Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Programlama Laboratuarı I

Sporcu Kart Oyunu

Taha Uz

maskeden_2000@hotmail.com

Projenin Özeti:

Programlama Laboratuarı II Projesi olarak bizden “Sporcu Kart Oyunu” adlı uygulama geliştirmemiz beklenmektedir.

Ben proje için Java programlama dilini kullanmayı ve Netbeans ile Intelij geliştirme ortamında yazmayı seçtim.

Proje dökümanından bizden beklenen bir oyuncunun otomatik oyuncuyla rekabet edebileceği basit bir kart oyunu tasarlamaktır.

Projede ben Java programlama dilinde bulunan “Swing” adlı arayüz tasarım kütüphanesinden yararlandım.Bu kütüphane sayesinde geliştirdiğim oyunun ana menüsünü ve oyun alanının tasarımını oluşturdum.

İlk olarak “test.java” classının içerisinde “Sporcu.java”,”Futbolcu.java”ve”Basketbolcu.java” adlı classların kontrolü ile oyundaki sporcuları tek tek tanımladım. Sonra “Oyuncu.java” , ”Bilgisayar.java” ve ”Kullanıcı.java” classları yardımıyla oyuncuları tanıttım.Oyunculara “test.java” classında destelerini vererek “OyunAlani.form” classına yolladım. “OyunAlani.form” classinda oyunun kurallarini nasıl oynanması gerektiğinin algoritmasını oluşturdum. Son olarak kazananı belirledim ve “EXIT GAME” butonu ile oyuncunun programdan çıkışını gerçekleştirdim .

1.GİRİŞ

Proje için Java programlama dilini kullanmayı ve Netbeans ile Intelij geliştirme ortamında yazmayı seçtim.

Java programlama dili ; Sun Microsystems mühendislerinden James Gosling tarafından geliştirilmeye başlanmış açık kodlu , nesneye yönelik , zeminden bağımsız , yüksek verimli , çok işlevli , yüksek seviye , adım adım işletilen (yorumlanan-interpreted) bir dildir.

Netbeans platformu; Oracle tarafından geliştirilen bir java geliştirme ortamıdır (IDE) ve ücretsiz olarak dağıtılmaktadır. Özellikle kullanıcı arayüzü tasarımında sağladığı kolaylıklardan dolayı tercih edilmektedir. Benim de iki geliştirme ortamı kullanmamın sebebi Intelij ile arayüz tasarımında zorlanmamdan dolayı Netbeans 'i tercih etmiş olmamdır.

IntelliJ IDEA Java'da bilgisayar yazılımı geliştirmek için kullanılan bir tümleşik geliştirme ortamıdır (IDE). JetBrains (resmi adı ile IntelliJ) şirketi tarafından geliştirilmektedir. Hem Apache 2 lisansı altında yayınlanan topluluk sürümü hem de sahipli ticari sürümü bulunmaktadır. Her ikisi de ticari amaçlı geliştirmelerde kullanılabilmektedir.Bu geliştirme ortamımı tercih etmemdeki sebep kullanıcıya kısa yollar ve çeşitli avantajlar ile hızlı kodlar yazabilme imkanı sağlamasıdır.

2.TEMEL BİLGİLER

Projemde;"Test.java","Sporcu.java","Futbolcu.java","Basketbolcu.java","Oyuncu.java","Bilgisayar.java","Kullanıcı.java","AnaMenu.java","OyunAlani.java" olmak üzere toplam 9 tane ".java" uzantılı dosyalardan oluşmaktadır.Bu raporda anlatım olarak teker teker ".java" uzantıların adı ile alt başlıklarına ayırarak anlatıcam.

2.1.Sporcu.java :

Bu "Sporcu.java" classinda bizden istenen sporcuların genel özelliklerini atıyorum. Sporcunun ismini "sporcuIsim" adlı String tipinde değişken ile tutuyorum. Sporcunun takımını "sporcuTakim" adlı String tipinde değişken ile tutuyorum.

Bu işlemleri yaptıktan sonra Sporcu classının yapısını hazırlıyorum aşağıdaki resimdeki gibi.

```
public Sporcu(String sporcuIsim, String sporcuTakim) {
    this.sporcuIsim = sporcuIsim;
    this.sporcuTakim = sporcuTakim;
}

public Sporcu() {
    this.sporcuIsim = "Default sporcuIsim";
    this.sporcuTakim = "Default sporcuTakim";
}
```

Classın yapısını hazırladıktan sonra classın özelliklerini metotlar ile tanımlıyorum sırayla ; **setSporcuIsim**,**setSporcuTakim**,**sporcuPuaniGoster**,**getSporcuIsim**,**getSporcuTakim**,**getPenalti**,**getSerbestAtis**,**getKaleciKarsiKarsiya**,**getIkilik**,**getUcluk**,**getSporTuru** olmak üzere metotları yazıyorum.

2.2.Futbolcu.java :

Bu "Futbolcu.java" classini ilk olarak "Sporcu.java" classindan kalıtım almasını sağladim aşağıdaki kod ile ;

public class Futbolcu **extends** Sporcu{

}

Bu işlemi yaptıktan sonra bu classa özel değişkenlerimi tanımladım ; "penalti","serbesAtis","kaleciKarsiKarsiya","KartKullanildiMi" olmak üzere.

Bu “Futbolcu.java” classın yapısını şu şekilde hazırladım ;

```
public Futbolcu(String sporcuIsim, String sporcuTakim, int penalti, int serbestAtis, int kaleciKarsiKarsiya)
    super(sporcuIsim, sporcuTakim);
    this.penalti = penalti;
    this.serbestAtis = serbestAtis;
    this.kaleciKarsiKarsiya = kaleciKarsiKarsiya;
}

public Futbolcu() {
    super();
    this.penalti = 0;
    this.serbestAtis = 0;
    this.kaleciKarsiKarsiya = 0;
}
```

Classın yapısını hazırladıktan sonra classın özelliklerini metotlar ile tanımlıyorum sırasıyla ; **getPenalti,setPenalti,setSerbestAtis,setKaleciKarsiKarsiya,setKartKullanildiMi,getserbestAtis,getKaleciKarsiKarsiya,SporTuru,sporcuPuaniGoster** şeklinde metodlarını oluşturdum.

Bazı metotları şu şekilde **Override** ederek ana classdaki (Sporcu.java) özelliklerden yararlanıcak şekilde özelleştirerek yazıyorum ;

```
@Override
public int getPenalti() {
    return penalti;
}

@Override
public int getSerbestAtis() {
    return serbestAtis;
}

@Override
public int getKaleciKarsiKarsiya() {
    return kaleciKarsiKarsiya;
}

@Override
public String SporTuru() {
    return "Futbolcu";
}

@Override
public String sporcuPuaniGoster() {
    super.sporcuPuaniGoster();
    this.kartKullanildiMi = true;
    System.out.println("Penalti : " + this.penalti + "\nSerbest Atis : " + this.serbestAtis + "\nKaleci ile karsi karsiya : " + this.kaleciKarsiKars

    return "";
}
```

2.3.Basketbolcu.java :

Bu “Basketbolcu.java” classini ilk olarak “Sporcu.java” kalıtım almasını sağladım aşağıdaki kod ile ;

public class Basketbolcu **extends** Sporcu{

}

Class a özel değişkenlerimi tanımladım ; “ikilik”,”ucluk”,”serbestAtis”,”kartKullanildiMi” olmak üzere.

Bu “Basketbolcu.java” classın yapısını şu şekilde hazırladım ;

```
public Basketbolcu(String sporcuIsim, String sporcuTakim, int ikilik, int ucluk, int serbestAtis) {
    super(sporcuIsim, sporcuTakim);
    this.ikilik = ikilik;
    this.ucluk = ucluk;
    this.serbestAtis = serbestAtis;
}

public Basketbolcu() {
    super();
    this.ikilik = 0;
    this.ucluk = 0;
    this.serbestAtis = 0;
}
```

Class yapısını hazırladıktan sonra classın özelliklerini metotlar ile tanımlıyorum sırasıyla ;**getIkilik,getUcluk,getSerbestAtis,setIkilik,setUcluk,setSerbestAtis,setKartKullanildiMi,SporTuru,sporcuPuaniGoster** şeklinde metodlarını oluşturdum.

Bazı metotları şu şekilde Override ederek ana classdaki (Sporcu.java) özelliklerinden yararlanıcak şekilde özelleştirerek yazıyorum;

```
@Override
public int getIkilik() {
    return ikilik;
}

@Override
public int getUcluk() {
    return ucluk;
}

@Override
public int getSerbestAtis() {
    return serbestAtis;
}

@Override
public String SporTuru() {
    return "Basketbolcu";
}

@Override
public String sporcuPuaniGoster() {
    super.sporcuPuaniGoster();
    this.kartKullanildiMi = true;
    System.out.println("Ikilik : " + this.ikilik + "\nUcluk : " + this.ucluk + "\nSerbest Atis : " + this.serbestAtis);

    return "";
}
```

2.4.Oyuncu.java :

Bu class oyuncuların temel özelliklerini atıyorum bu hem bilgisayarın hem de kullanıcının ortak özelliklerini kapsıyacak şekilde tanımladım.Bu özellikler şu şekilde; **“oyuncuID”** string tipinde bir değişkendir ve oyuncunun isim ile birlikte belirli numaralar atanmış ismini tutar.**”oyuncuAdi”** string tipinde bir değişkendir ve oyuncunun giriş ekranında alınan ismidir.**”skor”** int tipindeki bu değişken kullanıcının yaptığı hamlelerde aldığı puanları tutar ve bunun üzerine koyarak toplam puana kadar gider. **“A_oyuncuDeck”** ve **”B_oyuncuDeck”** adında ArrayList tipinde bilgisayarın ve kullanıcının karışmış bir şekilde gelen destesini tutar bu iki değişken.

Bu değişkenleri tanımladıktan sonra classın yapısını hazırlıyorum şu şekilde ;

```
public Oyuncu(String oyuncuID, String oyuncuAdi, int skor) {
    this.oyuncuID = oyuncuID;
    this.oyuncuAdi = oyuncuAdi;
    this.skor = skor;
}

public Oyuncu() {
    this.oyuncuID = "DefaultOyuncuID#0000";
    this.oyuncuAdi = "DefaultOyuncuID";
    this.skor = 0;
}
```

Classın yapısını hazırladıktan sonra classın özelliklerini metotlar ile tanımlıyorum sırasıyla;**setOyuncuID,setOyuncuAdi,setSkor,getOyuncuID,getOyuncuAdi,getSkor,kartSec,kartListesi,skorArttir,kartCikar,desteGonder,neKadarKaldi,SkorGoster** şeklinde metotlarını oluşturdum.

2.5.Bilgisayar.java :

Bu “Bilgisatar.java” classını ilk olarak “Oyuncu.java” kalıtım almasını sağlıyorum tanımladığım ortak özellikleri kullanması için.

public class Bilgisayar **extends** Oyuncu{

}

Bu işlemleri bitirdikten sonra bu classa ait özel değişkenleri tanımlıyorum;**”temp_SporTuru”** string tipindeki değişkendir.

Class yapısını da aşağıdaki resimdeki gibi hazırladım.

```
public Bilgisayar(int skor, ArrayList<Sporcu> A_oyuncuDeck) {
    super("Bilgisayar#0000", "Bilgisayar", skor);
    this.A_oyuncuDeck = A_oyuncuDeck;
}

public Bilgisayar() {
    super();
}
```

Class yapısını hazırladıktan sonra classın özelliklerini metotlar ile tanımlıyoruz sırasıyla şu şekilde;
kartSec,kartCikar,neKadarKaldi,kartListesi,desteGonder olmak üzere metotları yazıyorum.

Metotların çoğunu ana classdaki (Oyuncu.java) özelliklerden yararlanmak için **Override** ediyorum.

```
@Override
public Sporcu kartSec(int a) {
    int randomSayi = (int) (Math.random() * A_oyuncuDeck.size());
    int cikis = 1, sayacF = 0, sayacB = 0;
    for (int i = 0; i < A_oyuncuDeck.size(); i++) {
        if (A_oyuncuDeck.get(i).SporTuru().equals("Futbolcu")) {
            sayacF++;
        } else if (A_oyuncuDeck.get(i).SporTuru().equals("Basketbolcu")) {
            sayacB++;
        }
    }
    if (sayacF == 0 || sayacB == 0) {
        return A_oyuncuDeck.get(randomSayi);
    } else {
        while (cikis != 0) {
            if (A_oyuncuDeck.get(randomSayi).SporTuru().equals(temp_SporTuru)) {
                randomSayi = (int) (Math.random() * A_oyuncuDeck.size());
            } else if (!A_oyuncuDeck.get(randomSayi).SporTuru().equals(temp_SporTuru) && a != 5) {
                temp_SporTuru = A_oyuncuDeck.get(randomSayi).SporTuru();
                cikis = 0;
            } else if (a == 5) {//Bu kontrol arayüz tasarimda spor turunu kontrol etmek icin yapiliyor
                cikis = 0;
            }
        }
    }
    return A_oyuncuDeck.get(randomSayi);
}
```

```
@Override
public String kartCikar(Sporcu kart) {
    A_oyuncuDeck.remove(kart);
    return "";
}

@Override
public int neKadarKaldi() {

    return A_oyuncuDeck.size();
}

@Override
public String kartListesi() {
    System.out.println(getOyuncuID() + " OYUNCUSUNUN DESTESI");
    System.out.println("*****************************************************");
    for (int i = 0; i < A_oyuncuDeck.size(); i++) {
        System.out.println(A_oyuncuDeck.get(i).sporcuPuaniGoster());
    }
    System.out.println("*****************************************************");
    return "";
}

@Override
public Sporcu desteGonder(int a) {
    return A_oyuncuDeck.get(a);
}
```

2.6.Kullanıcı.java :

Bu “Kullanıcı.java” classını ilk olarak “Oyuncu.java” kalıtım almasını sağlıyorum tanımladığım ortak özellikleri kullanması için.

public class Kullanıcı **extends** Oyuncu{

}

Bunu tanımladıktan yapısını şu şekilde tanımlıyorum;

```
public Kullanıcı(String oyuncuID, String oyuncuAdi, int skor, ArrayList<Sporcu> B_oyuncuDeck) {
    super(oyuncuID, oyuncuAdi, skor);
    this.B_oyuncuDeck = B_oyuncuDeck;
}

public Kullanıcı(int skor) {
    super("DefaultOyuncuID#0000", "DefaultOyuncuID", skor);
}

public Kullanıcı() {
    super();
}
```

Class yapısını tanımladıktan sonra sırasıyla metotları tanımlıyorum;**kartSec,kartCikar,neKadarKaldi,kartListesi,destekGonder** olmak üzere metotları oluşturdum.

Metotların çoğunu ana classdaki (Oyuncu.java) özelliklerden yararlanmak için **Override** ediyorum.

```
@Override
public Sporcu kartSec(int a) {
    System.out.println("Kacinci karti atmak istiyorsunuz [1-" + B_oyuncuDeck.size() + "]?");
    int kullaniciClass = 0;
    while (true) {
        kullaniciClass = a;
        if (kullaniciClass <= B_oyuncuDeck.size()) {
            break;
        } else {
            System.out.println("*****************************************************");
            System.out.println("Hatali deger girdiniz tekrar deneyiniz !");
            System.out.println("*****************************************************");
        }
    }
    return B_oyuncuDeck.get(kullaniciClass);
}

@Override
public String kartCikar(Sporcu kart) {
    B_oyuncuDeck.remove(kart);
    return "";
}

@Override
public int neKadarKaldi() {

    return B_oyuncuDeck.size();
}

@Override
public String kartListesi() {
    System.out.println(getOyuncuID() + " OYUNCUSUNUN DESTESI");
    System.out.println("*****************************************************");
    for (int i = 0; i < B_oyuncuDeck.size(); i++) {
        System.out.println(B_oyuncuDeck.get(i).sporcuPuaniGoster());
    }
    System.out.println("*****************************************************");
    return "";
}

@Override
public Sporcu desteGonder(int a) {
    return B_oyuncuDeck.get(a);
}
```

2.7.Test.java :

"Test.java" classı bu programın kalbi gibidir. Tüm classlar burada işlenir ve yönlendirilir. Sporcu kartları burada tanımlanır basketbolcu ve futbolcu olmak üzere. Özel tanımlanan listelere atılır. Sonra bu listeler karıştırılarak Bilgisayar ve Kullanıcıya tek tek dağıtılır . Dağıtılma işlemi bittikten sonra "AnaMenu.java"ya yani programın giriş ekranına yollanır Bilgisayar ve Kullanıcı classlarından oluşan "A_oyuncu" ve "B_oyuncu" sırasıyla Bilgisayar ve Kullanıcı tipindeki değişkenler şu şekilde ;

```
String kullaniciAd = "";
Bilgisayar A_oyuncu = new Bilgisayar(A_Skor, A_oyuncuDeck);
Kullanıcı B_oyuncu = new Kullanıcı(kullaniciAd + "#1346", kullaniciAd, B_Skor, B_oyuncuDeck);

System.out.println(A_oyuncu.kartListesi());
System.out.println(B_oyuncu.kartListesi());

AnaMenu menu = new AnaMenu(A_oyuncu, B_oyuncu);
menu.setVisible(true);
```

2.8.AnaMenu.java :

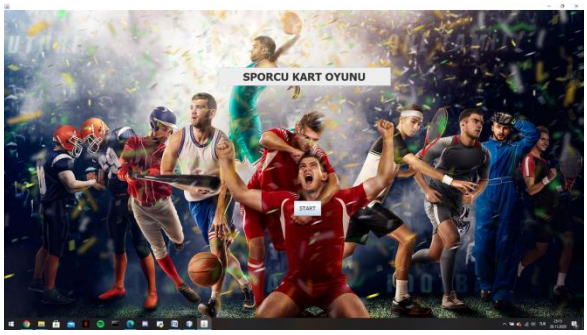
Bu class arayüz için oluşturuldu.Arayüz de Giriş ekranını oluşturuyor.

İlk olarak classa özel değişkenlerimi tanıttım sırasıyla; "bilgisayar","kullanıcı" olmak üzere bunların tipi Bilgisayar ve Kullanıcı 'dır sırasıyla.

"AnaMenu.java" class yapısı şu şekildedir;

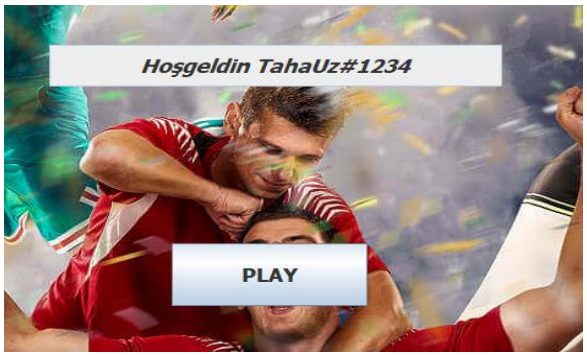
```
public AnaMenu() {
    initComponents();

}

public AnaMenu(Bilgisayar bilgisayar, Kullanıcı player) {
    initComponents();
    text2.setVisible(false);
    text3.setVisible(false);
    text4.setVisible(false);
    jButton2.setVisible(false);
    this.bilgisayar = bilgisayar;
    this.player = player;
}
```

Buradaki text1,text2,text3,text4,jButton2 olmak üzere bu değişkenler arayüzde kullanacağı için çeşitli amaçlara hizmet etmek üzere ilk olarak "Menu" çağırıldığında görünmemesi gereken değişkenlerdir.Bu değişkenlerden bazıları ilk olarak ;



Burada görülmemeleri için görünmez yapılmıştır karmaşa yaşanmaması için. Bu ekranda kullanıcı "Start" butonuna bastıktan sonra ;

bu ekranla karşılaşır.Burada kullanıcıdan "player Tag" 'ini (Kullanıcı ID) aldıktan sonra ikinci bir ekran ile karşılaşır ;



Daha sonra bu ekranda "Play" butonuna basarak "OyunAlani.java" classına geçişini yapar ve oyun başlar.

2.9.OyunAlani.java :

Bu class arayüz için oluşturuldu . Arayüz de Oyun Alanı ekranını oluşturuyor.

İlk olarak classa özel değişkenlerimi tanıttım sırasıyla;"bilgisayar","player","bilgisayarDeck", "playerDeck" olmak üzere değişkenler atadım.Veri tipleri sırayla; "Bilgisayar","Kullanıcı","ArrayList<Sporcu>","ArrayList<Sporcu>" şeklindedir.

"OyunAlani.java" class yapısı şu şekildedir;

```
public OyunAlani() {
    initComponents();

}

public OyunAlani(Bilgisayar bilgisayar, Kullanıcı player) {
    initComponents();
    this.bilgisayar = bilgisayar;
    this.player = player;
    this.kullaniciAdPanel.setText(player.getOyuncuID());
    Kazanan.setVisible(false);
    ExitButton.setVisible(false);
    buton1.setVisible(false);
    buton2.setVisible(false);
    buton3.setVisible(false);
    buton4.setVisible(false);
    buton5.setVisible(false);
    buton6.setVisible(false);
    buton7.setVisible(false);
    buton8.setVisible(false);
    Kurallar.setText("1-)Oyuna Baslamak Icın Ilk Olarak Kartlari Diz Butonuna BASINIZ !"
            +"\n2-)Oyundaki Hamle Sistemi Ardışık Olarak Hareket Eder ve Ilk Hamle Futbolcu 'dur .");
}
```

Bu classdaki metotlar sırasıyla ; **pozisyonSec,KimKazanir,sonKontrol,kartlariAreNayaYaz** ve textler,butonlar olmak üzere bu şekildedir.Bu metotları açıklamak gerekirse;

***pozisyonSec();**

Bu metot oyunun rasgele pozisyon belirlemesinde yardımcı olur ."Math.random()" yardımıyla oluşturulan rasgele bir sayı belirlenen koşullara göre rasgele pozisyon belirler bir sonraki resimdeki gibi .

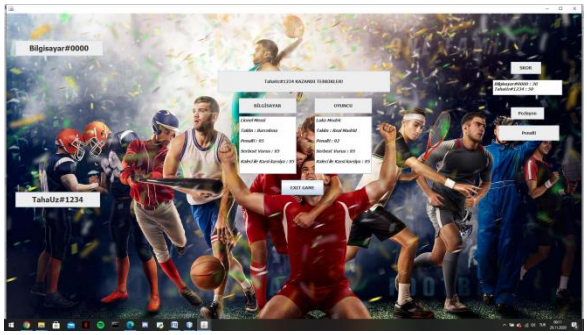
```
public String pozisyonSec(String HangiSpor) {

    int randomPozisyon = (int) (Math.random() * 3);

    if (HangiSpor.equals("Futbolcu")) {
        if (randomPozisyon == 0) {
            return "penalti";
        } else if (randomPozisyon == 1) {
            return "serbestVurus";
        } else if (randomPozisyon == 2) {
            return "kaleciKarsiKarsiya";
        }
    }

    if (HangiSpor.equals("Basketbolcu")) {
        if (randomPozisyon == 0) {
            return "ikilik";
        } else if (randomPozisyon == 1) {
            return "ucluk";
        } else if (randomPozisyon == 2) {
            return "serbestAtis";
        }
    }

    return "pozisyon";
}
```

***KimKazanir();**

Bu metotda kartları karşılaştırır.Hangi kartın kazanıcağını belirler.Bunun yanı sıra ekrana(arayüze) skor ve pozisyon bilgisini de verir.

***sonKontrol();**

Oyundaki son hamlenin durumunu kontrol eder.Eğer en son kart berabere kalırsa diğer kontrollerden özel olarak oyun , kazanan bir oyuncu olana kadar rasgele pozisyon belirler ve ona göre kimin kazandığını söyler. Eğer tüm pozisyonlar eşit ise berabere kalır ve kimseye puan verilmez. Bunların yanı sıra skorları ve pozisyonları arayüze verir ve kazananı belirleyerek skor tablosunun son halini gösterir.



***kartlariArenayaYaz();**

Kullanıcının elindeki tüm kartları oyun alanına dizer.

"OyunAlani.java" classının içindeki kartları oyun alanında kullanmak için "USE CARD" butonu vardır.Bu butonlarda her zaman ilk olarak oyunun kuralına göre ilk futbolcu sonra basketbolcu hamleleri kuralına uygun bu atılan kart onun kontrolu yapılır. Bu kontrolden geçtikten sonra bilgisayara rasgele kart çektirilir bilgisayarın destesinden kart çekildikten sonra arenadaki bilgisayara ayarlanan bölüme kart bilgileri yazılır.Bu bittikten sonra kullanıcının da tıkladığı kartın kart bilgileri arenadaki kullanıcıya ayrılan alana yazdırılır. Bu işlemler bittikten sonra iki koşul durumuna girerler ;

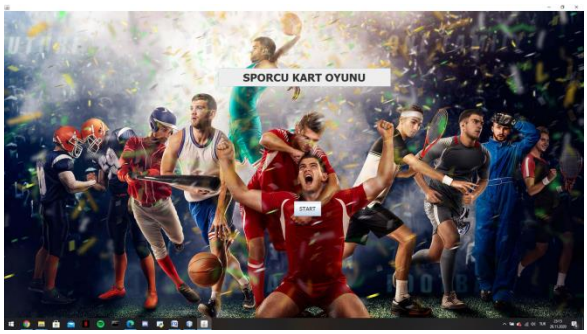
```
if (player.neKadarKaldi() == 1 && bilgisayar.neKadarKaldi() == 1) {
    sonKontrol(sportPerson, playerDeck.get(1));
    bilgisayar.kartCikar(sportPerson);
    player.kartCikar(playerDeck.get(1));
    buton2.setVisible(false);
    textpanel2.setVisible(false);
    jScrollPane2.setVisible(false);
}

if (player.neKadarKaldi() > 1 && bilgisayar.neKadarKaldi() > 1) {
    if (KimKazanir(sportPerson, playerDeck.get(1)) != true) {
        bilgisayar.kartCikar(sportPerson);
        player.kartCikar(playerDeck.get(1));
        buton2.setVisible(false);
        textpanel2.setVisible(false);
        jScrollPane2.setVisible(false);
    }

}
```

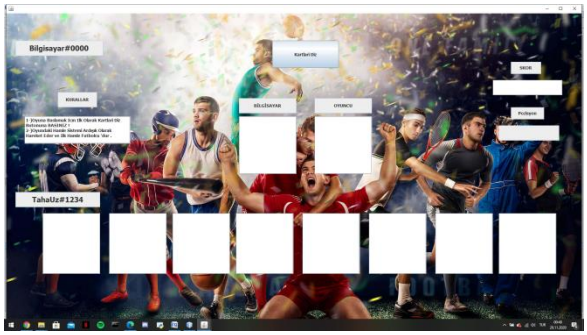
Bu koşullardan geçildikten sonra iki taraftan birisi kazanırsa kartlar yok olur eğer berabere kalındıysa kartlar aynı şekilde kalır ve tekrar kullanıma açık olur eğer son kart berabere kalırsa orada özel durum mevcuttur ona göre bir tutum sergilenir.

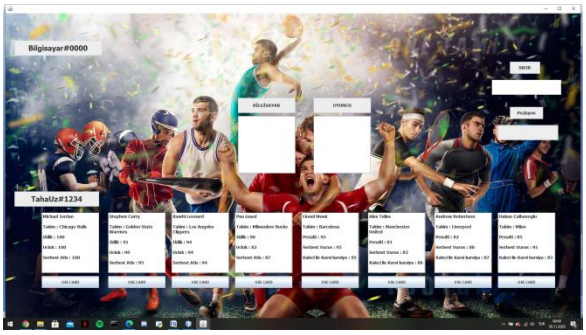
3.KULLANILAN FONKSİYONLAR:

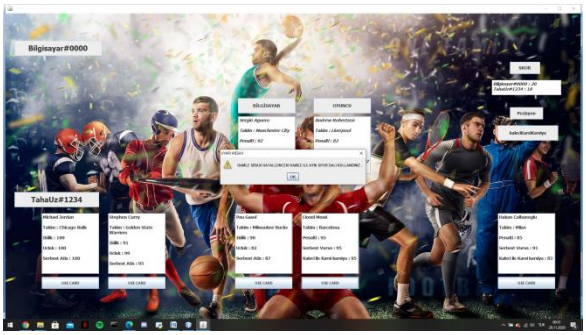
setSporcuIsim(),setSporcuTakim,sporcuPuaniGoster,getSporcuIsim,getSporcuTakim,getPenalti,getSerbestAtis,getKaleciKarsiKarsiya,getIkilik,getUcluk,getSporTuru,setOyuncuID,setOyuncuAdi,setSkor,getOyuncuID,getOyuncuAdi,getSkor,kartSec,kartListesi,skorArttir,kartCikar,desteGonder,neKadarKaldi,SkorGoster,pozisyonSec,KimKazanir,sonKontrol,kartlariArenayaYaz gibi benim yazdığım fonksiyonlar vardır . Bunların yanı sıra benimde kullandığım hazır fonksiyonlar vardır sırasıyla; **ArrayListlerin** ve **Arraylerin** yan fonksiyonları örneğin ; **".add",".get",".length", ".size"** gibi fonksiyonlar. **Stringlerin ".equals"** fonksiyonları. **Math.h** kütüphanesinin **".random()"** fonksiyonu.**Swing** kütüphanesinin **butonActionPerformer**,**"setText","setVisible", ".showMessageDialog"** gibi fonksiyonlarıda hazır olarak kullandım.

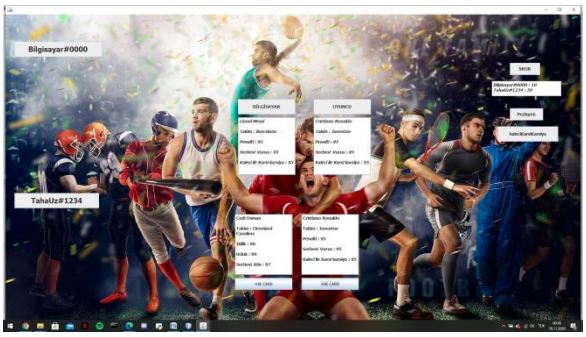
4.DENEYSEL SONUÇLAR:

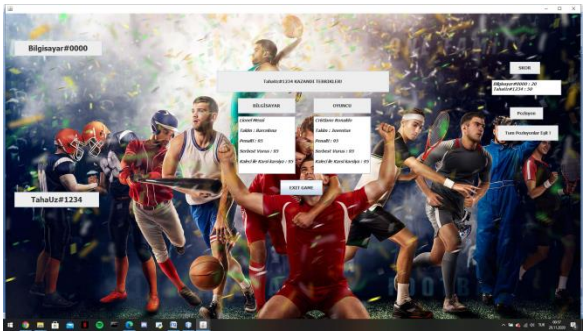












6.KARŞILAŞILAN SIKINTILAR ve ÇÖZÜMLERİ:

1.Son Kontrol Skoru 2 Kat Skor Arttırması

"USE CARD" butonlarına tıklandığında onlara bağlı olan kartın özellikleri ile işlem yapılır .Bu işlem yapılırken son 2 kart kaldığında her iki oyuncuda da oyun kimin kazanıcağını hesaplayıp kazananı bulduktan sonra puan verirken 10 puan vermesi gerekirken 20 puan veriyordu ve erkenden skor tablosu çıkarıyordu .Sıkıntı son 2 kart kaldığında ilk koşula girip kartı desteden çıkarınca benim son kart koşuluna da girince 2 kez işlem yapıyordu normalde 1 kez yapması gerekirken bu sorunu da koşulların sırasını ve birkaç kontrol mekanizması daha ekleyerek sorunu çözdüm aşağıda resimdeki gibi.

```
if (player.neKadarKaldi() == 1 && bilgisayar.neKadarKaldi() == 1) {
    sonKontrol(sportPerson, playerDeck.get(1));
    bilgisayar.kartCikar(sportPerson);
    player.kartCikar(playerDeck.get(1));
    buton2.setVisible(false);
    textpanel2.setVisible(false);
    jScrollPane2.setVisible(false);
}

if (player.neKadarKaldi() > 1 && bilgisayar.neKadarKaldi() > 1) {
    if (KimKazanir(sportPerson, playerDeck.get(1)) != true) {
        bilgisayar.kartCikar(sportPerson);
        player.kartCikar(playerDeck.get(1));
        buton2.setVisible(false);
        textpanel2.setVisible(false);
        jScrollPane2.setVisible(false);
    }

}
```

2.JFRAME ile Oluşturduğum Classların Yapısını Farklı Zannedip Parametre Yollayamamak

JFrame ile oluşturduğum her class a parametre yollayamıyordum farklı çalışıyor zannediyordum buradaki class yapısının ama diğerleri ile farklı yokmuş ve altta Swing oluşturduğu için arayüzü parametre olmadan da çağırmak istiyormuş bu classı .

Bu durumda classın bir de boş parametreli versiyonunu oluşturarak bu durumu çözdüm aşağıda resimdeki gibi.

```
public OyunAlani() {
    initComponents();

}
```

7.SONUÇ:

Bu proje sayesinde Java dilinde nesneye yönelik programlamayı daha detaylı bir biçimde kodlamayı ve bir uygulamada arayüz tasarımının nasıl yapıldığını , Swing arayüzünün nasıl kullanıldığını öğrendiğimi düşünüyorum .

8.KAYNAKÇA:

1.Intelij Hakkında Bilgi

https://tr.wikipedia.org/wiki/IntelliJ_IDEA

2.Java'nın Tarihi ve Nasıl Bir Programlama Dili Olduğuna Dair Bilgi

https://tr.wikipedia.org/wiki/Java_(programlama_dili)

3.Java ve Nesneye Yönelik Programlama ile İlgili Aldığım Kurs ve Bu Kursda Encapsulation, Inheritance, Polymorphism, Abstraction Yapılarıyla ilgili Edindiğim Bilgilerin Videoları

(Tim Buchalka-Java Programing Masterclass)

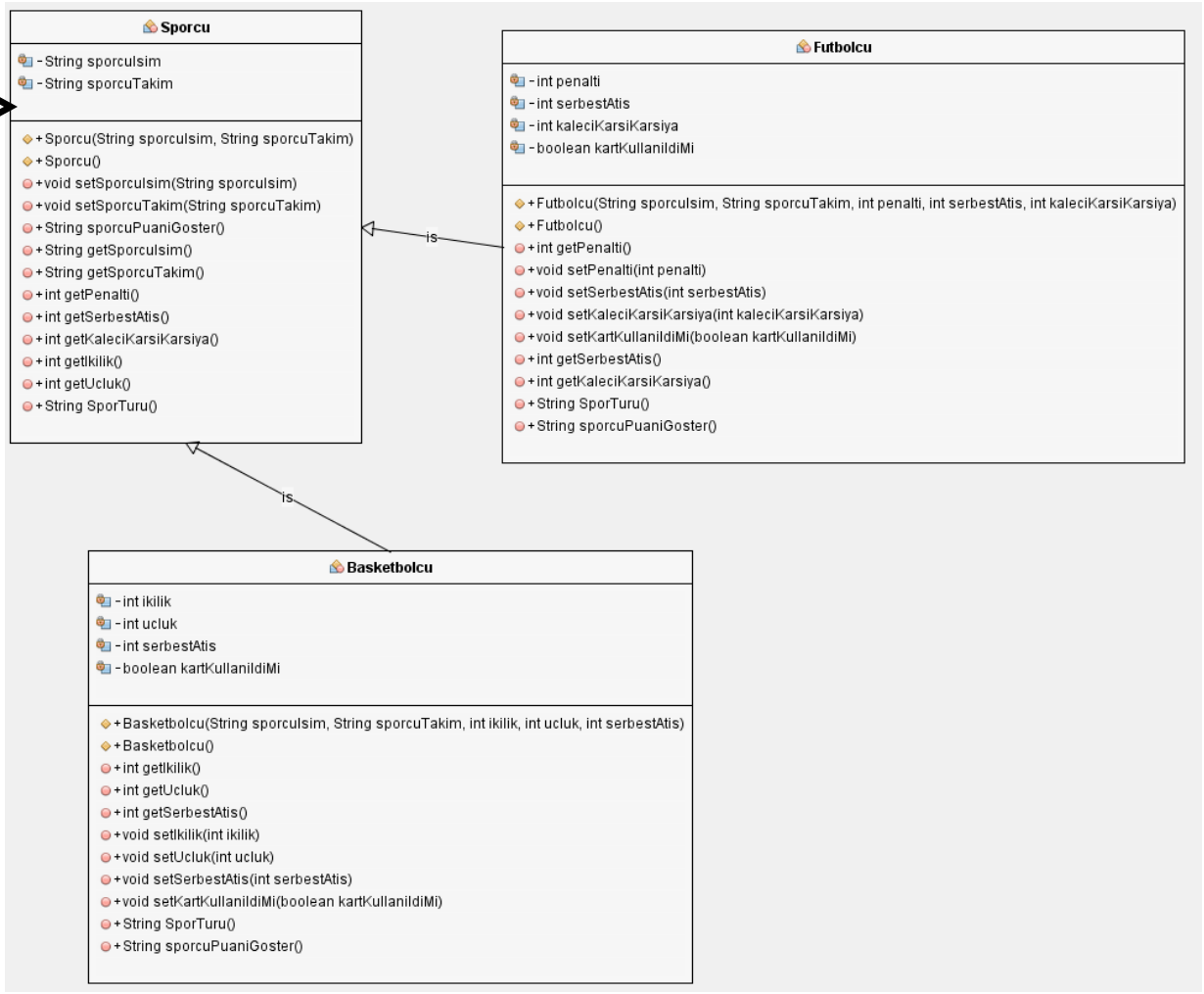
https://www.udemy.com/share/101WdqBksecl1RQHo=/

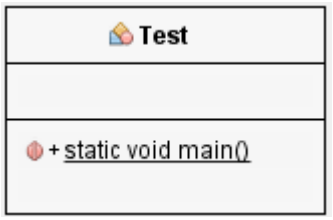
4.Java Swing Arayüzünü Öğrenmek için Kullandığım Video Eğitim Serisi

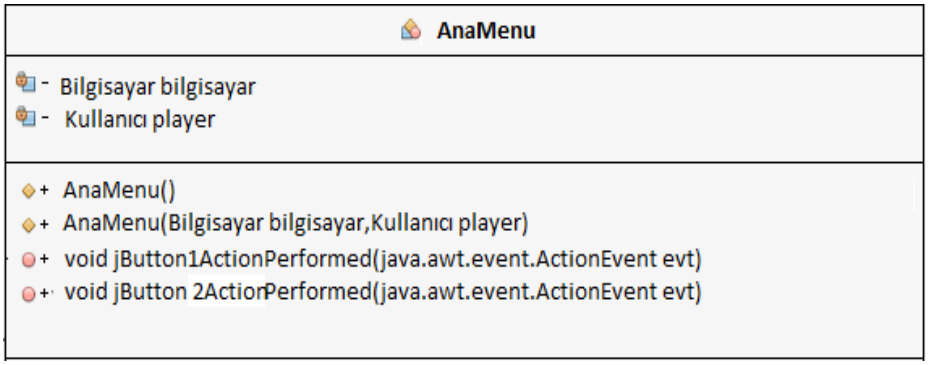
https://www.youtube.com/watch?v=QW3iL3zFJSU&list=PL4yfBYtaNjbRRGLQnrTPU2eiAB5rgi9lx

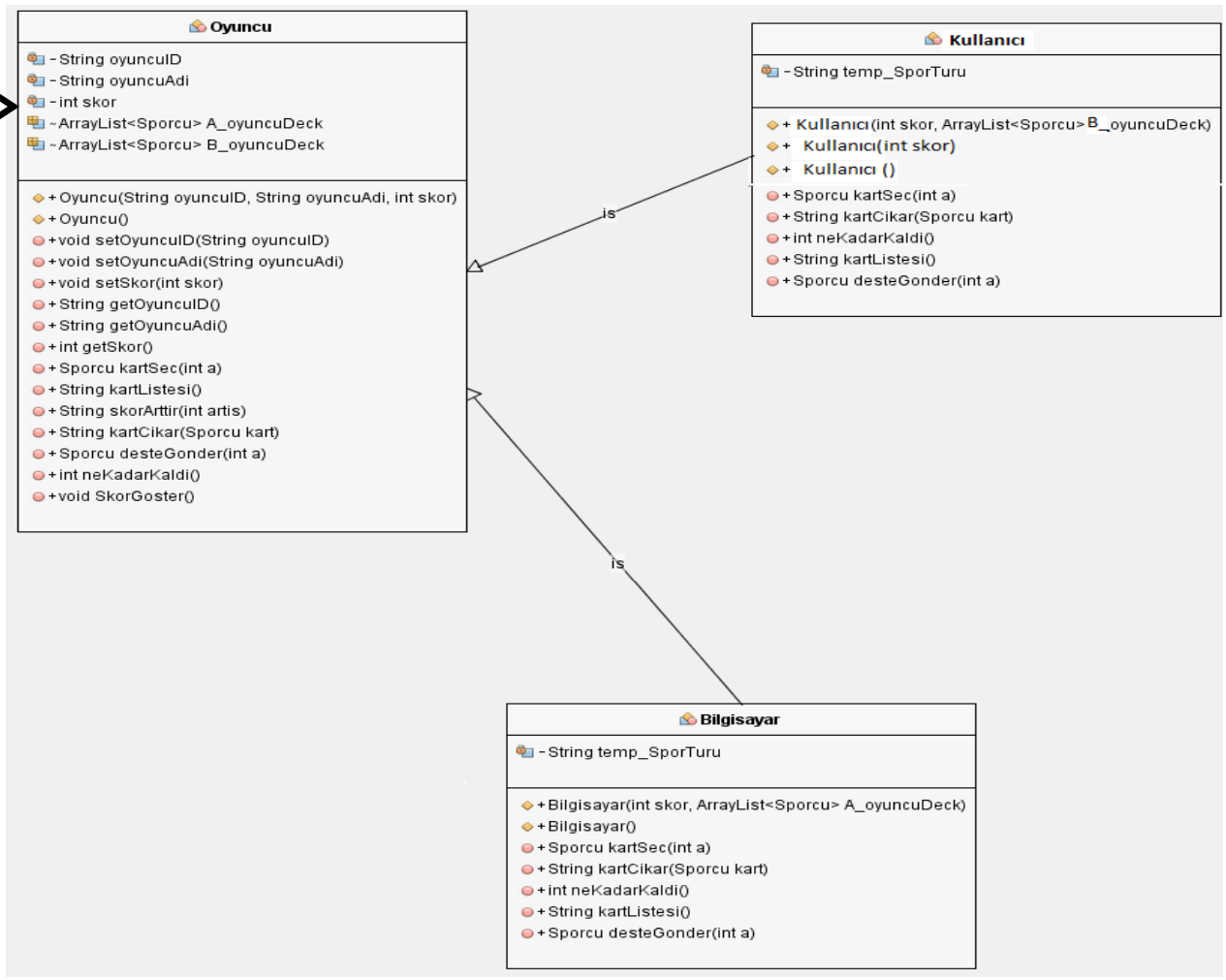
5.UML Sınıf Diyagramı Hakkında Bilgi Edindiğim Kaynaklar

https://www.youtube.com/watch?v=-DzRBYl6Pwk


Sporcu
- String sporculsim
- String sporcuTakim
+ Sporcu(String sporculsim, String sporcuTakim)
+ Sporcu()
+ void setSporculsim(String sporculsim)
+ void setSporcuTakim(String sporcuTakim)
+ String sporcuPuaniGoster()
+ String getSporculsim()
+ String getSporcuTakim()
+ int getPenalti()
+ int getSerbestAtis()
+ int getKaleciKarsiKarsiya()
+ int getIkilik()
+ int getUcluk()
+ String SporTuru()
Futbolcu
- int penalti
- int serbestAtis
- int kaleciKarsiKarsiya
- boolean kartKullanildiMi
+ Futbolcu(String sporculsim, String sporcuTakim, int penalti, int serbestAtis, int kaleciKarsiKarsiya)
+ Futbolcu()
+ int getPenalti()
+ void setPenalti(int penalti)
+ void setSerbestAtis(int serbestAtis)
+ void setKaleciKarsiKarsiya(int kaleciKarsiKarsiya)
+ void setKartKullanildiMi(boolean kartKullanildiMi)
+ int getSerbestAtis()
+ int getKaleciKarsiKarsiya()
+ String SporTuru()
+ String sporcuPuaniGoster()
is
is
Basketbolcu
- int ikilik
- int ucluk
- int serbestAtis
- boolean kartKullanildiMi
+ Basketbolcu(String sporculsim, String sporcuTakim, int ikilik, int ucluk, int serbestAtis)
+ Basketbolcu()
+ int getIkilik()
+ int getUcluk()
+ int getSerbestAtis()
+ void setIkilik(int ikilik)
+ void setUcluk(int ucluk)
+ void setSerbestAtis(int serbestAtis)
+ void setKartKullanildiMi(boolean kartKullanildiMi)
+ String SporTuru()
+ String sporcuPuaniGoster()


Test
+ static void main()


AnaMenu
- Bilgisayar bilgisayar
- Kullanıcı player
+ AnaMenu()
+ AnaMenu(Bilgisayar bilgisayar,Kullanıcı player)
+ void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
+ void jButton 2ActionPerformed(java.awt.event.ActionEvent evt)


Oyuncu
- String oyuncuID
- String oyuncuAdi
- int skor
~ArrayList<Sporcu> A_oyuncuDeck
~ArrayList<Sporcu> B_oyuncuDeck
+ Oyuncu(String oyuncuID, String oyuncuAdi, int skor)
+ Oyuncu()
+ void setOyuncuID(String oyuncuID)
+ void setOyuncuAdi(String oyuncuAdi)
+ void setSkor(int skor)
+ String getOyuncuID()
+ String getOyuncuAdi()
+ int getSkor()
+ Sporcu kartSec(int a)
+ String kartListesi()
+ String skorArttir(int artis)
+ String kartCikar(Sporcu kart)
+ Sporcu desteGonder(int a)
+ int neKadarKaldi()
+ void SkorGoster()
Kullanıcı
- String temp_SporTuru
+ Kullanıcı(int skor, ArrayList<Sporcu> B_oyuncuDeck)
+ Kullanıcı(int skor)
+ Kullanıcı ()
+ Sporcu kartSec(int a)
+ String kartCikar(Sporcu kart)
+ int neKadarKaldi()
+ String kartListesi()
+ Sporcu desteGonder(int a)
is
is
Bilgisayar
- String temp_SporTuru
+ Bilgisayar(int skor, ArrayList<Sporcu> A_oyuncuDeck)
+ Bilgisayar()
+ Sporcu kartSec(int a)
+ String kartCikar(Sporcu kart)
+ int neKadarKaldi()
+ String kartListesi()
+ Sporcu desteGonder(int a)


OyunAlani
- Bilgisayar bilgisayar
- Kullanıcı player
~ArrayList<Sporcu> A_oyuncuDeck
~ArrayList<Sporcu> B_oyuncuDeck
+ OyunAlani()
+ OyunAlani(Bilgisayar bilgisayar,Kullanıcı player)
+ String pozisyonSec(String HangiSpor)
+ Boolean KimKazanir(Sporcu A,Sporcu B)
+ Void sonKontrol(Sporcu A,Sporcu B)
+ Void kartlariArenayaYaz()