

PROGRAMLAMA LABORATUVARI

PROJE 2

Anıl Engin Keretli

Kocaeli Üniversitesi

Kocaeli/TÜRKİYE

230201128

I. Giriş

Projenin amacı, nesneye yönelik programlama (NYP) prensiplerini kullanarak bir savaş araçları kart oyunu tasarlamaktır. Proje boyunca C++ veya Java gibi bir programlama dili kullanılarak, oyuncu ile bilgisayarın rekabet ettiği bir oyun geliştirilmiştir.

Bu süreçte, sınıflar, miras, kapsülleme ve çok biçimlilik gibi temel NYP kavramları uygulamalı olarak öğrenilmiştir.

II. Sistem Gereksinimleri

A. Fonksiyonel Gereksinimler

Kart Yönetimi: Kartların sınıflara ayrılması ve oyun sırasında belirli kurallara göre kullanılabilir hale getirilmesi.

Oyuncu Yönetimi: Bilgisayar ve kullanıcı arasında kart seçimi ve rekabet mekanizmasının sağlanması.
Saldırı Mekanizması: Kartlar arasındaki dayanıklılık ve vuruş gücünün sınıflar aracılığıyla hesaplanması.

B. Fonksiyonel Olmayan Gereksinimler
Performans: Her bir oyun hamlesinin hızlı ve doğru bir şekilde tamamlanması.

Kullanılabilirlik: Oyuncuların görsel bir arayüz üzerinden kolaylıkla oyunu takip edebilmesi.

```
// Abstract Hava Sınıfı
public abstract class HavaAracları extends SavasAracları { 2 usages 2 inheritors
    protected String sınıf; 2 usages

    // Abstract Özellikler
    public abstract String getAltSınıf();  no usages 2 implementations
    public abstract void setAltSınıf(String altSınıf);  4 usages 2 implementations

    public abstract int getKaraVurusAvantajı();  no usages 2 implementations
    public abstract void setKaraVurusAvantajı(int karaVurusAvantajı);  4 usages 2 implementations

    // Constructor
    public HavaAracları() { 4 usages
        super();
        this.setSınıf("Hava");
    }

    public HavaAracları(int seviyePuanı) { super(seviyePuanı); // Üst sınıfın constructor'ını çağırır }

    @Override 1 usage 2 overrides
    public String getSınıf() { return sınıf; }

    @Override 3 usages 2 overrides
    public void setSınıf(String sınıf) { this.sınıf = sınıf; }
}
```

Ahmet Burak Karkaç

Kocaeli Üniversitesi

Kocaeli/TÜRKİYE

220201173

III. Veri Yapıları ve Sınıflar

SavasAracları Sınıfının Tanımı

1. Sınıfın Doğası:

SavasAracları bir abstract class'tır. Bu nedenle kendisinden doğrudan nesne oluşturulamaz. Alt sınıflar bu sınıfı extend ederek (miras alarak) abstract metotları doldurmalıdır.

Özellikler (Attributes):

seviyePuanı: Her savaş aracının sahip olduğu bir seviye puanı.

isim: Aracın adı.

kullanım: Savaş aracının kullanılıp kullanılmadığını gösteren bir boolean değişken.

Constructor (Yapıcı Metotlar):

Varsayılan constructor (SavasAracları()) tüm özelliklere başlangıç değerleri atar.

Parametreli constructor (SavasAracları(int seviyePuanı)) sadece seviyePuanı özelliğini dışarıdan alınan bir değerle başlatır.

Metotlar

Abstract Metotlar (Uygulaması Alt Sınıflarda Zorunlu):

getIsim / setIsim: Savaş aracının ismini almak ve değiştirmek için.

getDayanıklılık / setDayanıklılık: Aracın dayanıklılığını yönetmek için.

getSınıf / setSınıf: Aracın hangi sınıfa ait olduğunu belirtmek için (örneğin, tank, uçak vs.).

getVurus / setVurus: Savaş aracının saldırısı gücünü belirtmek için.

DurumGuncelle(int saldırıDegeri): Aracın durumunu güncellemek için bir mekanizma. Saldırı sonucu aracın dayanıklılığı gibi özelliklerini değiştirir.

Concrete Metotlar (İçeriği Tanımlı):

KartPuaniGoster(): Savaş aracının tüm özelliklerini (dayanıklılık, sınıf, vuruş gücü ve seviye puanı) ekrana yazdırır.

III. Yöntem

Projenin geliştirilmesi, nesneye yönelik programlama (NYP) ilkelerine uygun bir şekilde sistematik olarak yürütülmüştür. Proje, aşağıdaki adımlarla gerçekleştirılmıştır:

A. Sistem Tasarımı

Sistem, soyutlama, miras, kapsülleme ve çok biçimlilik özelliklerini destekleyen sınıflarla tasarlanmıştır. Tasarım süreci, UML diyagramları kullanılarak planlanmış ve aşağıdaki şekilde organize edilmiştir:

Sınıf Yapısı:

Tüm kartların ortak özelliklerini kapsayan bir Savaş Araçları sınıfı oluşturulmuştur. Savaş Araçları sınıfı, abstract bir sınıf olarak tanımlanmış ve tüm alt sınıflar için temel işlevselligi sağlamıştır. Alt sınıflar, kartların kategorilerine göre ayrılmıştır: Hava, Kara ve Deniz.

Kullanıcı ve Bilgisayar Yönetimi:

Oyuncuların kart seçimleri ve skorlarının tutulması için bir Oyuncu sınıfı oluşturulmuştur. Kullanıcı ve bilgisayar için ayrı nesneler türetilmiş ve bilgisayarın rastgele seçim yapması sağlanmıştır.

Oyun Yönetimi:

Oyun akışının kontrolü, Oyun sınıfı üzerinden gerçekleştirilmiştir. Her hamlede saldırı hesaplamaları ve skor güncellemeleri SaldırıHesapla() metodu ile yapılmıştır. Oyunun bitiş şartları (kartların tükenmesi veya maksimum hamle sayısının tamamlanması) bu sınıfta tanımlanmıştır.

▪Savaş Araçları pseudo code

▪class SavasAraclari (Abstract):

```
# Attributes
seviyePuani (int)      # Default: 0
isim (String)          # Default: "?"
kullanim (boolean)     # Default:
false
```

```
# Constructor - Default
function SavasAraclari():
    set seviyePuani = 0
    set isim = "?"
    set kullanim = false
```

```
# Constructor - With Parameter
function SavasAraclari(seviyePuani):
    set this.seviyePuani = seviyePuani
```

```
# Abstract Methods (Must be
implemented by subclasses)
abstract function getIsim(): String
abstract function setIsim(isim: String)

abstract function getDayaniklilik(): int
abstract function
setDayaniklilik(dayaniklilik: int)

abstract function getSinif(): String
abstract function setSinif(sinif: String)
```

```
abstract function getVurus(): int
abstract function setVurus(vurus: int)
```

```
abstract function
DurumGuncelle(saldırıDegeri: int)
```

```
# Concrete Method
function KartPuaniGoster():
    print("Dayanıklılık: " +
getDayaniklilik())
    print("Sınıf: " + getSinif())
    print("Vuruş: " + getVurus())
    print("Seviye Puanı:
" + seviyePuani)
```

B. Uygulama Süreci

1. Kodlama

Proje, Java programlama dili kullanılarak geliştirilmiştir. Aşağıda, sınıfların temel özellikleri ve bu süreçte kullanılan metodlar açıklanmıştır:

Savaş Araçları Sınıfı:

Bu sınıf, dayanıklılık, seviyePuanı, ve vurus gibi temel özellikleri içerir. Ayrıca, saldırısı ve dayanıklılık güncellemeleri için bir soyut metod olan DurumGuncelle() tanımlanmıştır.

Alt Sınıflar:

Her bir araç tipi için ayrı sınıflar türetilmiş ve bunların özellikleri özelleştirilmiştir. Örneğin, uçak kartları, kara araçlarına karşı avantajlı olacak şekilde programlanmıştır.

Uçak Sınıfı: Hava kategorisinde yer alır ve kara araçlarına ekstra hasar verir.

KFS Sınıfı: Kara kategorisindedir ve hava araçlarına karşı avantajlidir.

Oyuncu Sınıfı:

Bu sınıfıta kartListesi ve skor özellikleri tanımlanmış, oyuncuların kart seçimleri ve hamleleri için metodlar yazılmıştır. Kullanıcı, kartlarını manuel olarak seçenekken bilgisayar rastgele seçim yapmaktadır.

Oyun Sınıfı:

Oyunun ana akışı, main() metodu ile buradan kontrol edilmektedir. Kartların saldırısı değerleri ve dayanıklılık puanları karşılaştırılarak kazanan belirlenmiştir.

2. Veri Akışı ve Kurallar

Oyunun dinamikleri şu şekilde belirlenmiştir:

Her hamlede, oyuncular sırayla 3 kart seçer ve bu kartlar birebir karşılaşır.

Saldırı sonrası, dayanıklılık değeri 0 olan kartlar elenir.

Oyuncular her hamlenin başında yeni bir kart alır.

3. Örnek Senaryo

Bir oyunun örnek akışı aşağıdaki gibidir:

Başlangıç: Oyuncunun kartları: Uçak, Obüs, Firkateyn. Bilgisayarın kartları: Firkateyn, Obüs, SIDA.

Hamle 1: Oyuncu Uçak - Obüs - Firkateyn seçerken, bilgisayar Firkateyn - Obüs - SIDA seçer. Skorlar hesaplanır ve yeni kartlar dağıtılır.

Oyun Sonu: Beş hamlenin sonunda dayanıklılık ve skor değerlerine göre kazanan belirlenir.

C. Algoritmalar ve İşlevler

1. Kart oluşturma Algoritması

çift parametreli alır aldığı ilk string parametresine göre oluşturulacak kartın tek parametreli construractoru çağrılır. Constructura verilen integer değer oluşturulan kartın kaçinci defa oluşturduğu bilgisini içerir bu bilgiyi static değer olarak tutar

```
public static SavasAracları KartOlustur(String kart, int i) { 4 usages
    switch (kart) {
        case "ucak":
            return new Ucak(i);
        case "obus":
            return new Obus(i);
        case "firkateyn":
            return new Firkateyn(i);
        case "sida":
            return new Sida(i);
        case "siha":
            return new Siha(i);
        case "KFS":
            return new KFS(i);
        default:
            throw new IllegalArgumentException("Bilinmeyen Kart: " + kart);
    }
}
```

KATKILAR

Projede Anıl Engin Keretli, görsel fonksiyonlar ve constructor metodalar üzerinde çalıştı; raporlamayı da üstlendi. Ahmet Burak Karkaç ise hesaplama fonksiyonları ve kart sistemi üzerine çalıştı.

KAYNAKÇA

<https://docs.oracle.com/en/java/javase/22/>

<https://bulutistan.com/blog/iliskisel-veri-tabani-nedir-nasıl-calisır-ozellikleri-ornekleri-modelleri/>

<https://proceedingsoftheieee.ieee.org/journal/>

UML DİYARGRAMI

