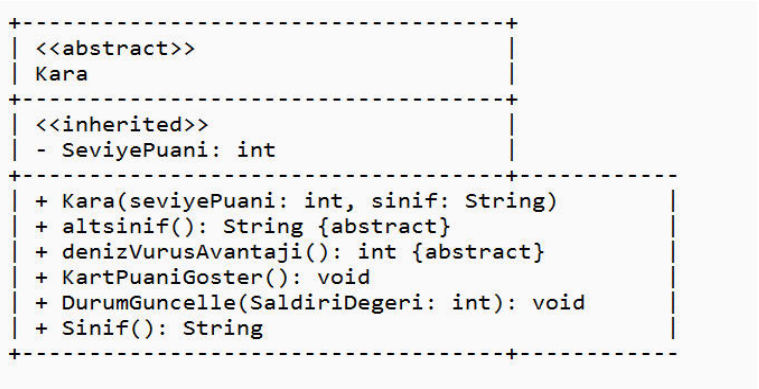
İki tarafında seçtiği kartlar belli oldu şimdi ise sıra hesaplama kısmına geldi. Burada kartları savaştırıp ölen kartları desteden çıkaracağız ve her tur +1 yeni rasgele kart ekleyeceğiz. Önce büyük bir for döngüsü açıldı ve burada **for (int i= 0; i < secilenKartlar.size(); i++)** kullanıcının seçtiği kartların büyüklüğü kadar dönmesi sağlandı. Daha sonra if ifadeleri ile **if (kullaniciKart instanceof Ucak)** eğer kullanıcının seçtiği kart Ucak türüne ait bir obje ise aşağıdaki işlemler yapıldı. Tek tek if yapıları ile bilgisayarın kartları kontrol edildi. if (bilgisayarKart instanceof Firkateyn) , else if (bilgisayarKart instanceof Obus) gibi ifadelerle bilgisayarın kartı kontrol edildi. Instanceof kartın o objeden olup olmadığını kontrol eder. Örneğin kullanıcının kartı Ucak karşısındaki kartda Firkateyn olsun. int kullaniciVurus= ucak.Vurus() + ucak.karaVurusAvantaji(); şeklinde tanımlanarak kullanıcının avantajlı olduğu vurma avantajı da hasarına eklendi. Daha sonra firkateyn.DurumGuncelle(kullaniciVurus); ile firkateyn kartının durumu güncellendi. Daha sonra if yapıları ile uçağın ve firkateynin seviye puanları ayarlandı. Eğer Uçak kartı öldüyse Firkateynin seviye puanını aldı. Eğer uçağın seviye puanı 10'dan düşükse +10 puan aldı eğer 10'dan yüksekse direkt firkateynin seviye puanı değerini aldı. Bu sistem dökümanda belirtilmişti. Firkateyn öldüyse aynı işlem uçak kartına uygulandı. Böylece ölen kart bulundu ve öldüren kartın seviye puanı ölen kartın seviye puanı kadar arttırıldı. Tüm bu işlemler diğer if yapıları için tekrar edildi. Kullanıcının kartı tek tek tarandı ve uygun if bloğuna girdi , bilgisayarın kartı tarandı ve if bloğuna girdi dayanıklılıkları hesaplandı ve öldülerse seviye puanları diğer karta atandı. Sonra dayanıklığı sıfıra düşen kartlar if bloğuna girerek ekrana "... kart öldü desteden çıkartılıyor." yazdırıldı. Ve arraylistten kullaniciKartlari.remove(k); ile kaldırıldı. Aynı işlem bilgisayar kartlarına da yapıldı ve böylece her tur kartlar savaştırıldı ve bu savaşta ölen kartlar hesaplandı. Öldüren kartın seviye puanı arttırıldı, ardından ölen kartlar desteden çıkarıldı. Her tur rasgele +1 kart eklemek için Object kullaniciYeniKart= sonrastgeleKartEkle(kullaniciKartlari, random, kullaniciUcakIndex, kullaniciObusIndex, kullaniciFirkateynIndex , kullanici.getSkor()); yapıldı. Böylece kullanıcı ve bilgisayara her tur rasgele bir kart eklendi. Ve ekranda gösterildi. sonrastgeleKartEkle ise aşağıda tanımlandı ve kullanıcıdan seviye puanı bilgisi aldı. Eğer seviye puanı 20 ' nin yukarısında ise diğer 3 kartı da rasgele eklemeye dahil etti. Eğer değilse sadece 3 adet kart arasından rasgele seçim yaptı. Giriş ekranındaki kart bilgileri ekranının içeriği private void kartBilgileriEkrani(Stage primaryStage) fonksiyonu içerisinde yazıldı. Kart görselleri ekranda gösterildi ve altlarına kartların isimleri yazıldı. Böylece oyun tamamlanmış oldu. Oyun başlıyor, giriş ekranı geliyor Start butonuna basınca program oyuncudan bir tur sayısı girmesini istiyor ve bunu bir değişkende tutuyor. Daha sonra oyun ekranı geliyor, iki tarafa da 6 adet kart dağıtılıyor. Oyuncu bilgisayarın kartlarını göremeyecek şekilde ayarlanıyor. Her iki taraf da her tur 3 kart seçip kartlar savaşıyor, ölen kartlar desteden çıkarılıp savaşın sonucunda kazanan bulunuyor.

4. TABLO

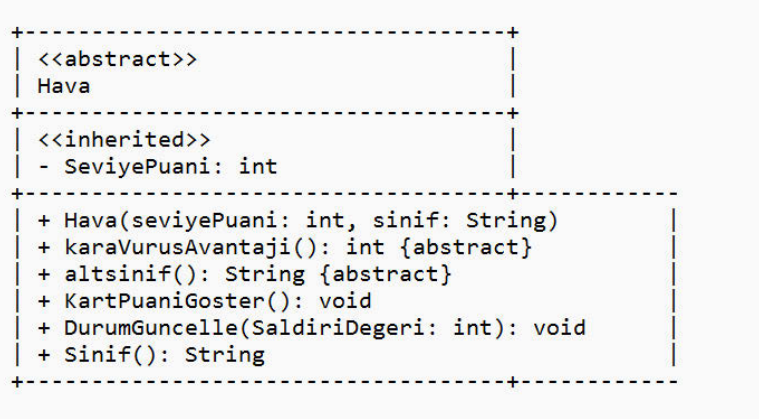
Dökümanda istenen projenin UML Diyagramı tüm sınıflar için verilmiştir.



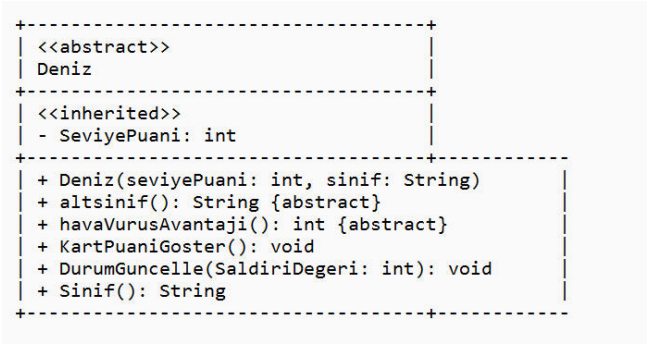
Tüm kartların kalıtım (inheritence) aldığı Savaş Araçları class UML diyagramı



Savaş araçları classından kalıtım alan Kara class UML diyagramı



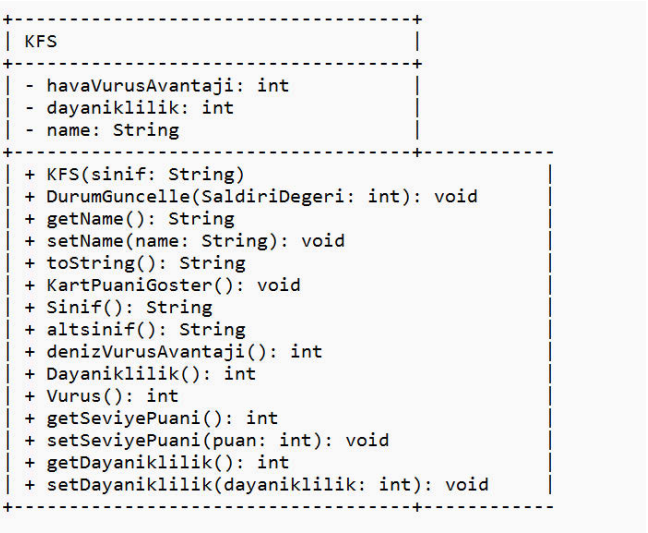
Savaş araçları classından kalıtım alan Hava class UML diyagramı

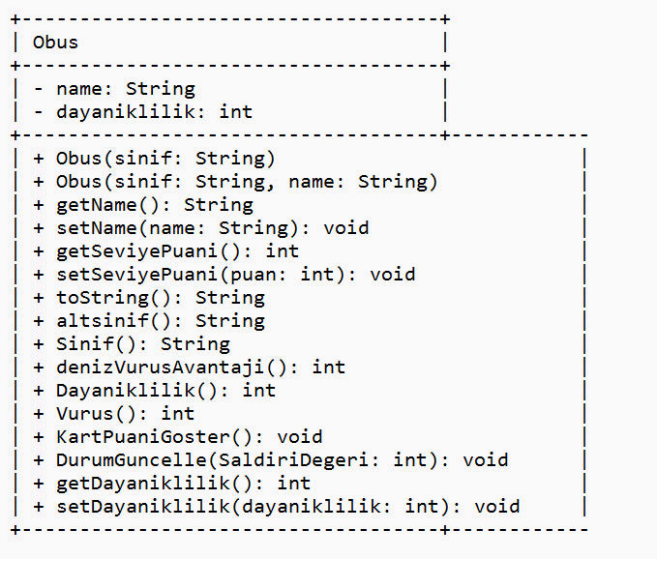


Savaş araçları classından kalıtım alana Deniz class UML diyagramı

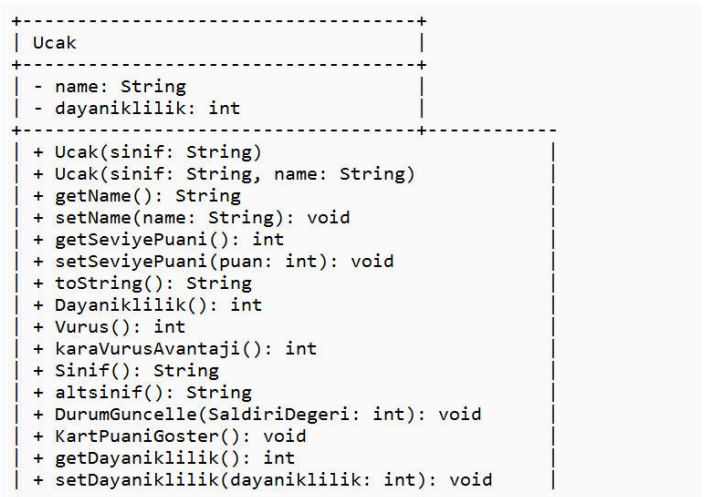
3 adet SavaşAraçları kalıtımı alan class yapısı var. Geri kalan classlar bu 3 classtan kalıtım alıyor : Deniz ,Hava, Kara.

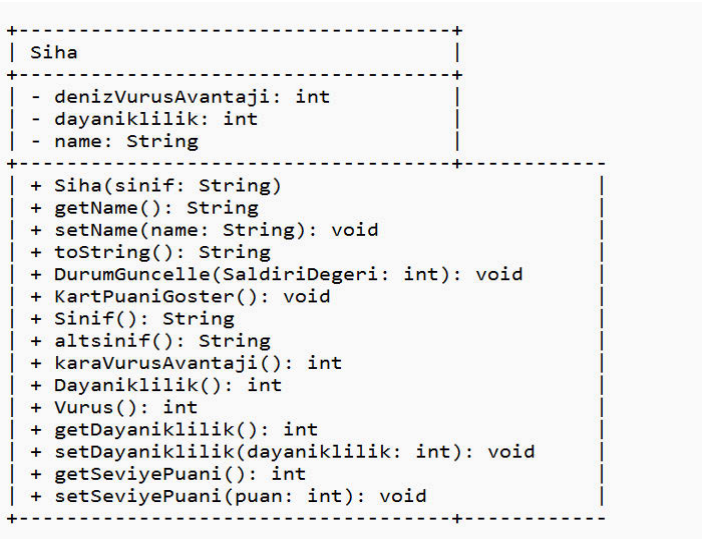
Kara Class yapısından kalıtım alan kartlarımızın (KFS, Obüs) UML diyagramı şu şekildedir :



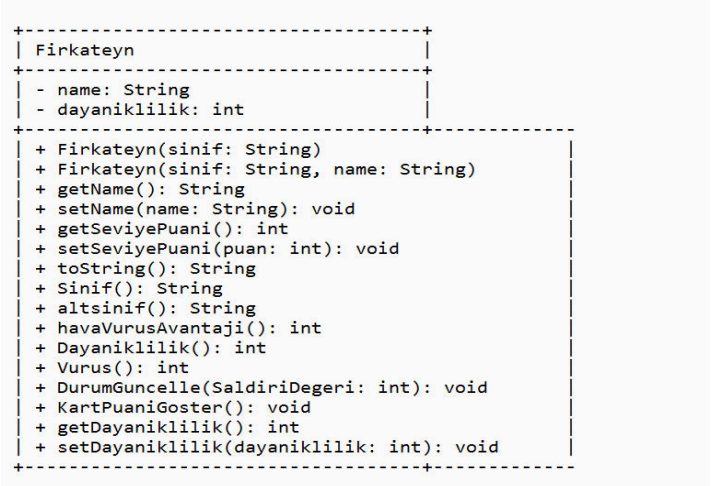


Hava Classından kalıtım ala kartlarımız (Ucak ve Siha) için UML diyagramı şu şekildedir.





Deniz classından kalıtım alan kartlarımız (Sida, Firkateyn) UML Diyagramı şu şekildedir :



```
+-----------------------------------+
| Sida                              |
+-----------------------------------+
| - karaVurusAvantaji: int          |
| - dayaniklilik: int               |
| - name: String                    |
+-----------------------------------+-------------
| + Sida(sinif: String)                          |
| + getName(): String                            |
| + setName(name: String): void                  |
| + toString(): String                           |
| + DurumGuncelle(SaldiriDegeri: int): void      |
| + KartPuaniGoster(): void                      |
| + Sinif(): String                              |
| + altsinif(): String                           |
| + havaVurusAvantaji(): int                     |
| + Dayaniklilik(): int                          |
| + Vurus(): int                                 |
| + getSeviyePuani(): int                        |
| + setSeviyePuani(puan: int): void              |
| + getDayaniklilik(): int                       |
| + setDayaniklilik(dayaniklilik: int): void     |
+-----------------------------------+-------------
```

5. KATKILAR

Proje iki grup üyesinin de katkılarıyla geliştirilmiştir. Projede iş bölümü yapılmış olup belirli yerlerde beraber haraket edilmiş belirli yerlerde görevler bölüşülmüştür.

6.SONUÇ

Proje, nesne yönelimli programlama ilkelerine uygun olarak geliştirilmiştir. Sınıflar arasında kalıtım ve ilişkiler doğru bir şekilde yapılandırılmıştır.

Savaş araçları kategorileri ve oyuncu türleri arasındaki soyutlama, sistemin genişletilebilirliğini artırmıştır.Kullanıcılar, oyun sırasında farklı stratejiler oluşturabilir ve keyifli bir deneyim yaşayabilir. Grafiksel kullanıcı arayüzü, oyunun kolay anlaşılmasını sağlamaktadır.

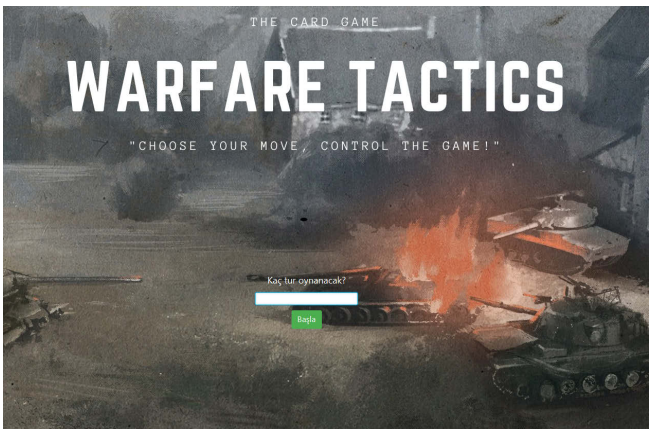
Kullanıcı ve bilgisayar kartları doğru bir şekilde dağıtılmış hesaplamalar doğru bir şekilde yapılmıştır.

Kartların seçilmesi , ekranda gösterilmesi ölen kartların ekrandan silinmesi , tur sayısının alınması başarıyla tamamlanmıştır.

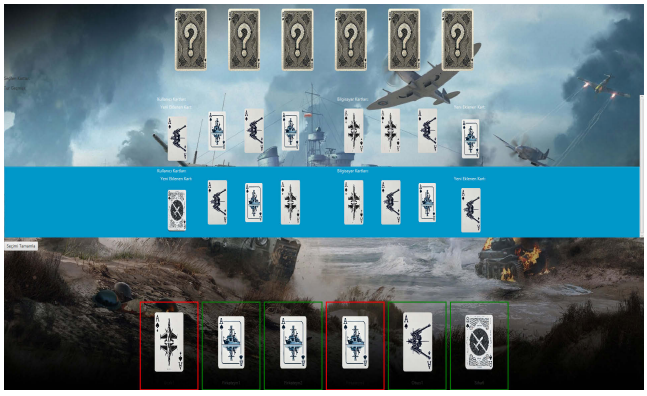
Oyunun sonucunda hangi tarafın kazandığı ekrana gösterilmiştir.

Oyun giriş ekranı

Tur Sayısı Ekranı



Oyun içi görüntü



7. KAYNAKÇA

Projede nesneye yönelik bilgisi, abstract classlar, kalıtım , polimorfizm gibi konularda kaynaklardan yardım alınmıştır.

Yardım alınan bazı kaynaklar aşağıda verilmiştir.

https://www.geeksforgeeks.org/abstract-classes-in-java/

https://www.programiz.com/java-programming/method-overloading

https://codegym.cc/tr/groups/posts/tr.136.java-polimorfizmi

https://code.tutsplus.com/introduction-to-javafx-for-game-development--cms-23835t