PROGRAMLAMA LABORATUVARI 1 PROJE 2

1st Muhammet Berat Ak
Kocaeli üniversitesi

Kocaeli/Türkiye
230202037

2nd Lütfi Alpaslan Hazar
Kocaeli üniversitesi

Kocaeli/Turkiye
230202067

I. ÖZET

Bu proje; PROLAB adlı dersin 2. projesi olan bir kart oyununun geliştirilmesine yönelik bir yazılım projesidir. Proje kapsamında UML diyagramları, akış şeması ve yazılım tasarım ilkelerine uygun bir yapı geliştirilmiştir. Bunlara ek olarak oyunda ekran görüntüleri de bulumaktadır

II. GİRİŞ

Proje, bir kart oyununun temel mantığını uygulamayı ve bu süreci yazılım tasarımında kullanılan yöntemlerle desteklemeyi amaçlamaktadır. Oyuncular, çeşitli kartlarla savaşarak skor kazanmaya çalışmaktadır. Bu makalede, projenin yapısı, algoritmaları ve sonuçları açıklanmaktadır.

III. YÖNTEM

Proje java dilinde yazılmıştır arayüz için ise swing kullanılmıştır. Projede kullanılan sınıf yapıları ve ilişkileri Şekil Fig. 3 üzerinde detaylı bir şekilde gösterilmiştir.

A. public class SavasAraclari (int dayaniklilik, int seviyePuani, String sinif, int vurus)—:

SavasAraclari sınıfı, projede tüm kartların ortak özelliklerini ve davranışlarını tanımlayan bir temel sınıftır. Bu sınıf soyut abstract olarak tasarlanmıştır ve alt sınıflar tarafından genişletilir.

—

B. public class DenizSinifi

`DenizSinifi` sınıfı, deniz sınıfına ait kartların özelliklerini ve davranışlarını tanımlayan bir soyut sınıftır. Bu sınıf, `SavasAraclari` sınıfından türetilmiştir ve alt sınıflar (örneğin `Firkateyn` ve `Sida`) tarafından genişletilir.

—

C. public class KaraSinifi

`KaraSinifi` sınıfı, kara sınıfına ait kartların özelliklerini ve davranışlarını tanımlayan bir soyut sınıftır. Bu sınıf, `SavasAraclari` sınıfından türetilmiştir ve alt sınıflar (örneğin `Obus` ve `KFS`) tarafından genişletilir.

—

D. public class HavaSinifi

`HavaSinifi` sınıfı, hava sınıfına ait kartların özelliklerini ve davranışlarını tanımlayan bir soyut sınıftır. Bu sınıf, `SavasAraclari` sınıfından türetilmiştir ve alt sınıflar (örneğin `Siha` ve `Ucak`) tarafından genişletilir.

—

E. public class Firkateyn

`Firkateyn` sınıfı, deniz sınıfına ait bir karttır. Bu sınıf, `DenizSinifi` sınıfından türetilmiştir ve hava birimlerine karşı avantaj sağlar.

—

F. public class Obus

`Obus` sınıfı, kara sınıfına ait bir karttır. Bu sınıf, `KaraSinifi` sınıfından türetilmiştir ve deniz birimlerine karşı avantaj sağlar.

—

G. public class KFS

`KFS` sınıfı, kara sınıfına ait bir karttır. Bu sınıf, `KaraSinifi` sınıfından türetilmiştir ve hem deniz hem de hava birimlerine karşı avantaj sağlar.

—

H. public class Sida

`Sida` sınıfı, deniz sınıfına ait bir karttır. Bu sınıf, `DenizSinifi` sınıfından türetilmiştir ve kara ile hava birimlerine karşı avantaj sağlar.

—

I. public class Siha

`Siha` sınıfı, hava sınıfına ait bir karttır. Bu sınıf, `HavaSinifi` sınıfından türetilmiştir ve kara ile hava birimlerine karşı avantaj sağlar.

—

J. public class Ucak

`Ucak` sınıfı, hava sınıfına ait bir karttır. Bu sınıf, `HavaSinifi` sınıfından türetilmiştir ve kara birimlerine karşı avantaj sağlar.

—

K. public class Oyuncu

`Oyuncu` sınıfı, oyuncuların bilgilerini ve sahip oldukları kartları yönetmek için kullanılır. Her oyuncu, bir `oyuncuID`, `oyuncuAdi`, `skor` ve bir `kartListesi` içerir.

—

L. public class Oyun

`Oyun` sınıfı, oyunun ana akışını ve kurallarını yönetir. Oyuncuların kart seçimi, kartların karşılaştırılması, hasar hesaplama ve skor güncellemeleri gibi tüm süreçler bu sınıfta tanımlanmıştır.

—

M. public class KartOyunuGorsel

`KartOyunuGorsel` sınıfı, oyunun görsel arayüzünü yönetmek için kullanılır. Oyuncuların kartlarını seçmesi, skorların görüntülenmesi ve oyun sonu durumlarının kullanıcıya gösterilmesi bu sınıf tarafından kontrol edilir.

Fonksiyonlar ve Açıklamaları:

- `getDayaniklilik()`: Kartın mevcut dayanıklılığını döndürür. Bu metod, kartın savaş sırasındaki dayanıklılığını kontrol etmek için kullanılır.
- `setDayaniklilik(int dayaniklilik)`: Kartın dayanıklılığını günceller. Bu metod, savaş sırasında hasar alındığında dayanıklılık değerini azaltmak için çağrılır.
- `getSeviyePuani()`: Kartın mevcut seviye puanını döndürür. Bu puan, kartın oyun boyunca ne kadar güçlü olduğunu gösterir.
- `setSeviyePuani(int seviyePuani)`: Kartın seviye puanını ayarlar. Örneğin, bir kart rakip kartı elediğinde bu metod kullanılarak seviye puanı artırılır.
- `getSinif()`: Kartın hangi sınıfa (`"Deniz"`, `"Kara"` veya `"Hava"`) ait olduğunu döndürür.
- `setSinif(String sinif)`: Kartın sınıfını günceller. Alt sınıflar bu metodu kullanarak sınıf değerini ayarlayabilir.
- `getVurus()`: Kartın temel saldırı gücünü döndürür.
- `setVurus(int vurus)`: Kartın saldırı gücünü günceller. Örneğin, bazı özel durumlarda (yeni özellikler veya geliştirmeler) saldırı gücü artırılabilir.
- `isKullanildiMi()`: Kartın daha önce kullanılıp kullanılmadığını kontrol eder. Bu metod, bir kartın aynı savaşta birden fazla kez kullanılmasını engellemek için önemlidir.
- `setKullanildiMi(boolean kullanildiMi)`: Kartın kullanıldı durumunu günceller. Örneğin, bir kart bir savaşta kullanıldığında bu değer `true` olarak ayarlanır.
- `DurumGuncelle(int hasar)`: Kartın dayanıklılığını alması gereken hasara göre günceller. Eğer dayanıklılık sıfırın altına düşerse sıfır olarak ayarlanır ve kart elenmiş olur.
- `SaldiriHesapla(SavasAraclari rakip)`: Kartın rakip karta karşı uygulayacağı saldırı gücünü hesaplar. Alt sınıflarda özel hesaplamalar yapılır (örneğin, avantaj durumları).
- `KartAdiGoster()`: Kartın adını döndürür. Bu metod, oyun sırasında kartın adını göstermek için kullanılır.
- `KartPuaniGoster()`: Kartın mevcut seviye puanını gösterir. Oyuncuların kartın ne kadar güçlü olduğunu anlaması için bu bilgi önemlidir.
- `getKartGorselYolu()`: Kartın görsel dosya yolunu döndürür. Bu metod, görsel arayüzde kartın doğru bir şekilde gösterilmesini sağlar.
- `baslangicKartDagitimi()`: Oyunculara başlangıçta 6 adet rastgele kart dağıtır. Oyunun başında kartların doğru bir şekilde verilmesini sağlar.
- `rastgeleKartOlustur(boolean yirmiPuanaUlasildi)` Yeni bir kart oluşturur. Kartın türü (`Deniz`, `Kara` veya `Hava`) rastgele belirlenir. Eğer oyuncunun skoru 20 veya daha fazlaysa, daha güçlü kartlar oluşturulur.
  - Eğer `yirmiPuanaUlasildi` `true` ise, daha üst düzey kartlar oluşturulur.
  - Kart türleri rastgele belirlenir (`Firkateyn`, `Obus`, `Ucak` vb.).
- `kartSec(Oyuncu oyuncu)`: Bir oyuncunun 3 adet kart seçmesini sağlar. Bilgisayar için rastgele seçim yapılırken, kullanıcı için manuel seçim yapılabilir.
  - Bilgisayar oyuncusu: Kartlar rastgele seçilir.
  - Kullanıcı oyuncusu: Oyuncunun manuel olarak 3 kart seçmesi sağlanır.
- adimKaydet(Oyuncu saldiran, Oyuncu savunan, SavasAraclari saldiranKart, SavasAraclari savunanKart, int vurulanHasar, String saldiriYapan)—: Bir savaş adımının sonuçlarını bir log (`log`) dosyasına kaydeder. Bu, oyunun tekrar oynatılabilirliği ve analiz edilebilirliği için önemlidir.
  - `saldiran` ve `savunan` parametreleri, bu savaşta yer alan iki oyuncuyu ifade eder.
  - `saldiranKart` ve `savunanKart`, bu savaşta kullanılan iki kartı ifade eder.
  - `vurulanHasar`, saldırı sonucunda rakibe verilen hasar değeridir.
  - `saldiriYapan`, bu saldırının hangi oyuncu tarafından yapıldığını belirtir (`"Oyuncu"` veya `"Bilgisayar"`).
- `oyunuSonlandir()`: Oyunun bitip bitmediğini kontrol eder. Eğer oyun biterse, sonuçları ekrana yazdırır ve bir günlük dosyasına kaydeder. Ayrıca kazanan oyuncuyu belirler:
  - Eğer bir oyuncunun kart listesi boşsa, oyun sona erer.
  - Eğer belirlenen maksimum hamle sayısına ulaşılmışsa, oyun sona erer.
  - Sonuçlar skor üzerinden hesaplanır.
- `guncelleOyuncuKartlariPanel()`: Oyuncunun elindeki kartları görsel olarak günceller. Seçilen kartlar veya savaş sonrası eklenen kartlar bu panelde gösterilir.
- `guncelleBilgisayarKartlariPanel()`: Bilgisayar oyuncusunun elindeki kartları günceller. Bilgisayarın kartları yüzü kapalı bir şekilde gösterilir.

- `oyunSonuEkrani()`: Oyunun sona erdiğini kullanıcıya gösterir. Kazanan veya berabere durumu ekrana yazdırılır.
- `kartSeviyePuaniGuncelle(int kartSeviyePuani)`: Oyuncuların ve bilgisayarın kartlarındaki seviye puanlarını günceller. Bu metod, özellikle başlangıçta verilen değerler için kullanılır.
- `SavasBaslatListener`: Oyuncuların seçtikleri kartları savaştıran bir dinleyicidir. Bu dinleyici, savaş işlemini başlatır ve sonuçları görsel olarak günceller.
- `SonrakiAdimListener`: Bir sonraki tura geçmek için kullanılan bir dinleyicidir. Kullanıcı bir sonraki tur için hazırlık yapar ve arayüz yeniden güncellenir.
- `tumKartlarKullanildiMi()`: Oyuncunun elindeki tüm kartların kullanılıp kullanılmadığını kontrol eder. Eğer tüm kartlar kullanılmışsa, yeni kartlar dağıtılır.
- `kartSeviyePuaniAta(int puan)`: Oyuncuların elindeki tüm kartlara başlangıçta verilen bir seviye puanı atar. Bu metod, oyunun başlangıç dengesi için kullanılır.

IV. KAZANIMLAR

Bu proje, hem teknik hem de bireysel/ekip olarak birçok kazanım sağlamıştır. Biz, ekip olarak bu süreçte aşağıdaki kazanımları elde ettik:

- **Nesne Yönelimli Programlama Prensiplerinin Uygulanması:** Java programlama dili kullanılarak, sınıflar arasında soyutlama (*abstraction*), kalıtım (*inheritance*) ve çok biçimlilik (*polymorphism*) gibi nesne yönelimli programlama prensiplerini uygulamayı öğrendik. Özellikle soyut sınıflar ile alt sınıflar arasındaki ilişkiyi doğru bir şekilde kurarak yazılım mimarisini geliştirdik.
- **UML Diyagramlarının Kullanımı:** Projeye başlamadan önce, ekip olarak UML diyagramlarını hazırladık. Bu diyagramlar, sınıflar arasındaki ilişkiyi görselleştirmemize ve projeyi daha planlı bir şekilde hayata geçirmemize yardımcı oldu.
- **Java Swing ile GUI Geliştirme:** Proje kapsamında, Java Swing kullanarak bir grafiksel kullanıcı arayüzü (GUI) tasarladık. Oyuncuların kart seçimlerini kolaylaştıran ve savaş mekaniklerini görsel olarak yansıtan bu arayüz, kullanıcı dostu bir deneyim sunmaktadır.
- **Loglama ve Dosya İşlemleri:** Oyundaki her hamlenin kaydedildiği bir günlük (*log*) sistemi geliştirdik. Bu günlük sistemi, oyun sırasında gerçekleşen işlemlerin kayıt altına alınmasını sağladı ve oyun sonrası analiz yapmamıza olanak tanıdı.
- **Ekip Çalışması:** Arkadaşımla birlikte görev paylaşımı yaparak projeyi yönetmeyi öğrendik. Kodun belirli bölümlerini ayrı ayrı geliştirip birleştirdik. Bu süreçte, fikir alışverişi ve ortak karar alma gibi ekip çalışması gerektiren becerilerimizi geliştirdik.
- **Problem Çözme Yeteneği:** Karşılaştığımız zorlukları çözmek için alternatif yollar geliştirdik. Örneğin, kartların savaş mekaniklerini ve avantajlarını dengelemek için birkaç test senaryosu oluşturduk ve oyunun matematiksel doğruluğunu test ettik.

V. YORUMLAR

Projeyi tamamlarken hem teknik hem de süreçle ilgili değerlendirmeler yaptık. Süreçte karşılaştığımız zorlukları ve bu zorluklara bulduğumuz çözümleri aşağıda açıkladık:

A. Karşılaşılan Zorluklar

- **Grafiksel Arayüz Geliştirme:** Java Swing ile görsel bir arayüz oluştururken, dinamik kart güncellemelerini ve kullanıcının seçimlerine uygun bir yapı oluşturmayı öğrenmemiz gerekti. Özellikle kart seçimi ve savaş sonuçlarının arayüze yansıtılması sırasında çeşitli hatalarla karşılaştık. Bu sorunları olay dinleyiciler (`ActionListener`) ve arayüz güncellemeleriyle çözdük.
- **Kart Savaş Mekaniklerinin Dengelenmesi:** Kartların avantaj ve dezavantajlarının saldırı gücüne nasıl ekleneceği konusunda bazı zorluklar yaşadık. Arkadaşımla birlikte test senaryoları oluşturarak ve sonuçları analiz ederek bu problemi dengelemeyi başardık.

B. Proje Değerlendirmesi

Projemizi tamamladıktan sonra aşağıdaki değerlendirmeleri yaptık:

- **Genel Başarı:** Ekip olarak hem teknik hem de yazılım geliştirme süreçleri açısından başarılı bir projeye imza attığımızı düşünüyoruz. Kart oyununun mekanikleri çalışır durumda ve kullanıcı deneyimi açısından tatmin edici bir sonuç ortaya koyduk.
- **Etkili Görev Paylaşımı:** Arkadaşımla görevleri bölerek kodlama sürecini hızlandırdık. Birimizin yazdığı bir sınıf, diğerinin yazdığı sınıflarla uyumlu çalıştı. Bu da ekip içi koordinasyonumuzun güçlü olduğunu gösterdi.
- **Öğrenme Süreci:** Proje sırasında nesne yönelimli programlama prensiplerini gerçek bir uygulama üzerinde deneyimleme şansı bulduk. Ayrıca, grafiksel arayüz geliştirme ve oyun mantığını entegre etme gibi beceriler kazandık.

VI. SONUÇ

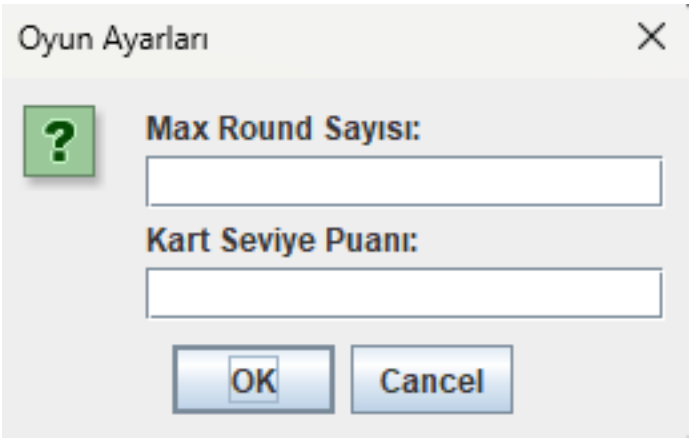

Fig. 1. Başlangıçta kullanıcıdan girilmesi istenen değerler

Oyunun savaş ekranı fig.2.'deki görseldeki gibidir.

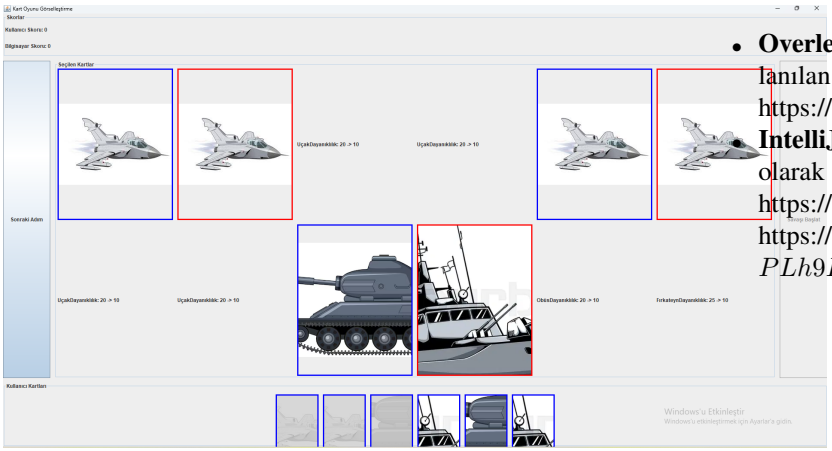

Fig. 2. Oyun savaş ekranı

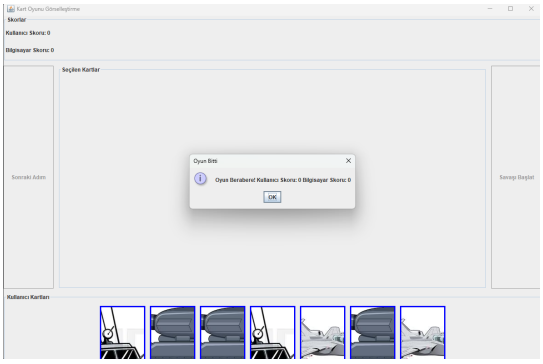

Fig. 3. Bitiş ekranı

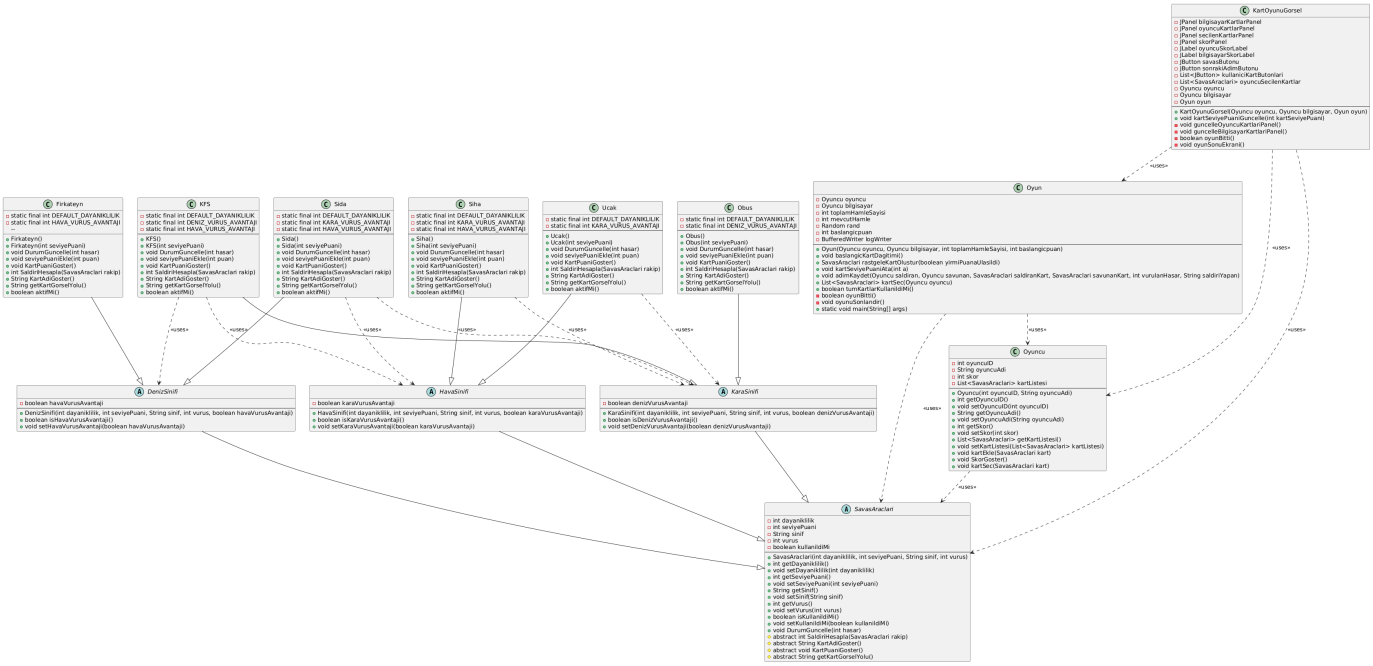

Fig. 4. Kodun UML Diyagramı

VII. KAYNAKÇA

- **Overleaf Platformu:** IEEE raporu hazırlanırken kullanılan LaTeX düzenleyici. https://www.overleaf.com/
- **IntelliJ IDEA Documentation:** Proje geliştirme ortamı olarak IntelliJ IDEA kullanılmıştır. https://www.jetbrains.com/idea/documentation/
- https://www.youtube.com/watchv=H$_S$Kv07648$list$ = $PLh9ECzBB8tJPFTpuHKhYayis0H9pS6_rIchatgpt.com$