PROGRAMLAMA LABORATUVARI 1

2. PROJE

Muhammet Ali BAKINÇ
Abdulhalik SARIÇİÇEK
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi

Özet

Bu doküman Programlama Laboratuvarı 1 dersi 2. Projesi için çözümümüzü açıklamaya yönelik oluşturulmuştur. Dokümanda projenin tanımı, çözüme yönelik yapılan araştırmalar, kullanılan yöntemler, proje hazırlanırken kullanılan geliştirme ortamı ve kod bilgisi gibi programın oluşumunu açıklayan başlıklara yer verilmiştir. Doküman sonunda projemizi hazırlarken kullandığımız kaynaklar ve proje derlenirken dikkat edilmesi gereken hususlar bulunmaktadır.

1. Proje Tanımı

Projede bizden istenen, C++, C# veya Java dillerinden biri kullanarak, bir oyuncunun otomatik oyuncuyla (bilgisayar), yarışabileceği basit bir sporcu kart oyunu yaratmamızdır.

Tasarlayacağımız oyunda, toplamda 16 sporcu kartı olacak ve her bir kullanıcıya ilk başta rastgele olarak 4 basketbolcu ve 4 futbolcu olmak üzere 8 er sporcu kartı dağıtılacaktır. Kullanıcı ve bilgisayar kendilerine dağıtılan 8 karttan birini seçerek ortaya koyacaktır. İki taraf kartları ortaya açık bir şekilde koyacak ve yüksek özellik puanına sahip olan taraf 10 puan kazanacaktır. Daha sonra kullanıcı ve bilgisayar atılan spordan farklı bir sporun kartıyla devam edecektir. Kart puanları eşit olması durumunda kartlar geri alınacak ve devam edilecektir. Eldeki kartlar bitene kadar oyun devam edecektir. Oyun bittiğinde en yüksek puana sahip oyuncu kazanacaktır. Bu oyunu nesneye yönelik programlama yöntemini kullanarak yapmamız beklenmektedir.

Oyun kurallarına ek olarak proje gereği olan isterler bulunmaktadır. Oyunu ve oyuncuların ellerinde bulunan kartlar görülebilecek ve takip edilebilecek bir ara yüz tasarlamamız beklenmektedir.

Oyun işleyişi aşağıdaki gibi olmalıdır:

- Masada her oyuncunun önünde 8 adet sporcu kartı bulunmalıdır.
- Oyuncu istediği kartı bilgisayarla birlikte ortaya atar.
- Sistem tarafından spor dalına göre belirlenen üç özellikten biri random seçilir.
- Kartların belirlenen özellik değerlerine göre yüksek kartın sahibi 10 puan kazanır.
- Bilgisayar seçimi oyuncun attığı sporla aynı olmak üzere random olarak gerçekleştirir.
- Özellik değerlerinin eşit olması halinde kartlar geri alınır ve diğer spor dalından kart atılması beklenir.
- Bir spor dalına ait kartların bitmesi halinde sürekli diğer spor kartı oyun sonuna kadar atılmaya devam eder.
- Oyuna sürülen kartlar bir daha kullanılamayacak.

(Şekil 1: Bilgisayar oyuncusunun kapalı kartlarının görünümü)

(Şekil 2: Ortaya atılan kartlarının görünümü)

- Şekil 2'de gösterildiği üzere, seçilen özellikler penaltı veya karşı karşıya olması halinde Lewandowski kazanacak, serbest vuruş olması halinde puan verilmeyip kartlar geri alınacaktır.

Projede belirli sınıfların ve bu sınıflarda belirli özelliklerin kullanılması zorunlu kılınmıştır. Nesneye yönelik programlama yönteminde geçerli olan *Encapsulation, Inheritance, Polymorphism, Abstraction* yapılarının gerekli olanların kullanılması da beklenmektedir. Projede oluşturmamız gereken sınıflar ve özellikleri kabaca bölüm 1.1. de anlatılmıştır.

1.1. Oluşturulması Gereken Sınıflar

Aşağıda belirtilen tüm sınıflarda ortak olarak yapıcı (*constructor*) metotları, sınıflardaki tüm özellikler için *get, set* metotları tanımlanmalıdır.

Sporcu sınıfı

Kartların toplam puanını göstermek için *sporcuPuaniGoster* metodu yazılmalıdır. Sınıfta *sporcuIsim, sporcuTakim* özellikleri tutulmalıdır. Sınıf hiçbir sınıftan kalıtım almamaktadır.

Futbolcu sınıfı

Sporcu sınıfından kalıtım alacaklardır. Futbolcu sınıfında bulunan *futbolcuIsim* ve *futbolcuTakim* özelliklerine atama yapmak için *super* metodu kullanılacaktır. Futbolcu özellikleri olan *penalti, serbestVurus, kaleciKarsiKarsiya* özellikleri bu sınıfta tutulacaktır ve Sporcu sınıfında bulunan *sporcuPuaniGoster* metodu *override* edilerek her bir futbolcu için özelleştirilecektir. Ayrıca *boolean* veri tipinde *kartKullanildiMi* bilgisi tutulmalıdır. Kullanılan kartların oyunda bir daha kullanılmasını engellemek için bu veri tipinden yararlanılacaktır. Parametreli ve parametresiz iki *constructor* bulunmalıdır. Değişkenler için *getter* ve *setter* metotları oluşturulmalıdır.

Basketbolcu sınıfı

Sporcu sınıfından kalıtım alacaklardır. Basketbolcu sınıfında bulunan *basketbolcuIsim* ve *basketbolcuTakim* özelliklerine atama yapmak için *super* metodu kullanılacaktır. Basketbolcu özellikleri olan *ucluk, ikilik, serbestAtis* özellikleri bu sınıfta tutulacaktır ve Sporcu sınıfında bulunan *sporcuPuaniGoster* metodu *override* edilerek her bir futbolcu için özelleştirilecektir. Ayrıca *boolean* veri tipinde *kartKullanildiMi* bilgisi tutulmalıdır. Kullanılan kartların oyunda bir daha kullanılmasını engellemek için bu veri tipinden yararlanılacaktır. Parametreli ve parametresiz iki *constructor* bulunmalıdır. Değişkenler için *getter* ve *setter* metotları oluşturulmalıdır.

Oyuncu sınıfı

Bilgisayar ya da kullanıcı olmak üzere oyunu oynayan iki oyuncu olacaktır. Aynı özellikleri temsil etmek için *Oyuncu* temel

sınıfı oluşturulacaktır. Bu sınıfta *oyuncuID, oyuncuAdi* ve *skor* özellikleri bulunmalıdır. *kartListesi* özelliği ile oyuncuların elinde bulunan kartlar listede tutulacaktır. *skorGoster* fonksiyonu ile oyuncuların skorları gösterilecektir. *kartSec* fonksiyonu yazılmalı, bu fonksiyonun bilgisayar ve kullanıcı için farklı durumlarda çalışacağı unutulmamalıdır.

Bilgisayar sınıfı

Oyuncu sınıfından kalıtım alacaktır. Oyuncu sınıfında bulunan *oyuncuID, oyuncuAdi* ve *skor* özelliklerine atama yapmak için *super* kullanılacaktır. Oyuncu sınıfında bulunan *kartSec* metodu *override* edilecektir. Bilgisayar rastgele olarak aldığı kartlar arasından yine rastgele kart seçerek ortaya koyacaktır.

Oyuncu sınıfı

Oyuncu sınıfından kalıtım alacaktır. Oyuncu sınıfında bulunan *oyuncuID, oyuncuAdi* ve *skor* özelliklerine atama yapmak için *super* kullanılacaktır. Oyuncu sınıfında bulunan *kartSec* metodu *override* edilecektir. Kullanıcı rastgele olarak aldığı kartlar arasından kendi istediği kartı seçerek ortaya koyacaktır.

2. Araştırmalar ve Yöntem

Projeye proje tanımında anlatılan sınıfları oluşturmakla başladık. Projede belirtilen Sporcu ana sınıfını ve altındaki 2 farklı sınıfı tanımlarken, kullanmamız zorunlu olan birçok nesneye yönelik programlama özelliğini hali hazırda kullanmış oluyoruz. Her alt sınıf Sporcu sınıfından oluşacağı için *inheritance* özelliğini, sınıflardaki özelliklere dışarıdan *get, set* metotları ile ulaşım sağladığımızdan *encapsulation* özelliğini kullanmış oluyoruz. Sporcuların her alt sınıfını birlikte Sporcu tipinde bir *kartListesi* değişkeninde sakladığımız için *polymorphism* özelliğini kullanmış oluyoruz. Böylece projede kullanılması tavsiye edilen özellikler sınıfları oluştururken kullanılmış oluyor.

Projede nasıl yapılması gerektiğini düşündüren bir diğer nokta da oyunun nasıl yönetileceğine karar vermekti. Test sınıfını, kartları dağıtmak ve kendisinde oyuncuları bulundurmak gibi gerçek bir kart oyununda ne yapılıyorsa onu yaptık. Bu yöntem işlerimizi gayet kolaylaştırdı. Test sınıfında oyunu yönetmeye dair temel fonksiyonları tanımladık. Oyunu yönettiği için kullanıcı ara yüzünün de Test sınıfında çizilmesinin uygun olduğuna karar verdik.

Takıldığımız bir diğer nokta ise Test sınıfında ara yüzü nasıl yapacağımıza karar vermekti. Öncelikle çizim işlemleri için JavaFX kütüphanesi kullanmaya karar verdik. Daha sonra çizimleri farklı fonksiyonlar içinde yazdık. Böylece Test'te yaptığımız tüm çizimler dinamik ve yönetimi kolay oldu. Bu yöntemdeki sorun hazır bir editör kullanamayacak olmamızdı, çizilen her element için koordinatları ve boyutu kendimiz bulmamız gerekiyordu ancak bizim için sorun olmadı.

3. Geliştirme Ortamı

Projemizi Windows sistemde, *Intellij Idea* üzerinde geliştirip derledik. *Java* dilini ve çizim yapabilmek için *JavaFX* kütüphanesini kullandık.

4. Kod Bilgisi

4.1. UML Diyagramı ve Akış Şeması

Son sayfadadır.

4.2. Algoritma

Bu kısımda projenin genel algoritmasına açıklık getireceğiz.

Oyun başladığında kullanıcıya oyunu başlatması için bir başlama butonu karşılıyor. Oyunu yöneten ana sınıf Test sınıfı ve yönetime dair tüm fonksiyonlar (*kartOlustur, kartAt, ...*) ve oyuncu nesneleri de bu sınıfta bulunuyor. Test sınıfının kodlarına bakıldığı zaman oyunun ana mantığı ve kuralları kolayca görülebilir.

Önce main metodu içerisinde birer Basketbolcu ve Futbolcu tipinde diziler oluşturduk ve sınıflarındaki *oyuncuOlustur*

metodu ile oluşturulan oyuncuları atadık. Main dışındaki *kartDagit* fonksiyonuna bu dizileri göndererek kart dağıtma işlemini yaptık. Bu fonksiyonda random olarak önce kullanıcıya 4 futbolcu ve 4 basketbolcu kartı belirledik ve Sporcu sınıfında bulunan *kartListesi* 'ne gönderdik. Aynı şekilde geriye kalan kartları da bilgisayar için ayırdık. Main fonksiyonundaki son işlem olarak da *launch* komutuyla ara yüz çizimlerini başlatmış olduk.

JavaFX kütüphanesinin çizimleri başlattığı *start* metodunu override ettik. Bu metotta öncelikle ekranı oluşturduk. Çizim nesnelerini bir *group* nesnesinde topladık. Başlangıç butonu ile oyunu başlatmak için diğer çizim metotlarını butonun tıklama olaylarının yazıldığı *action* kısmının içerisine yazdık. Burada belirlediğimiz konumlara döngü ile önce bilgisayara sonra kullanıcıya ve ortaya atılması gereken kartlar için *kartOlustur* metodunu tetikledik. Aynı döngüler içinde kart çizimlerinin üzerinde görünecek olan kart resimlerini de *resimOlustur* metodu ile oluşturduk.

Bu metotların döndürdüğü nesneleri daha sonra kullanabilmek için listelere attık. Aynı zamanda gösterim için *group* nesnesine ekledik.

resimOlustur metodunda sporcu isimleriyle aynı oluşturduğumuz resim dosyalarını *images* klasöründen çağırarak her birine tıklama komutları ekledik. Bu komutlarda önce geri alma butonun aktif olması durumunu daha sonra atılan son kartların spor dallarına ve elde kalan kartların durumlarına göre değerlendirerek *kartAt* metodunu çalıştırdık. Aynı zamanda kartların atıldığını belirtmek için oluşturulan kart ve resim listeleri aracılığıyla group nesnesinden tıklanan kartların silinmesini sağladık.

kartOlustur metodu ile standart bir dikdörtgen kart çerçevesi çizdirerek kartların yerini belirttik.

kartAt metodunda başlangıçta atılan kartın sporunu belirleyerek *kartSirasi* değişkenine atadık. Böylece en son atılan kartın sporunu takip ettik. Daha sonra ekranın orta kısmında daha önce oluşturduğumuz konumlarda tıklanan resimleri çizdirdik. Kullanıcı ve bilgisayarın atılan kartlarını ortaya çizdikten sonra bunları *karsilastir* metoduna ilettik.

Bu metotta önce random olarak hangi sporcu özelliğinin karşılaştırılacağını her iki sporcu durum için kontrol ettirdik. Daha sonra bir sonraki aşamada seçilmesi gereken sporu bildirmek için *yaziOlustur* metodu aracılığıyla ekrana yazdırdık. Bunun için son atılan kartın sırasını, elde kalan veya biten sporcu kartlarının durumlarını ayrı ayrı ele alarak değerlendirdik. Devamında karşılaştırdığımız özellik değerlerini kıyaslayarak yüksek olan oyuncunun skorunu güncelledik. Eşit olması halinde ise bir geri al butonu ile kartların geri alınmasını sağladık. Bu butonun komutunda geri alınan kartların kullanılma durumlarını yeniden *false* yaparak tekrar kullanılmasını sağladık. Oyuncuların önünden silinen kart görsellerini yeniden çizerek ve ortadan atılan kart görsellerini silerek geri alma işlemini ekranda belirttik. Son olarak kartların bitmesi halinde kazananı veya berabere kalma durumunu ekrana yazdırmak için tekrar *yaziOlustur* metodunu kullandık. Ve *puanYazdir* metodunu tetikledik.

Bu metot *karsilastir* 'dan aldığı sonuç değerine göre puanı kazanan oyuncuyu ekrana yazmaktadır.

Son olarak *yaziOlustur* metodu ise aldığı metin değerini ekrana yazdırmaktadır.

4.3. İstatistik

Program kodu boşluksuz ve yorumsuz yaklaşık 675 satırdan oluşmaktadır. Kod düzenini sağlamak için yaklaşık 100 boş satır kullanılmıştır.
Kullandığımız kütüphaneler ve ne için kullandığımız kabaca aşağıdaki gibidir:
javafx
Çizim yapmak için kullandığımız kütüphanedir. Yaklaşık 17 alt kütüphanesi import edilmiştir.
*java.awt.**
Ekran boyutunu belirlemek için kullanıldı.

java.util
Random, arrayList gibi temel sınıflar için kullanıldı.

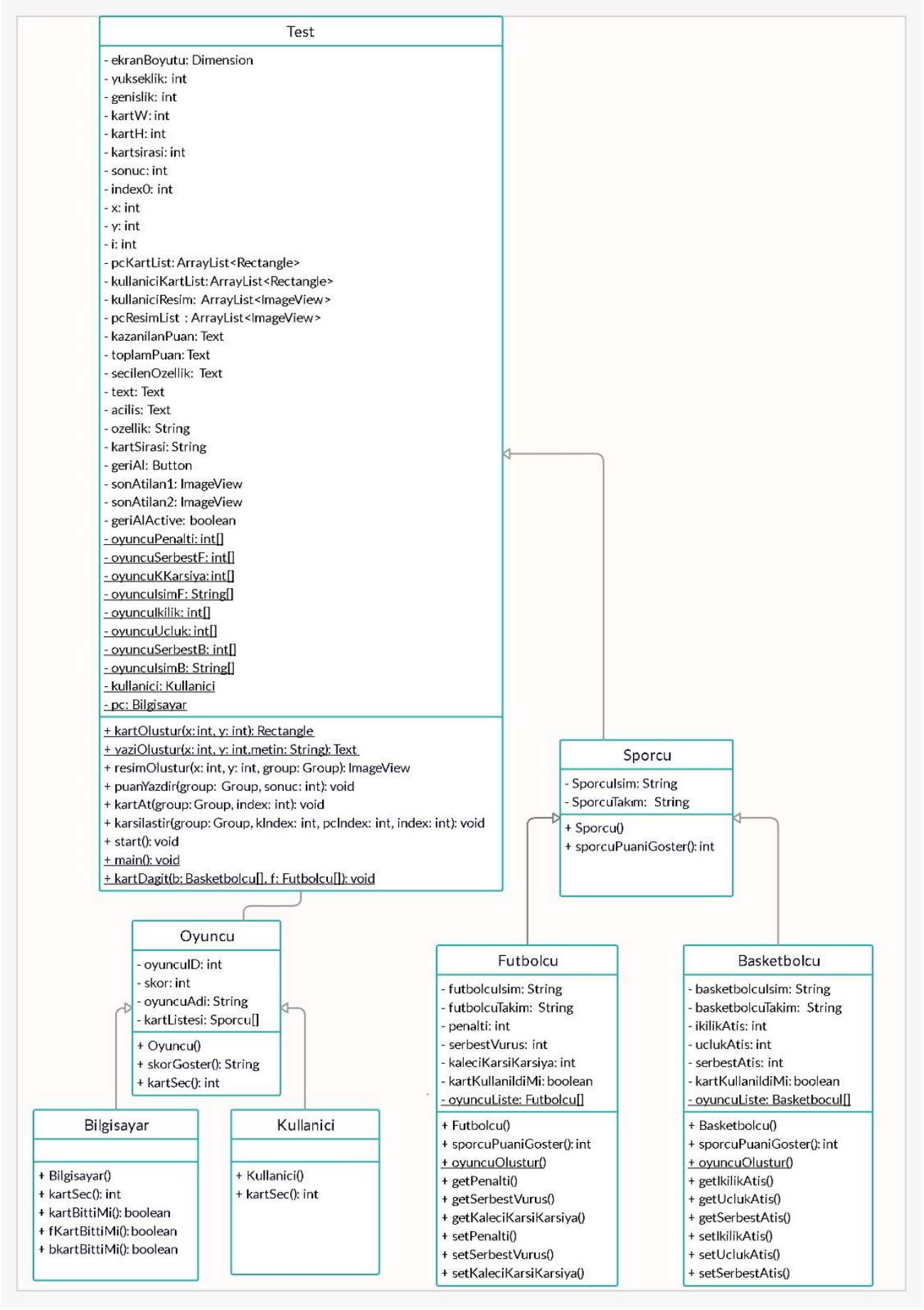
4.4. Programın Derlenmesi

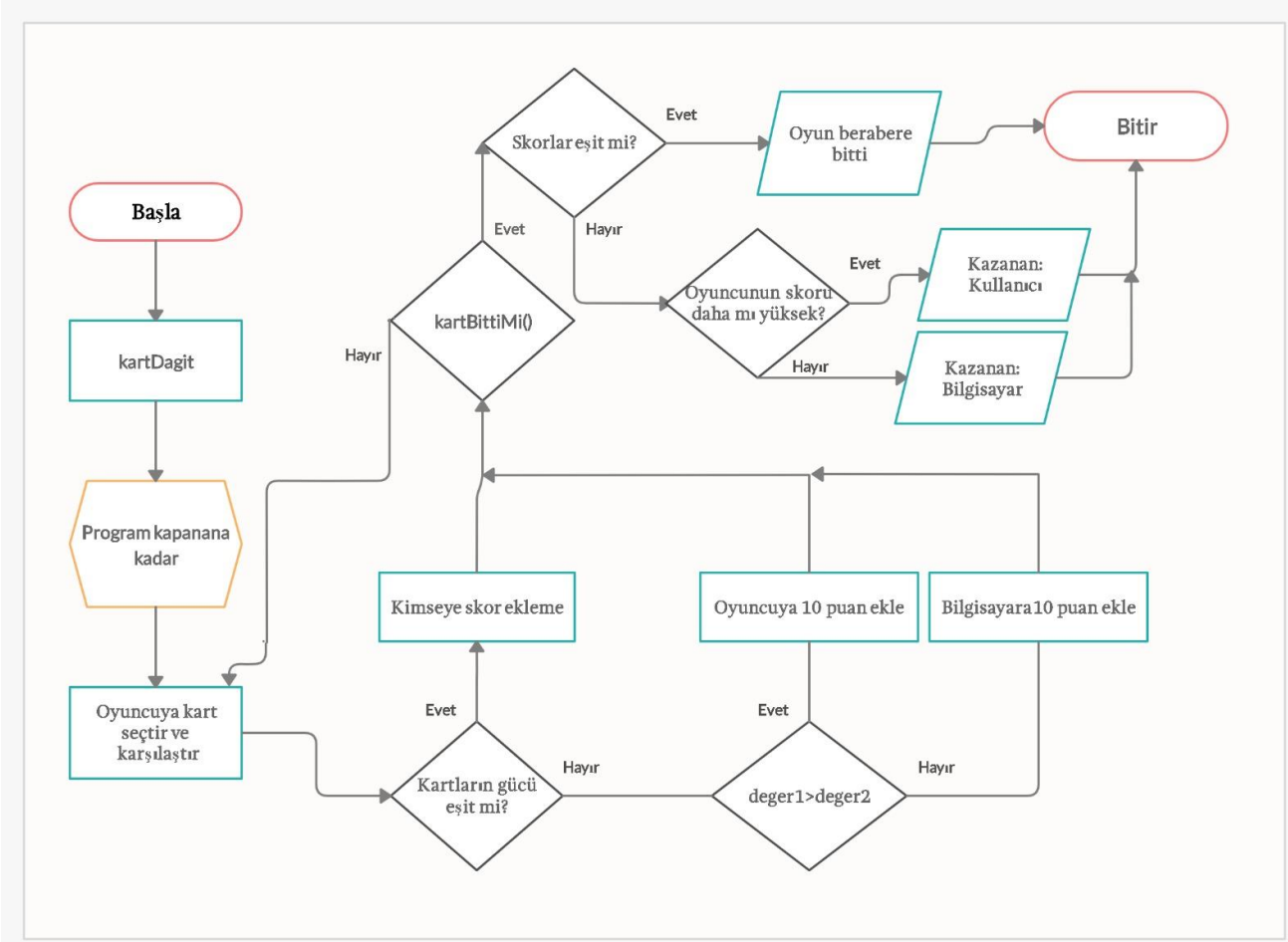
Programın kaynak kodu toplam 7 sınıf ve 18 resim dosyasından oluşmaktadır. Bu projeyi *Intellij Idea*, *Netbeans* veya farklı bir Java IDE'si kullanarak derleyebilirsiniz. Derlerken dikkat edilmesi gereken mesele ise kullandığınız IDE'ye *JavaFX* kütüphanesinin kurulmuş olması gereklidir.

Kaynakça

1. JavaFX kütüphanesi kullanım rehberi, https://www.javatpoint.com/javafx-tutorial

2. Çeşitli sorunları çözmek için, https://stackoverflow.com/


Test
- ekranBoyutu: Dimension
- yukseklik: int
- genislik: int
- kartW: int
- kartH: int
- kartsirasi: int
- sonuc: int
- index0: int
- x: int
- y: int
- i: int
- pcKartList: ArrayList<Rectangle>
- kullaniciKartList: ArrayList<Rectangle>
- kullaniciResim: ArrayList<ImageView>
- pcResimList : ArrayList<ImageView>
- kazanilanPuan: Text
- toplamPuan: Text
- secilenOzellik: Text
- text: Text
- acilis: Text
- ozellik: String
- kartSirasi: String
- geriAl: Button
- sonAtilan1: ImageView
- sonAtilan2: ImageView
- geriAlActive: boolean
- oyuncuPenalti: int[]
- oyuncuSerbestF: int[]
- oyuncuKKarsiya: int[]
- oyuncuIsimF: String[]
- oyuncuIkilik: int[]
- oyuncuUcluk: int[]
- oyuncuSerbestB: int[]
- oyuncuIsimB: String[]
- kullanici: Kullanici
- pc: Bilgisayar
+ kartOlustur(x: int, y: int): Rectangle
+ yaziOlustur(x: int, y: int,metin: String): Text
+ resimOlustur(x: int, y: int, group: Group): ImageView
+ puanYazdir(group: Group, sonuc: int): void
+ kartAt(group: Group, index: int): void
+ karsilastir(group: Group, kIndex: int, pcIndex: int, index: int): void
+ start(): void
+ main(): void
+ kartDagit(b: Basketbolcu[], f: Futbolcu[]): void
Sporcu
- SporcuIsim: String
- SporcuTakım: String
+ Sporcu()
+ sporcuPuaniGoster(): int
Oyuncu
- oyuncuID: int
- skor: int
- oyuncuAdi: String
- kartListesi: Sporcu[]
+ Oyuncu()
+ skorGoster(): String
+ kartSec(): int
Futbolcu
- futbolcuIsim: String
- futbolcuTakim: String
- penalti: int
- serbestVurus: int
- kaleciKarsiKarsiya: int
- kartKullanildiMi: boolean
- oyuncuListe: Futbolcu[]
+ Futbolcu()
+ sporcuPuaniGoster(): int
+ oyuncuOlustur()
+ getPenalti()
+ getSerbestVurus()
+ getKaleciKarsiKarsiya()
+ setPenalti()
+ setSerbestVurus()
+ setKaleciKarsiKarsiya()
Basketbolcu
- basketbolcuIsim: String
- basketbolcuTakim: String
- ikilikAtis: int
- uclukAtis: int
- serbestAtis: int
- kartKullanildiMi: boolean
- oyuncuListe: Basketbocul[]
+ Basketbolcu()
+ sporcuPuaniGoster(): int
+ oyuncuOlustur()
+ getIkilikAtis()
+ getUclukAtis()
+ getSerbestAtis()
+ setIkilikAtis()
+ setUclukAtis()
+ setSerbestAtis()
Bilgisayar
+ Bilgisayar()
+ kartSec(): int
+ kartBittiMi(): boolean
+ fKartBittiMi(): boolean
+ bkartBittiMi(): boolean
Kullanici
+ Kullanici()
+ kartSec(): int


Başla
kartDagit
Program kapanana kadar
Oyuncuya kart seçtir ve karşılaştır
Kartların gücü eşit mi?
Evet
Hayır
deger1>deger2
Evet
Hayır
Kimseye skor ekleme
Oyuncuya 10 puan ekle
Bilgisayara10 puan ekle
kartBittiMi()
Evet
Hayır
Skorlar eşit mi?
Evet
Oyun berabere bitti
Hayır
Oyuncunun skoru daha mı yüksek?
Evet
Kazanan: Kullanıcı
Hayır
Kazanan: Bilgisayar
Bitir