# SOY AĞACI UYGULAMASI

1st Tuna GÜLCAN
*Bilgisayar Mühendisliği*
*ProLab-1 3. Proje*
İzmit,KOCAELİ
tuna.gulcan@outlook.com

1st Fahrettin DOĞAN
*Bilgisayar Mühendisliği*
*ProLab-1 3. Proje*
İzmit,KOCAELİ
fahrettinndogann7@gmail.com

## I. ÖZET

Bu projede nesneye yönelik programlama ile excel dosyalarını okumayı ve bu excel dosyalarının içeriğiyle soy ağacı ile ilgili istenen maddeleri yaptık. Ve swing kütüphanesini kullanarak bu soy ağacının görselleştirilmesini yaptık.

## II. GİRİŞ

Bu rapor soy ağacı uygulamasının özet, giriş, yöntem, deneysel sonuçlar, sonuçlar, sözde kodlar ve kaynakça bölümlerinden oluşmaktadır.

## III. YÖNTEM

main() fonksiyonu : Csv dosyalarını okuyoruz. Okuduğumuz dosyaları AgacYapisi sınıfındaki kisiler arraylistinde tutuyoruz. AgacYapisi sınıfındaki root'umuzu ilk kişiye atıyoruz. İlk olarak bu kişilerin eşlerini buluyoruz. İç içe iki for döngüsüyle kişileri tuttuğumuz arraylisti geziyoruz. Birinci kişi evli ise ikinci kişinin bu kişinin eşi olup olmadığını kontrol ediyoruz. Birinci kişinin id'si ikinci kişinin eş id'si ile aynı mı? diye kontrol ediyoruz. Eğer kişinin eşi excel dosyasında var ise kişi olarak atama yapılıyor. Kişinin eşi excel dosyasında yoksa yeni bir Kişi oluşturup ismini, soyismini, idsini, kadınsa kızlık soyismini atıyoruz. Bu işlemin ardından çocuk bulmayı yapıyoruz. İç içe iki for ile kişi arraylistimizi geziyoruz. İkinci kişi birinci kişinin çocuğu mu diye kontrol sağlıyoruz. Eğer çocuğuysa kişinin çocuk arraylistine atama yapıyoruz. Çocuğun anne ve babasını da atama yapıyoruz. Ardından kullanıcıdan gelen menüden bir seçim yapmasını istiyoruz. Seçimi aldıktan sonra switch-case yapısı başlıyor. Case 1 : Elimizde bir yigin adında bir Stack ve cocuguolmayanlar adında bir Arraylist bulunuyor. yigin'a root değerimizi ekliyoruz. Depth First Search Algoritması ile kişinin soyağacını geziyoruz. Bu kişilerin çocuk arraylistinin boş olup olmamasına göre çocuğunun olup olmadığını kontrol ediyoruz. Eğer çocuğu yoksa bu kişiyi cocuguolmayanlar arraylistine ekliyoruz. Kontrol işlemi bittikten sonra ocugu olmayanlar arraylistindeki kişilerin yaş sıralamasını yapıyoruz. Burada kullandığımız algoritma ise Selection Sort. Case 2 : Elimizde kuyruk adında bir Queue ve uveykardesler adında bir Arraylist bulunuyor. Case 3 : Kullanıcıdan bulmak istediğimiz kan grubunu alıyoruz. Kişiler arraylistimizde gezerek kan grubunu girilen kan grubu ile karşılaştırıyoruz. Eğer kan grupları aynı ise kişiyi kangrubuayni arraylistine ekliyoruz. Case 4 : Kullanıcıdan id alıyoruz. Alınan id'yi kişiler arraylistinde arıyoruz ve bulunan yerin indexini değişkende tutuyoruz. Eğer bulduğumuz index agaçtaki kişi sayımızdan fazlaysa yeni rootumuzu ikinci ailenin başlangıç kişisine veriyoruz. Böylelikle istediğimiz ailedeki aynı işi yapan kişileri bulabileceğiz. Bu işlemin ardından başlangıç kişisinin soyağacını geziyoruz. İki kuyruğumuz var. Birinci kuyruktaki kişiyi ikinci kuyruktaki kişi ile karşılaştırıyoruz. Eğer ki aynı mesleği yapıyorlar ise aynimeslek arraylistimize bu kişileri ekliyoruz. İd'lerini de bir Set'te tutuyoruz ki aynı kişiyi iki kere eklemesin. Herkesi gezdikten sonra aynı mesleği yapanları bulmuş oluyoruz. Case 5 : Elimizde kuyruk1 adında bir Queue , idkontrol2 adında bir Set ve aynisimler adında bir Arraylist bulunuyor. Kuyruğumuza root'umuzu ekliyoruz ve kişinin soyağacını gezmeye başlıyoruz. Kişiler arraylistinde aynı isme sahip kişiyi bulursa bu kişileri aynisimler arraylistine ekliyoruz. Aynı kişileri eklememek adına bu kişilerin idlerini de idkontrol2 arraylistine ekliyoruz. Gezme işlemi bittikten sonra ayni isme sahip kişiler bulunmuş oluyor. Case 6 : Kullanıcıdan yakınlığını bulmak istediği iki kişinin idsini alıyoruz. Alınan idleri kişi listesinde gezerek bulunduğu indexleri değişkenlere atıyoruz. Bu indexleri yakinlikbul() fonksiyonuna yolluyoruz. Ve gelen stringi ekrana bastırıyoruz. Case 7 : Kullanıcıdan alınan id ile kişinin soyağacı ekrana bastırılıyor. Case 8 : nesilbulma() fonksiyonuna root'umuzu, static nesilsayisi değişkenimizi ve nesilsayi arraylistini yolluyoruz. Fonksiyon çalıştıktan sonra nesilsayi arraylistindeki en büyük değer soyağacının derinliğini veriyor. Case 9 : Kullanıcıdan id alıyoruz. Aldığımız id bizim root'umuz oluyor. nesilbulma() fonksiyonuna root'umuzu, static nesilsayisi değişkenimizi ve nesilsayi arraylistini yolluyoruz. Fonksiyon çalıştıktan sonra nesilsayi arraylistindeki en büyük değer soyağacının derinliğini veriyor. Case 10 : Excel dosyasındaki soyağacı ekrana bastırılıyor.

nesilbulma() fonksiyonu : Parametre olarak Başlangıç kişisini gösteren bir root , integer bir nesilsayisi değişkeni ve bulduğu nesil sayılarını içinde tutacağı bir arraylist alıyor. Fonksiyonumuz recursive bir fonksiyon. Alınan kişinin çocuğu varsa nesilsayisi değişkeninin değerini 1 artırıyor. Kişinin çocuklar arraylistini gezip her çocuğu tekrardan nesilbulma() fonksiyonuna yolluyor. Böylelikle kişinin çocuğu olmayıncaya kadar devam ediyor. Nesil sayısı değeri arrayliste aktarılıyor. Böylelikle tüm derinlikler bulunmuş oluyor. Aralarındaki en

büyük değer ise bizim aradığımız derinlik oluyor. Agacksisayisi() fonksiyonu : Parametre olarak Başlangıç kişisini gösteren bir root ve integer bir kisisayisi değişkeni alıyor. Alınan root'tan itibaren kişinin soyağacı geziliyor ve her kişi için kisisayisi değişkeninin değeri 1 artırılıyor. Return olarak kisisayisi değişkenini döndürüyor. yasHesapla() fonksiyonu : Eğer kişilerin herhangi birinde doğum tarihi yoksa doğum tarihi ekleniyor. Doğum tarihi String bir ifade olduğu için bunu Date türüne casting yapıyoruz. Şimdiki zaman ile arasındaki farkı bulup kişinin yaşını bu değere eşitliyoruz.

## IV. DENEYSEL SONUÇLAR

```
List<KisiDugum> a1 = new CsvToBeanBuilder<KisiDugum>(
        new FileReader(fileName: "/Users/tunagulcan/Desktop//Sayfa1.csv"))
        .withType(type: KisiDugum.class).build().parse();
List<KisiDugum> a2 = new CsvToBeanBuilder<KisiDugum>(
        new FileReader(fileName: "/Users/tunagulcan/Desktop/Sayfa2.csv"))
        .withType(type: KisiDugum.class).build().parse();
List<KisiDugum> a3 = new CsvToBeanBuilder<KisiDugum>(
        new FileReader(fileName: "/Users/tunagulcan/Desktop/Sayfa3.csv"))
        .withType(type: KisiDugum.class).build().parse();
List<KisiDugum> a4 = new CsvToBeanBuilder<KisiDugum>(
        new FileReader(fileName: "/Users/tunagulcan/Desktop/Sayfa4.csv"))
        .withType(type: KisiDugum.class).build().parse();
for (KisiDugum a : a1)
    AgacYapisi.kisiler.add(a);
for (KisiDugum a : a2)
    AgacYapisi.kisiler.add(a);
for (KisiDugum a : a3)
    AgacYapisi.kisiler.add(a);
for (KisiDugum a : a4)
    AgacYapisi.kisiler.add(a);
```

Fig. 1. *

**Dosya Okuma**

Excel dosyalarının okunduğu ve kişilerin listeye eklendiği kısım.

```
case 1:
    Stack<KisiDugum> yigin = new Stack<KisiDugum>();
    List<KisiDugum> cocuguolmayanlar = new ArrayList<>();
    yigin.push(AgacYapisi.root);
    while (!yigin.isEmpty()) {
        KisiDugum islemdugum = yigin.pop();
        if (islemdugum.cocuklar.size() != 0) {
            yigin.addAll(islemdugum.cocuklar);
        } else {
            cocuguolmayanlar.add(islemdugum);
            islemdugum.yasHesapla();
        }
    }
    System.out.println(x: "\nÇocuğu olmayan kişiler\n");
    for (KisiDugum asa : cocuguolmayanlar) {
        System.out.println(asa.getIsim() + "==>" + asa.yas);
    }
    KisiDugum gecicii;
    int adim = 1;
    System.out.print(s: "\nBaslangic adimi=  ");
    for (KisiDugum k : cocuguolmayanlar) {
        System.out.print(k.yas + "\t");
    }
    for (int i = 0; i < cocuguolmayanlar.size(); i++) {
        System.out.print("\n" + (adim++) + ". adim=  ");
        int min = i;
        for (int j = i + 1; j < cocuguolmayanlar.size(); j++) {
            if (cocuguolmayanlar.get(j).yas < cocuguolmayanlar.get(min).yas) {
                min = j;
            }
        }
        gecicii = cocuguolmayanlar.get(i);
        cocuguolmayanlar.set(i, cocuguolmayanlar.get(min));
        cocuguolmayanlar.set(min, gecicii);
        for (KisiDugum aas : cocuguolmayanlar) {
            System.out.print(aas.yas + "\t");
        }
    }
    break;
```

Fig. 2. *

**İster 1**

Çocuğu olmayan düğümlerin listesinin yaş sıralamasına göre kaydedilmesi.

```
case 2:
    Queue<KisiDugum> kuyruk = new ArrayDeque<KisiDugum>();
    List<KisiDugum> uveykardesler = new ArrayList<>();
    Set<String> idkontroll = new HashSet<>();
    kuyruk.add(AgacYapisi.root);
    while (!kuyruk.isEmpty()) {
        KisiDugum kisi1 = kuyruk.remove();
        for (KisiDugum kisi2 : AgacYapisi.kisiler) {
            if (!kisi1.cocuklar.isEmpty() && !kisi2.cocuklar.isEmpty()) {
                if (kisi1.partner.getId().equals(kisi2.getId())) {
                    for (KisiDugum kiisi1 : kisi1.cocuklar) {
                        sayac = 1;
                        for (KisiDugum kiisi2 : kisi2.cocuklar) {
                            if (kiisi1.getId().equals(kiisi2)) {
                                sayac = 0;
                            }
                        }
                        if (sayac == 1) {
                    //      uveykardesler.addAll()
                        }
                    }
                }
            }
        }
        if (kisi1.cocuklar.size() != 0) {
            kuyruk.addAll(kisi1.cocuklar);
        }
    }
    System.out.println(x: "Uvey kardeşler");
    for (KisiDugum a : uveykardesler) {
        System.out.println(a.getIsim() + "\t" + a.getId());
    }
    break;
```

Fig. 3. *

**İster 2**

Üvey kardeşler bulunarak harf sıralamasına göre kaydedilmesi.

```
case 3:
    kangrubuayni.clear();
    System.out.print(s: "Lütfen aramak istediğiniz kan grubunun giriniz:");
    String kangrubu = scan.nextLine();
    Set<String> kontrol = new HashSet<>();
    for (KisiDugum kisi : AgacYapisi.kisiler) {
        if ((kangrubu.compareTo(kisi.getKanGrubu())) != 0) {
            continue;
        } else {
            if (!kontrol.contains(kisi.getId())) {
                kangrubuayni.add(kisi);
                kontrol.add(kisi.getId());
            }
        }
    }
    for (KisiDugum a : kangrubuayni) {
        System.out.println(a.getIsim());
    }
    break;
```

Fig. 4. *

**İster 3**

Kan grubu aynı olanların listesinin kaydedilmesi.

```
case 4:
    System.out.print(s: "Lutfen id giriniz:");
    String kisiidsi = scan.nextLine();
    KisiDugum yeniroot = AgacYapisi.root;
    int kisisayi = 0;
    int agakisisayi = 0;
    for (KisiDugum a : AgacYapisi.kisiler) {
        if (a.getId().equals(kisiidsi)) {
            yeniroot = a;
            break;
        }
        kisisayi++;
    }
    agakisisayi = Agackisisayisi(AgacYapisi.root, agakisisayi);
    if (kisisayi >= agakisisayi) {
        yeniroot = AgacYapisi.kisiler.get(agakisisayi);
    }
    Queue<KisiDugum> kuyruks = new ArrayDeque<>();
    Queue<KisiDugum> kuyruks2 = new ArrayDeque<>();
    Set<String> idds = new HashSet();
    List<KisiDugum> aynimeslek = new ArrayList<>();
    kuyruks.add(yeniroot);
    sayac = 0;
    while (!kuyruks.isEmpty()) {
        kuyruks2.add(yeniroot);
        KisiDugum a = kuyruks.remove();
        while (!kuyruks2.isEmpty()) {
            KisiDugum a22 = kuyruks2.remove();
            if (!idds.contains(a22.getId())) {
                if (a.getMeslek().equals(a22.getMeslek()) && a.getId() != a22.getId()
                        && a.getMeslek().length() != 0) {
                    if (sayac == 0) {
                        aynimeslek.add(a);
                        idds.add(a.getId());
                        sayac++;
                    }
                    aynimeslek.add(a22);
                    idds.add(a22.getId());
                }
            }
            if (!a22.cocuklar.isEmpty()) {
                kuyruks2.addAll(a22.cocuklar);
            }
        }
        if (!a.cocuklar.isEmpty()) {
            kuyruks.addAll(a.cocuklar);
        }
    }
    for (KisiDugum a : aynimeslek) {
        System.out.println(a.getIsim() + " " + a.getId() + "==>" + a.getMeslek());
    }
    break;
```

Fig. 5. *

**İster 4**

Soyunda aynı mesleği yapan çocuklar veya torunlar gösterilmesi.

```
case 5:
    Set<String> idkontrol2 = new HashSet<>();
    Queue<KisiDugum> kuyruk1 = new ArrayDeque<KisiDugum>();
    List<KisiDugum> aynisimler = new ArrayList<>();
    kuyruk1.add(AgacYapisi.root);
    int sayac = 0;
    while (!kuyruk1.isEmpty()) {
        KisiDugum a = kuyruk1.remove();
        for (KisiDugum a22 : AgacYapisi.kisiler) {
            if (a.getIsim().equals(a22.getIsim())) {
                if (a.getId() != a22.getId()) {
                    if (sayac++ == 0) {
                        idkontrol2.add(a.getId());
                        aynisimler.add(a);
                    }
                    idkontrol2.add(a22.getId());
                    aynisimler.add(a22);
                }
            }
        }
    }
    for (KisiDugum a : aynisimler) {
        a.yasHesapla();
        System.out.println(a.getIsim() + "\t" + a.yas + "\t" + a.getId());
    }
    break;
```

Fig. 6. *

**İster 5**

Soy ağacında aynı isme sahip kişilerin ismi ve yaşlarının gösterilmesi.

```
case 6:
    System.out.print(s: "Lutfen birinci kisinin idsini giriniz.");
    String birindex = scan.nextLine();
    System.out.print(s: "Lutfen ikinci kisinin idsini giriniz:");
    String ikinciindex = scan.nextLine();
    int birindexx = 0, ikinciindexx = 0;
    for (KisiDugum k : AgacYapisi.kisiler) {
        if (k.getId().equals(birindex)) {
            birindexx = AgacYapisi.kisiler.indexOf(k);
            break;
        }
    }
    for (KisiDugum k : AgacYapisi.kisiler) {
        if (k.getId().equals(ikinciindex)) {
            ikinciindexx = AgacYapisi.kisiler.indexOf(k);
            break;
        }
    }
    KisiDugum birincikisi = AgacYapisi.kisiler.get(birindexx);
    KisiDugum ikincikisi = AgacYapisi.kisiler.get(ikinciindexx);
    System.out.println(birincikisi.getIsim());
    System.out.println(ikincikisi.getIsim());
    System.out.println(yakinlikbul(birincikisi, ikincikisi));
    break;
```

Fig. 7. *

**İster 6**

Kullanıcıdan alınacak 2 tane isim girdisinden sonra büyük olan kişinin küçük olan kişiye yakınlığının gösterilmesi.

```
case 7:
    String id;
    System.out.print(s: "Soy ağacını görmek istediğiniz kişinin ID'sini giriniz: ");
    id = scan.nextLine();
    for (KisiDugum i : AgacYapisi.kisiler) {
        if (i.getId().equals(id)) {
            soyAgaciSwing.ister(i);
            break;
        }
    }
    break;
```

Fig. 8. *

**İster 7**

Kullanıcıdan alınan kişi bilgisi ile o kişiye ait soy ağacının gösterilmesi.

```
case 8:
    KisiDugum kisi = AgacYapisi.root;
    int nesilsayisi = 1;
    List<Integer> nesilsayi = new ArrayList<>();
    nesilbulma(kisi, nesilsayisi, nesilsayi);
    int enbuyuk = 0;
    for (Integer s : nesilsayi) {
        if (s > enbuyuk) {
            enbuyuk = s;
        }
    }
    System.out.println(enbuyuk);
    break;
```

Fig. 9. *

**İster 8**

Soy ağacının kaç nesilden oluştuğunun bulunması.

```
case 9:
    System.out.print(s: "Lutfen kişi id giriniz:");
    String kisiid = scan.nextLine();
    KisiDugum kisi1 = AgacYapisi.root;
    int ids = 0;
    for (KisiDugum k : AgacYapisi.kisiler) {
        if (kisiid.equals(k.getId())) {
            ids = AgacYapisi.kisiler.indexOf(k);
            break;
        }
    }
    kisi1 = AgacYapisi.kisiler.get(ids);
    int nesilsayisi1 = 0;
    List<Integer> nesilsayi1 = new ArrayList<>();
    nesilbulma(kisi1, nesilsayisi1, nesilsayi1);
    int enbuyuk1 = 0;
    for (Integer s : nesilsayi1) {
        if (s > enbuyuk1) {
            enbuyuk1 = s;
        }
    }
    System.out.println(enbuyuk1);
    break;
```

Fig. 10. *
**İster 9**

Kullanıcıdan alınan isim girdisinden sonra o isimden sonra kaç nesil geldiğinin bulunması.

## V. SONUÇLAR

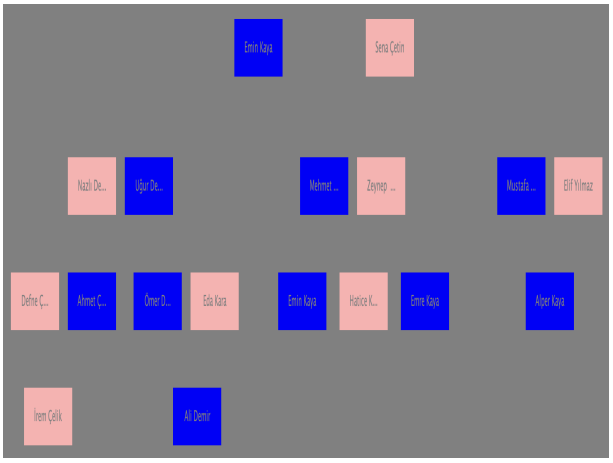


Fig. 11. *
**Soy Ağacı Ekran Çıktısı**

Bu proje nesneye yönelik programlama yapısını daha iyi kavramamıza yardımcı oldu. Neredeyse tüm özelliklerini bu projede kullandık ve öğrendik. Excel dosyalarını okumayı öğrendik bu proje sayesinde. Bu excel dosyalarında bulunan kişileri listelerin içerisine atayarak projenin isterlerini gerçekleştirdik. Ve soy ağacının görselleştirilmesini yaparken swing kütüphanesi ile nasıl görselleştirme yapılacağını ayrıntılarıyla öğrenmemize yardımcı oldu.

## REFERENCES


[1] https://stackoverflow.com
[2] https://www.javatpoint.com
[3] https://www.tutorialspoint.com/index.htm
[4] https://www.geeksforgeeks.org
[5] https://www.java-examples.com
[6] https://www.delftstack.com
[7] https://www.thejavaprogrammer.com


## VI. SÖZDE KOD

**1. Adım:** Program başlar ve csv dosyalarının içeriklerini okur.
**2. Adım:** Okunan dosya içerikleri bir listenin içerisine atılır.
**3. Adım:** Kullanıcıdan yapmak istediği işlemi seçmesi isteniyor.
**3.1. Adım:** Kullanıcı 1. isteri seçerse eğer program çocuğu olmayan kişilerin yaşlarını sıralanmış olarak göstermektedir.
**3.2. Adım:** Kullanıcı 2. isteri seçerse üvey kardeşleri alfabetik olarak sıralamaktadır.
**3.3. Adım:** Eğer 3. ister seçilirse kullanıcıdan bir kan grubu girmesi isteniyor. Ardından program o kan grubundaki kişileri gösteriyor.
**3.4. Adım:** Kullanıcı 4. isteri seçerse aynı mesleği yapan kişiler gösterilmektedir.
**3.5. Adım:** Kullanıcı 5. isteri seçerse soy ağacında aynı isme sahip kişilerin isimleri ve yaşları gösteriliyor.
**3.6. Adım:** 6. isterde kullanıcıdan iki adet kişi bilgisi alınıyor. Ardından program bu kişilerden büyük olanı gösteriyor. Ve büyük olan kişinin küçük olana göre yakınlık bilgisi de gösteriliyor.
**3.7. Adım:** Kullanıcı 7. isteri seçtikten sonra bir tane kişi bilgisi giriyor ve program bu kişiye ait soy ağacının görselleştirilmiş halini gösteriyor.
**3.8. Adım:** Kullanıcı 8. isteri seçerse program soy ağacının kaç nesilden oluştuğunu gösteriyor.
**3.9. Adım:** 9. ister seçildikten sonra kullanıcıdan kişi bilgisi isteniyor. Ve ardından bu kişiden sonra kaç nesil geldiği gösteriliyor.
**3.10. Adım:** Son ister olan 10. ister seçildiği zaman soy ağacının tamamının görselleştirilmesi yapılarak kullanıcıya gösteriliyor.
**3.11. Adım:** Kullanıcı eğer 0 seçimini yaparsa program tamamen sonlanıyor. Fakat kullanıcı 0 seçimini yapana kadar program devam etmektedir.