

2022 BAHAR DÖNEMİ

ALGORİTMA VE PROGRAMLAMA

DÖNEM PROJESİ

Konu	OOP(Nesne Yönelimli Programlama)
Teslim Eden Kişi	Muhammed Esad ÇİNGİLİ
Numarası	220541060
Ders Sorumlusu	Dr.Öğr.Üyesi Yaman AKBULUT
Lab Sorumlusu	Arş.Gör.Çağrı ŞAHİN

Nesne yönelimli programlama mantığını öğrenmek için Factory adı altında bir fabrika simülasyonu yapmaya çalıştım.Buradaki amacım bir fabrikada neler vardır? Bu yapıyı nasıl ifade edebilirim düşüncesiyle yola çıktım.Projeyi yaparken kullanıcı girişleri ve if else gibi karar yapılarından oluşarak bir menü tasarlamaya çalışmadım.Bunun yerine Nesne Yönelimli Programlama mantığını kavrayabilmek için Main class'ın içinde kendim kurucu çağrıları ve metotları kullandım.Şimdi nasıl modellediğime geçelim.

Person: Öncelikle neden person yani kişi sınıfı oluşturdum? Bir fabrika düşündüğüm zaman hiç şüphesiz insan faktörünün olmadığını düşünemedim ve böylelikle kalıtım hiyerarşisinin en üst tepesindeki sınıfımız oluşmuş oldu. 10 adet private özellik ,1 adet final özellik , default ve gelişmiş kurucu oluşturdum.Daha sonra setter getter ve toString metotlarını oluşturdum.Buradaki toString metodu Java'nın kendi kütüphanesindeki java.lang paketindeki Object sınıfından override edilmiştir. toString kullanımı genellikle information (bilgi) almak için kullanılır.

Employee: İşçi sınıfı person sınıfını kalıtım alıyor. Çünkü her işçi aslında bir kişidir.Bu sınıf default ve gelişmiş kurucu , 2 adet private özellik , setter getter metotları ve toString adında bilgi almak için string yazdıran bir metot içeriyor.

CalculateSalary: Soyut bir sınıf olarak maaş hesaplama sınıfını oluşturdum ve içerisinde salaryInfo adında bir metot içeriyor ve PayrollOffice sınıfında override edeceğim.

PayrollOffice: İşçilerin maaşlarının ödendiği , azaltılma ve artırma yapıldığı , ve maaş bilgisi yazdıran metotlar içeren sınıftır.

Decision: İçerisinde fire(kov) ve hire(işe al) metotları olan arayüz sınıfını oluşturdum.

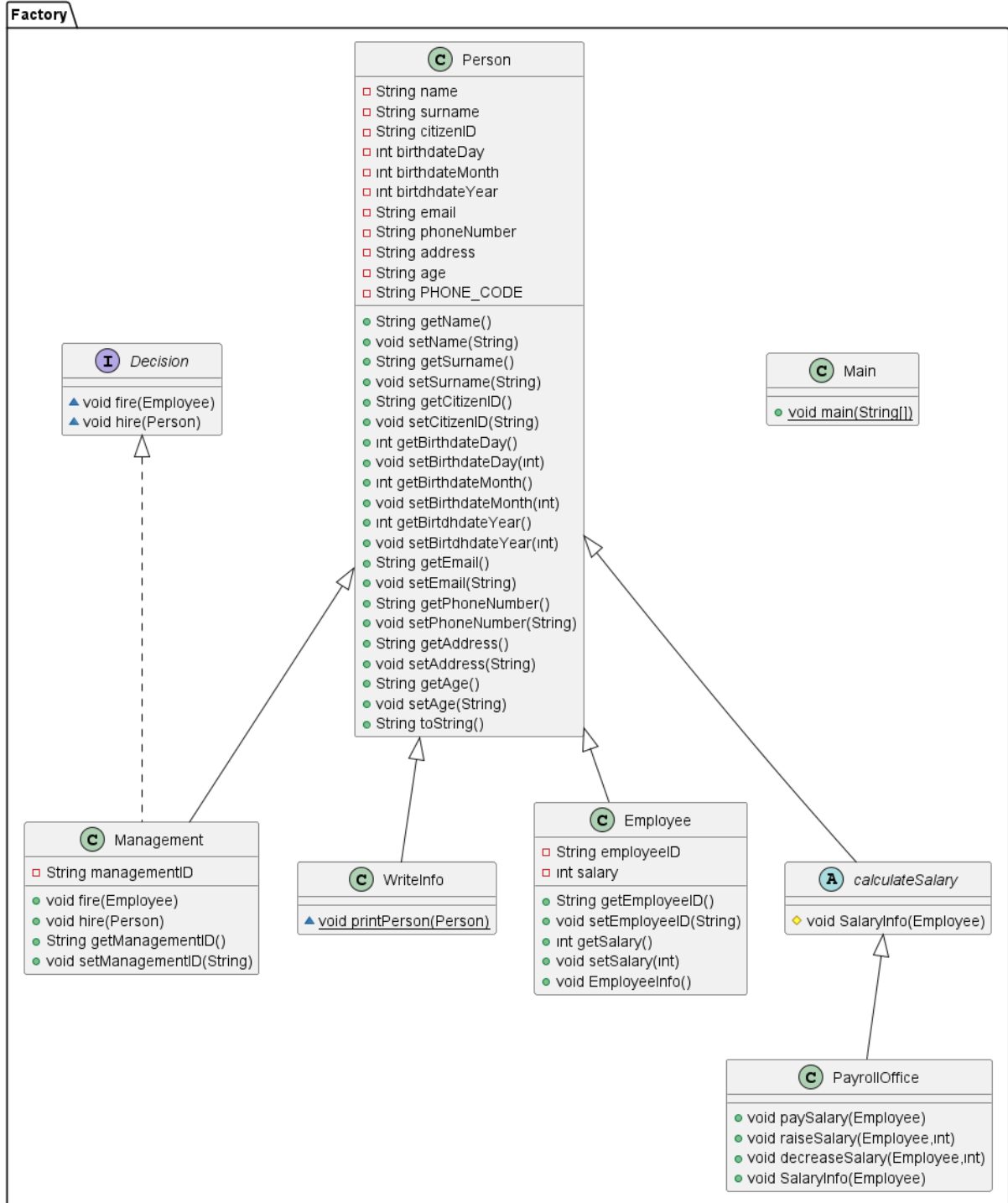
Managament sınıfında bunları implement edeceğiz.

Managament: 1 adet private özellik , 2 kurucusu , setter getter metotları bulunmaktadır.Aynı zamanda Decision sınıfını kendisine implement eder. Fire ve hire

metotlarının içini burada dolduruyoruz. Bu metotlar Person tipindeki referansları parametre olarak alıp farklı bir nesneyi göstermeyi yada nesne silmeyi hedefliyor.

WriteInfo: İçerisinde FileWriter , PrintWriter , File sınıflarından nesneler oluşturan printPerson adında statik bir metota sahiptir. Başka classlar üzerinden WriteInfo sınıfı üzerinden metot çağırısı yaparak kolaylıkla kullanabiliriz. Çünkü statik olarak belirledim. WriteInfo sınıfından nesne oluşturmadan metodu kullanabiliyorum.

UML diyagramı üzerinde modelleniş:



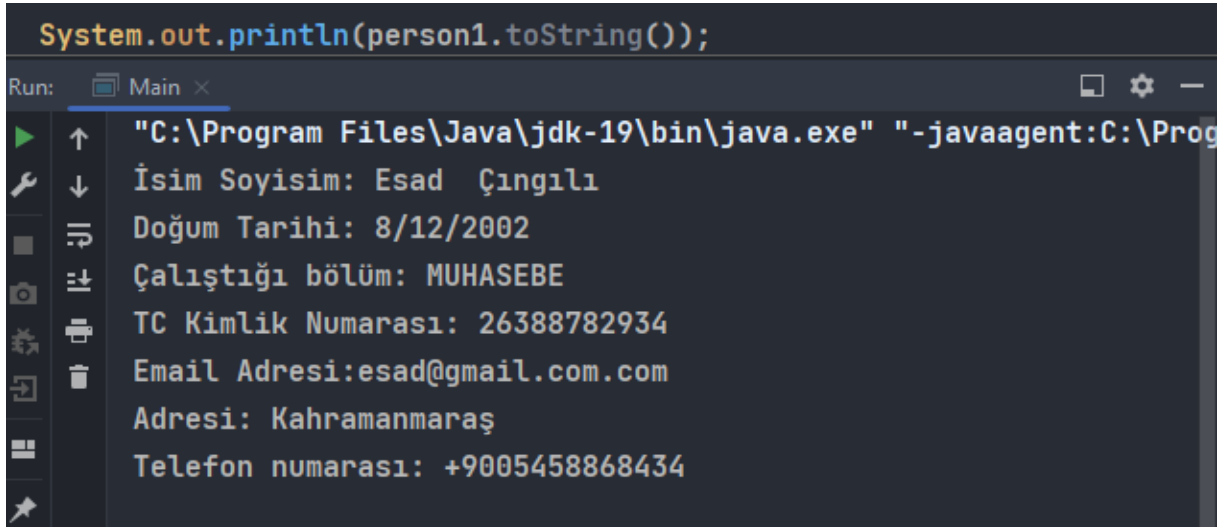
TEST SÜRECİ

Test sürecine **Person** sınıfından nesneler oluşturarak başladım.

```
Person person1 = new Person( name: "Esad ", surname: "Çingili", citizenID: "26388782934",  
    birthdateDay: 8, birthdateMonth: 12, birthdateYear: 2002,  
    email: "esad@gmail.com.com", phoneNumber: "05458868434", address: "Kahramanmaraş",  
    age: "MUHASEBE");
```

Daha sonra **Person** nesnesi üzerinden **person1** referansı ile metot çağrılarını yaptım.

```
System.out.println(person1.toString());
```



Daha sonra **Polimorfik nesneler** oluşturmaya çalıştım.

```
Person person2 = new Employee( name: "Şeyma", surname: "Dağdelen", citizenID: "220423903",  
    birthdateDay: 18, birthdateMonth: 4, birthdateYear: 1999, email: "seymawq@gmail.com",  
    , phoneNumber: "05054322132", address: "Ankara,Emek",  
    department: "İK", employeeID: "220432", salary: 15000);
```

Bu nesnelerin içerisine **polimorfik metotlar** yerleştirdim aşağıdaki gibi.

```
public String toString(){  
    return super.toString()+ "\nÇalışan ID: "+  
        getEmployeeID()+ "\nMaaşı: " + salary;  
}
```

Daha sonra yukarıdaki gibi **person2** referansı üzerinden **toString** metotunu çağırarak kontrollerimi yaptım. Bu noktada getter setter metotlarını kullanıp göstermeyeceğim.

Çünkü istediğimiz şekilde çalışabiliyor.Şimdi benim için **dönüm noktası** olan noktaya gelelim.

Managament sınıfının ne yaptığını yukarıda anlatmıştım.Burada yapmak istediğim şey girilen Person tipindeki referansı bir alt sınıfın nesnesini oluşturarak göstermeye çalışmaktı. Aynı zamanda alt sınıfın kendine has özellikleri olan employeeid ve maas özelliklerini kurucuda set ettim.(Buradaki id oluşturma semboliktir daha gelişmiş bir şekilde id oluşturulabilir.) Tüm bunları yaptıktan sonra toString metodu çağırarak Employee nesnesinin oluşup oluşmadığını kontrol etmek istedim.

```
@Override
public void hire(Person person) {
    System.out.println(person.getName() + " " + person.getSurname() + " adlı kişi işe alındı.");
    System.out.println("Employee id oluşturuluyor...");
    java.util.Random rastgele = new java.util.Random();
    java.util.Scanner girdi = new java.util.Scanner(System.in);
    int rastgeleid = rastgele.nextInt( origin: 10000, bound: 99999);
    String employeeid = String.valueOf(rastgeleid);
    System.out.println("İşçi maaş belirle: ");
    int maas = girdi.nextInt();
    person = new Employee(person.getName(), person.getSurname(),
        person.getCitizenID(), person.getBirthdateDay(),
        person.getBirthdateMonth(), person.getBirthdateYear(),
        person.getEmail(),person.getPhoneNumber(), person.getAddress(),
        person.getAge(),employeeid,maas);
}
```

Aşağıda görüldüğü gibi Çıktı ekranında employee bilgisi beklerken sadece person bilgilerini geldiğini farkettim.Çünkü Person tipindeki referans alt sınıfı olan Employee nesnesini gösterebiliyor ve oluşturabiliyordu.

```
management.hire(person1);
System.out.println(person1.toString());
```

Run: Main

Esad Çingılı adlı kişi işe alındı.
Employee id oluşturuluyor...
İşçi maaş belirle:
9500
İsim Soyisim: Esad Çingılı
Doğum Tarihi: 8/12/2002
Çalıştığı bölüm: MUHASEBE
TC Kimlik Numarası: 26388782934
Email Adresi:esad@gmail.com.com
Adresi: Kahramanmaraş
Telefon numarası: +905458868434

Ama referans üst sınıf olarak tanımlandığı için **alt sınıfın metotlarına ulaşamıyordu**.Amacım person tipindeki referansı alt tip olan employeee dönüştürmekti.Sayısız denemelerim sonucunda çok değişik sonuçlar aldım.Bunlardan biri **“class cast exception”** idi.Compile-time da herhangi bir sıkıntı yok iken Run-time da sıkıntı yaşıyordum.”**instance of** “ kullanarak bir metot oluşturmaya çalıştım ama başarısız oldum.Nesnenin referansını değiştirmeye

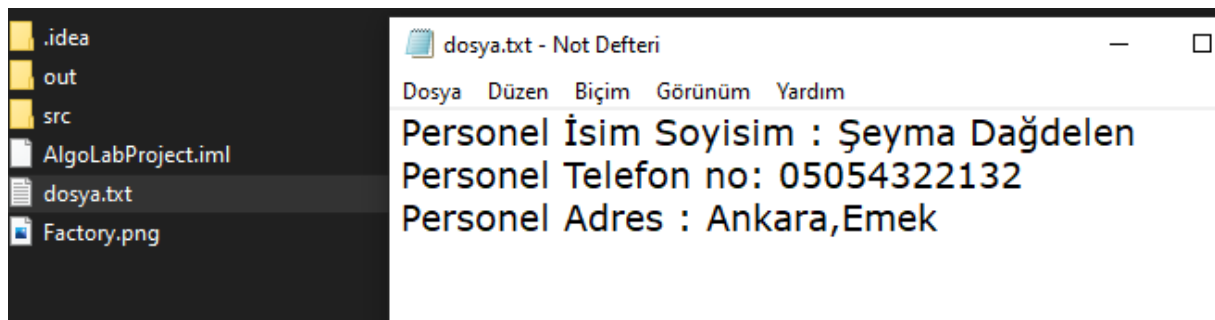
çalışmayı bıraktım ve araştırmalarımaya devam ediyorum. Onun yerine hire metodunun içine toString ekledim ve işçi bilgisini getirdi.

Sıra **PayrollOffice** sınıfını test etmeye geldi. Bu sınıfta maaş kontrolü sağlayan bir muhasebe departmanı gibi düşündüm. Employee üzerindeki maaş bilgileri üzerinde değişiklikler yaptım.

```
payrollOffice.SalaryInfo((Employee) person2);  
// Şimdi zam yapmaya çalışıyoruz.  
payrollOffice.raiseSalary((Employee) person2, addition: 5000);  
// Şimdi maaşı azaltıyoruz.  
payrollOffice.decreaseSalary((Employee) person2, decrease: 10000 );  
// Şimdi maaşı ödüyoruz.  
payrollOffice.paySalary((Employee) person2);
```

Şimdi işçi bilgisi yazdıran **WriteInfo** sınıfını deniyoruz.

```
// Personel bilgisini txt yazdırmak için kullanıyoruz  
WriteInfo.printPerson(person2);
```



Başarılı bir şekilde yazdı.

Evet genel anlamda OOP prensibini anlamaya ve geliştirmeye çalıştığım bir proje oldu.

Değerli dönüşlerinizi bekliyorum.

Mail: muhammedasadcingili@gmail.com