PROGRAMLAMA LABORATUVARI 1

2. PROJE

Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü
Programlama Laboratuvarı 1-2.Proje

Hakan AKGÜN 180202103
Gökalp GÖZÜBOL 180202084

1.Giriş

Bu doküman Programlama Laboratuvarı 1 dersi 2. Projesi için çözümü açıklamaya yönelik oluşturulmuştur. Dökümanda 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 projeyi hazırlarken kullanılan kaynaklar bulunmaktadır

2.Temel Bilgiler

Program Eclipse IDE üzerinde JAVA program dili ile yazılmıştır.

Görsel olarak oynanan bir oyun tasarlanmamıştır. Konsol uygulaması olarak oluşturulmuştur.

3.Proje Tanımı

Bir oyuncunun diğer bir oyuncuyla rekabet edebileceği taş kağıt makas oyunu tasarlanmıştır. Oyun kullanıcı-bilgisayar, bilgisayar-bilgisayar şeklinde oynanabilecektir.

3.1.Proje İsterler

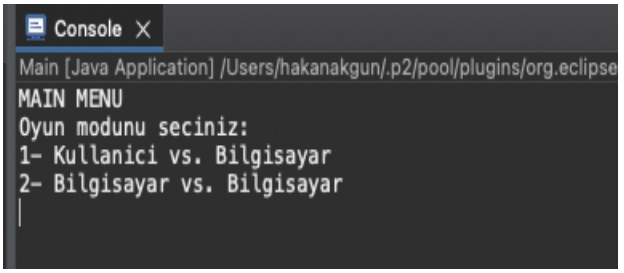
• Sınıf tanımlamaları, değişken tanımlamaları, override edilmesi gereken yöntemler, get,set metotları gibi kurallar olacaktır.

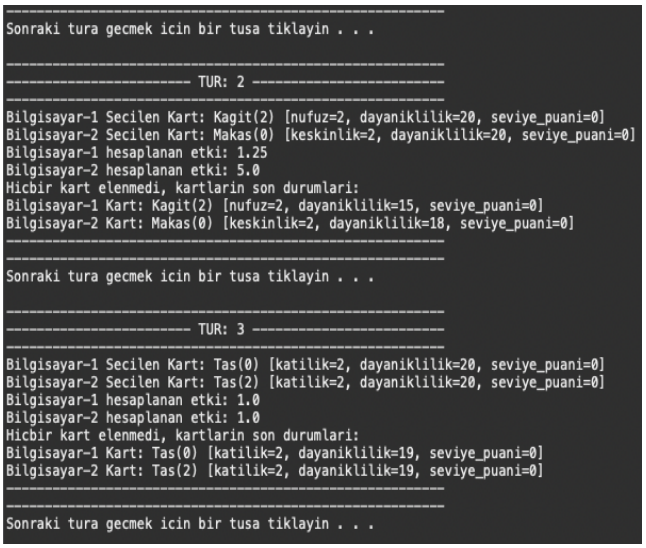
• Oyun dinamik bir şekilde ilerlemelidir.

• Dayanıklılık ve seviye puanı hesaplamaları, oyuncuların her adımdaki etki puanları, seviye atlama aşaması, nesnelerin eksildiği adımlar ve diğer bütün adımlar anlaşılır bir şekilde gösterilmelidir.

• Kullanıcı-bilgisayar oyununda kullanıcı nesne seçimini oyunun başında kendisi yapacak, bilgisayar ise rastgele bir şekilde nesneleri seçecektir.

Arayüzler hakkında birkaç görsel;





•Encapsulation: Sınıflardaki degerler dışarıdan doğrudan erişilmesini engellemek amaciyla protected yapilmistir, erisim icin getter ve setterler kullanilmak zorundadir.
•Inheritance: UML diyagramından da gorulecegi uzere toplam 8 adet inheritance özelliği kullanılmıştır.

•Polymorphism: Overload'a ihtiyac duymadigi icin kullanılmamıştır ama inheritance kisimlarinda override edilen metotlar bulunmaktadir. Örneğin Tas sinifindaki etkiHesapla() fonksiyonu AgirTas sinifinda override edilmistir.

•Abstraction: Nesne sinifinda kullanilmistir, böylelikle doğrudan Nesne sınıfından nesne üretilmesi engellenmistir. Ek olarak Nesne sinifindaki etkiHesapla() ve durumGuncelle() fonksiyonlari abstract yapılarak alt sınıflarda override edilmesi zorunlu kılınmıştır.

3.2 Yapılan Örnek Fonksiyonlar ve Bazı Source Code Files

Main.java : Uygulama kodlarının başlatılıldığı classtır. Run metodu buradadır..

Oyuncu.java: Bilgisayar ve kullanıcı olmak üzere oyunu oynayan iki oyuncu oluşturulmuştur. Bu iki oyuncunun farklı ve aynı özellikleri olacak şekilde implement edilmiştir. Aynı özelliklerini temsil etmek icin Oyuncu temel sınıfı oluşturulmuştur.

Bu sınıfta bulunan özellikler ve fonksiyonlar:

Bu sınıf abstract class yapısıyla tanımlandı.
oyuncuID, oyuncuAdi ve skor özellikleri eklendi.
Yapıcı(constuctor)metotları ve Parametreler oyuncuID, oyuncuAdi ve skor eklendi.
nesneListesi özelliği ile oyuncuların elinde bulunan nesneler listede tutuldu.
SkorGoster fonksiyonu ile oyuncuların skorları gösterildi.
nesneSec fonksiyonu yazılmalı fakat bu sınıf bilgisayar ve kullanıcı için farklı durumlarda çalışacağı unutulmamalıdır. üzerindeki sonraki dosyaya geçiyoruz. Bu şekilde fonksiyon tamamlanmış oluyor.

EtkiHesaplama(): Nesnelerin rakip nesneye karşı atak etkisini hesaplamak için oluşturulan bir metottur.

Örnek Fonksiyon

```
52      public void durumGuncelle(double etki) {
53          dayaniklilik -= etki;
54      }
```

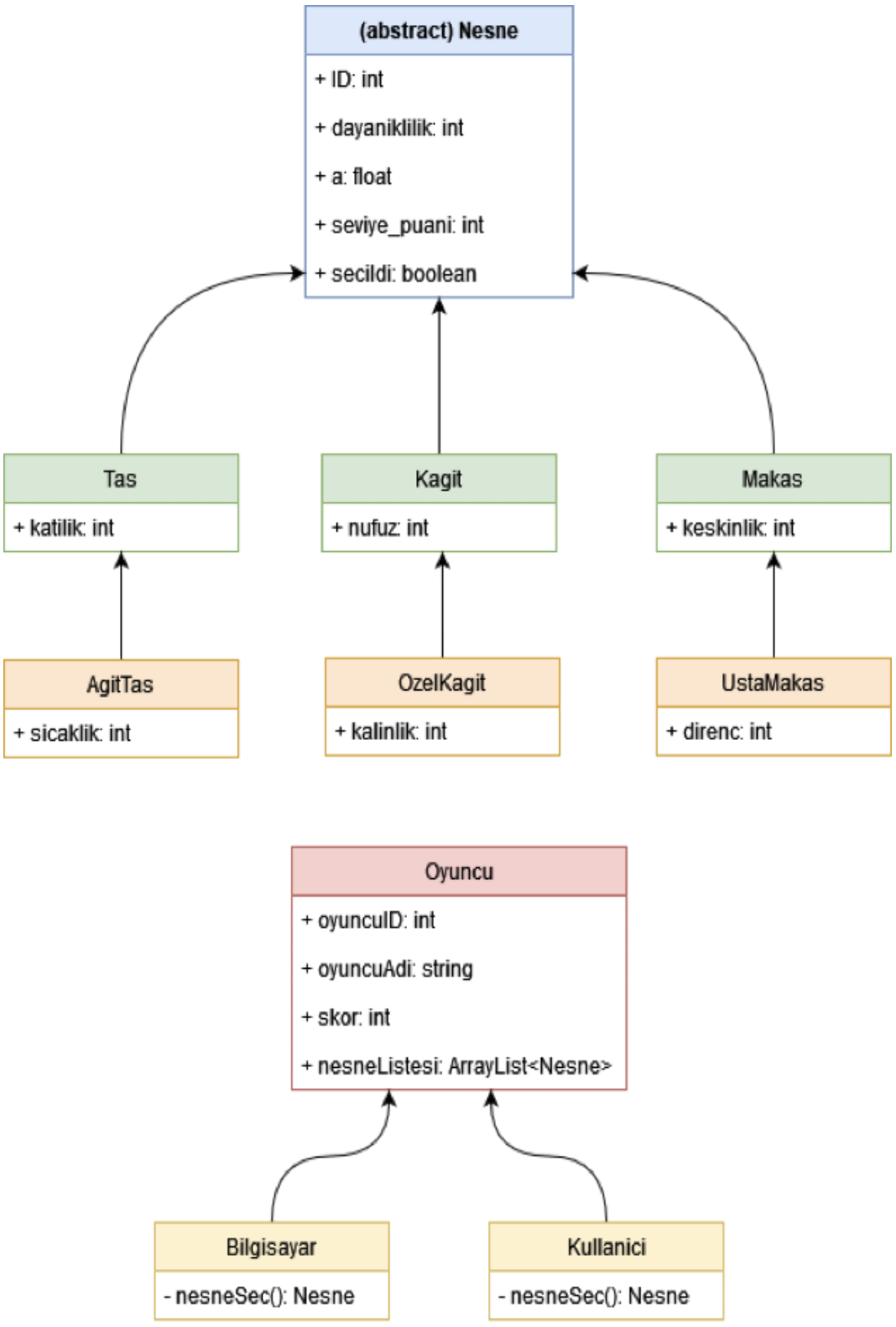
```
public double etkiHesapla(Nesne nesne) {
    // etki hesaplamasi
    if (nesne instanceof OzelKagit) // eger nesne OzelKagit ise
    {
        OzelKagit ozelNesne = (OzelKagit) nesne;
        return (keskinlik) / (ozelNesne.a * ozelNesne.getNufuz() * ozelNesne.getKalinlik());
    }
    if (nesne instanceof Kagit)
    {
        Kagit ozelNesne = (Kagit) nesne;
        return (keskinlik) / (ozelNesne.a * ozelNesne.getNufuz());
    }
    if (nesne instanceof AgirTas) // eger nesne AgirTas ise
    {
        AgirTas ozelNesne = (AgirTas) nesne;
        return (keskinlik) / ((1-ozelNesne.a) * ozelNesne.getKatilik() * ozelNesne.getSicaklik());
    }
    if (nesne instanceof Tas)
    {
        Tas ozelNesne = (Tas) nesne;
        return (keskinlik) / ((1-ozelNesne.a) * ozelNesne.getKatilik());
    }
```

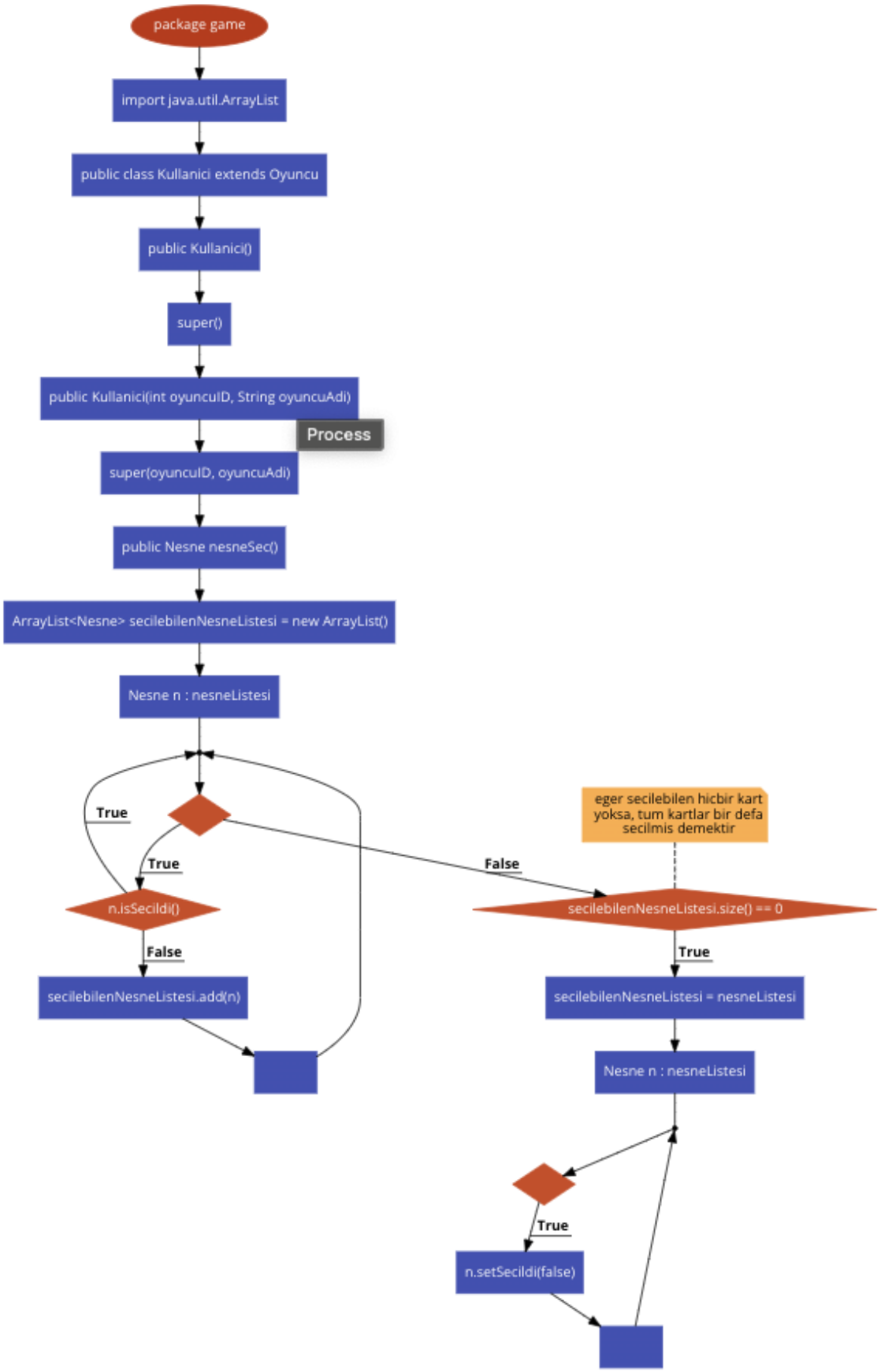
4. Kısaca Program Çalışma Yöntemi

Bir oyun nesnesi olusturulur ve Run methodu cagirlir. Run methodu icerisinde oncelikle oyun modu secilir, sonrasinda secilen oyun moduna gore(switch) oyuncular olusturulur. Daha sonra surekli while dongusu kullanilarak ekrana veriler yazdirilir ve kullanicidan veri girisi istenir(kullanici oyuna dahilse). Sonrasinda secilen kartlara gore etki degerleri hesaplanir ve kartlarin dayanikliklari bu hesaba gore dusurulur. Daha sonra kartlarin dayanikliklari kontrol edilere elenme durumu var ise kart oyuncudan silinir ve karsi oyuncunun kartina seviye puani eklenir. Eger bu durum sonrasinda kartin seviye puani yeterli ise kart kullanicidan silinir ve bir ust siniftan yeni bir nesne olusturularak kullaniciya eklenir. Ayni durumlar diger kart icin de yapilir. Bu islemlerden sonra iki oyuncunun da kartlari kontrol edilir, kart sayisi 0 olan herhangi bir oyuncu var ise ve ya hamle sayisini belli bir degere gelmis ise oyun sonlanir ve while dongusunden **break** kullanilarak cikilir. Sonrasinda while

dongusunun altinda, iki oyuncunun da kartlari iterate edilerek kartlarin dayaniklilik degerleri kullanicinin skotuna eklenir. Olusan skorlar karsilastirilir ve ekrana yazilir.

5. Akış Şeması




package game
import java.util.ArrayList
public class Kullanici extends Oyuncu
public Kullanici()
super()
public Kullanici(int oyuncuID, String oyuncuAdi)
Process
super(oyuncuID, oyuncuAdi)
public Nesne nesneSec()
ArrayList<Nesne> secilebilenNesneListesi = new ArrayList()
Nesne n : nesneListesi
True
True
False
n.isSecildi()
False
secilebilenNesneListesi.add(n)
eger secilebilen hicbir kart yoksa, tum kartlar bir defa secilmis demektir
secilebilenNesneListesi.size() == 0
True
secilebilenNesneListesi = nesneListesi
Nesne n : nesneListesi
True
n.setSecildi(false)