**FITNESS CENTER**

# Grup üyeleri ve GitHub proje linkleri

Gökhan Güneşoğlu 201817014 https://github.com/gokhanmuzaffer/Nesne-Yonelimli-Programlama-Proje-Odevi

Damla Nur Emlik 201816009 https://github.com/Damlanuremlikk/Fitness-Center

# ÖZET

Projemiz Fitness merkezindeki tüm çalışanların ve müşterilerin bilgilerini işleyerek ve Fitness merkezi içerisindeki bölüm, oda ve gerekli ekipmanların hepsini kayıt altına tutarak Fitness merkezimizin işleyişini düzene sokmayı amaçlar. Fitness merkezimizin içerisinde genel müdür, müdür, eğitmen, kayıtçı ve temizlikçi ve müşteriler(Gold,Ekonomik,Standart) bulunmaktadır. Her bir çalışan ve müşteri için alanına göre farklı id ler atanmakta ve farklı görevlere erişim sağlayabilmektedirler. Örneğin Genel müdürün şube silme yetkisi olması ve aynı zamanda müdürün yapabileceği tüm işlere yetkisi olması gibi veya genel müdürümüzün çalışan ekleme çıkarma yetkisi olduğu gibi vb. Programımızda giriş yaptığı kişiye göre yapabileceği işlemleri önüne getirebilmesi için giriş ekranı mevcuttur.

**Anahtar Sözcükler:** Fitness Merkezi, Loglama, Kayıt güncelleme, Kayıt ekleme, Kayıt çıkarma.

# GİRİŞ

# Projemizin amacı hedef kitlemiz olan tüm personel ve müşterilere kendilerine özgü işlemleri yapma imkanı sunmaktır.

# PROJE İÇERİĞİ

* 1. **PROJE TANITIMI**

Projemizin çözüm sunacağı problemler:

Müşteri bilgilerini kayıt etmede ,depolamada karşılaşılan sıkıntılar Sistemde yapılan değişiklerin kimler tarafından yapıldığının belirsizliği Genel müdür, müdür, müşteri ve diğer çalışanlara kimlik atama problemi Fitness merkezi içerisindeki demirbaş listesindeki karışıklıklar Kullanıcıların bilgilerini görememe ve güncelleyememe problemi   
Kötü niyetli çalışanların takip edilememesi

Çözüm önerileri tasarkarken sergilediğimiz yaklaşım:

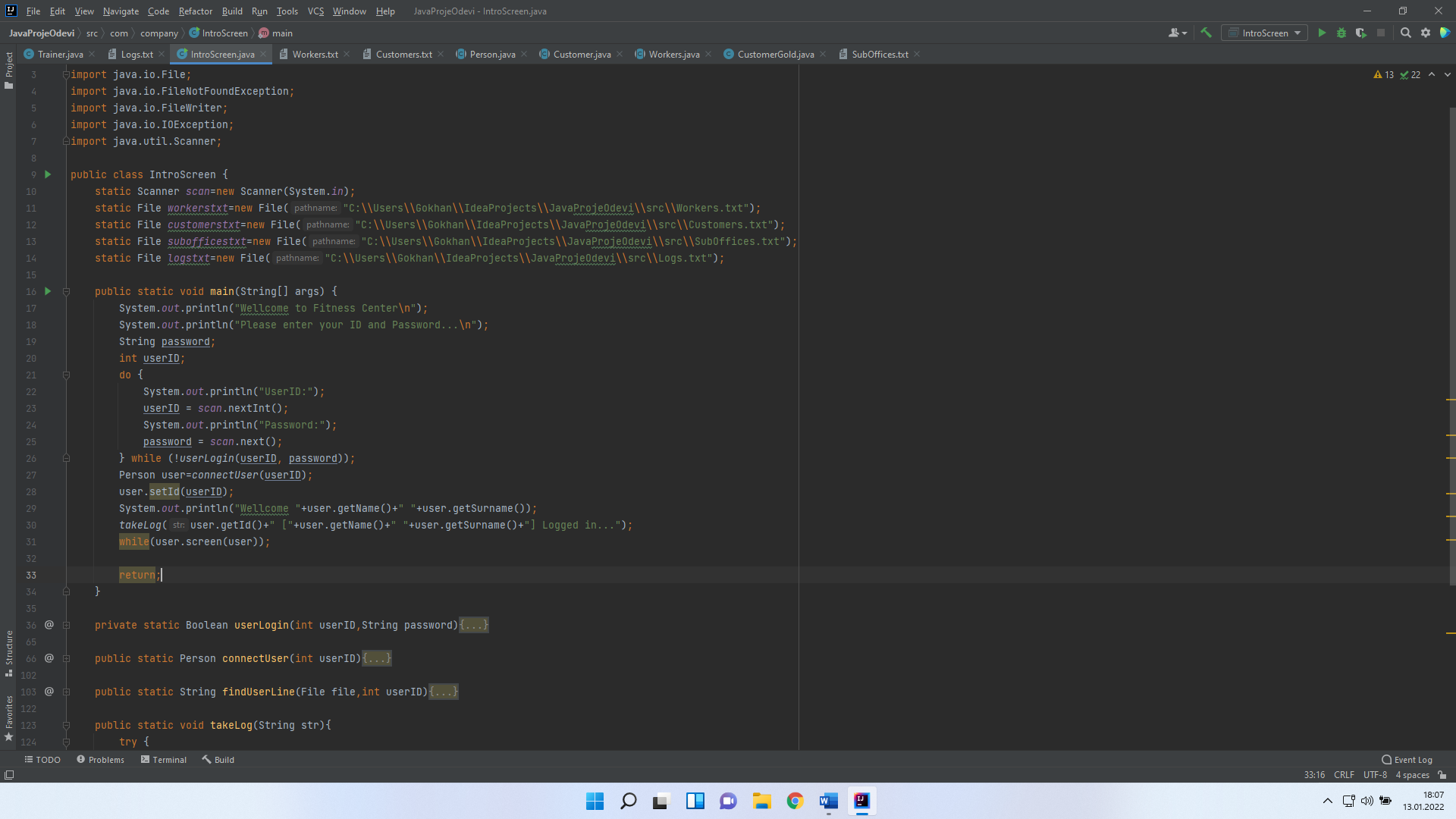
1) Hedef kitlemizi belirledik: Fitness merkezindeki tüm çalışanlar ve müşteriler. Kullanıcı olarak adlandırdık.  
Çalışanlar: Genel müdür, müdür, kayıtçı, eğitmen, temizlikçi.  
Müşteriler: Farklı tipte üyeliklere sahip müşteriler olabileceğini düşündük.

2)Farklı türdeki kullanıcıların ihtiyaclarını düşündük ve onların ihtiyaçlarına göre operasyonlar kurduk. Örneğin genel müdür için şube oluşturma operasyonu varken bu yetkiyi diğer hiçbir çalışana sunmadık. Ancak bilgilerini getirme ve genel bilgilerini düzenlene operasyonlarını her tip kullanıcıya açık olarak sunduk.

3)Programa eklenebilir ve sistemi güzelleştirebilir operasyonlar düşündük. Kayıtlarda değişiklik yapabilen operasyonlar kullanıldığında (bilgilerini güncelleme/yeni müşteri/çalışan ekleme/silme), yapılan değişikliği ve kim tarafından yapıldığını kayıt eden bir operasyon (Loglama) kurduk.

Genel olarak kod mimarisi:

Projemizde kullandığımız bazı teknikleri ve algoritmaları burada belirteceğiz.



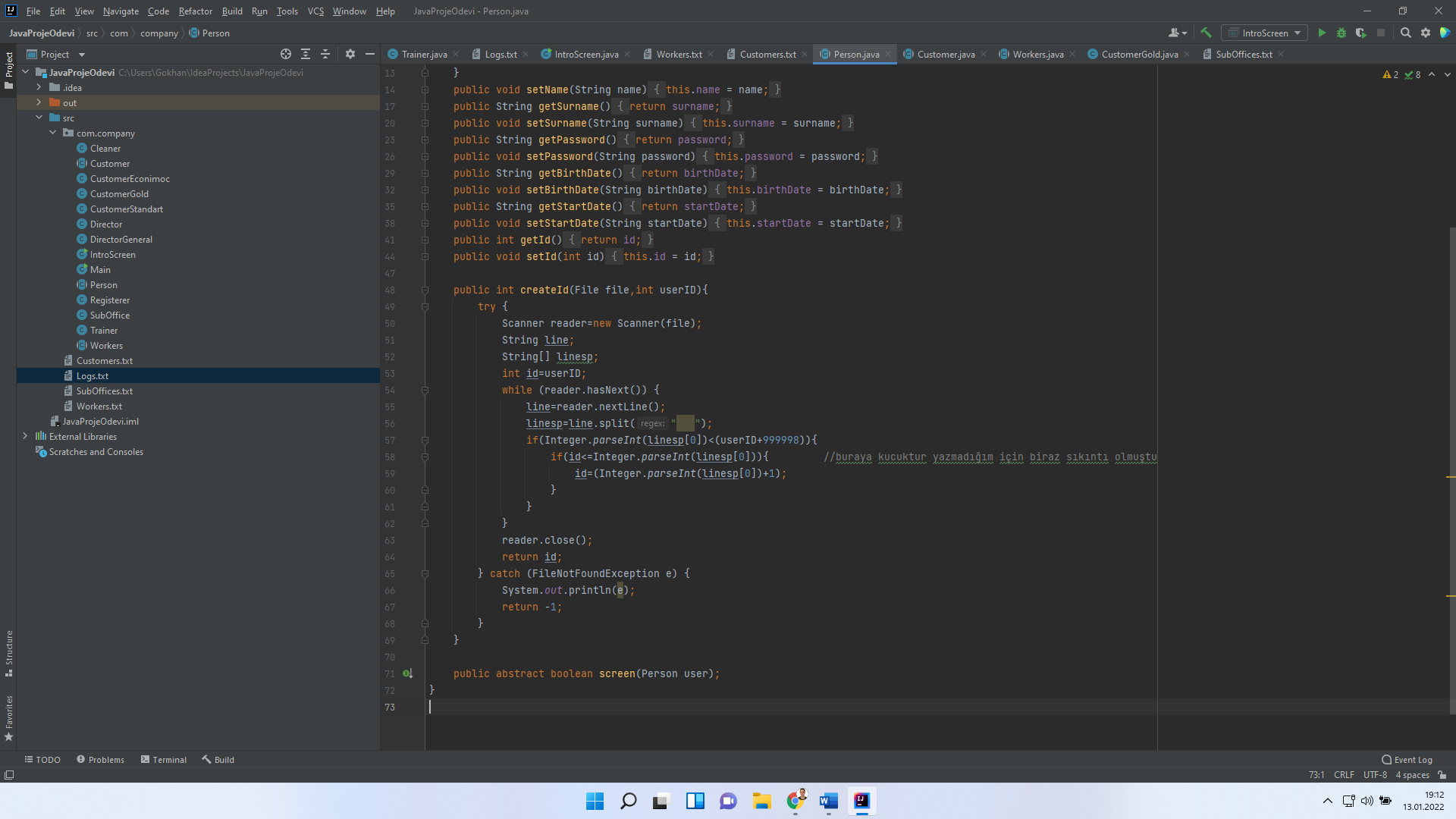
Programı IntroScreen classını çalıştırarak başlattığınızda konsola bir takım yazılar çıkar ve sonra kullanıcıdan ID ve şifresini isteyen bir do-while döngüsüne girer. While ın içine boolean bir değer döndüren userLogin metodu koyulmuştur. Bu metod kullanıcının girdiği ID ilk önce Workers.txt dosyasında, daha sonra Customers.txt dosyasında satır satır okuyarak arar. Ne zaman bir id eşleşmesi yakalarsa, aynı satırdaki password bilgisi ile kullanıcının girdiği password ü karşılaştırır. Eğer password:  
 doğru ise True döner döngüden çıkar.   
 yanlış ise False döner ekrana şifrenin yanlış olduğunu yazdırır ve döngü devam eder.  
Eğer hiç id bulamazsa False döner ve id nin bulanamadığını ekrana yazdırır.

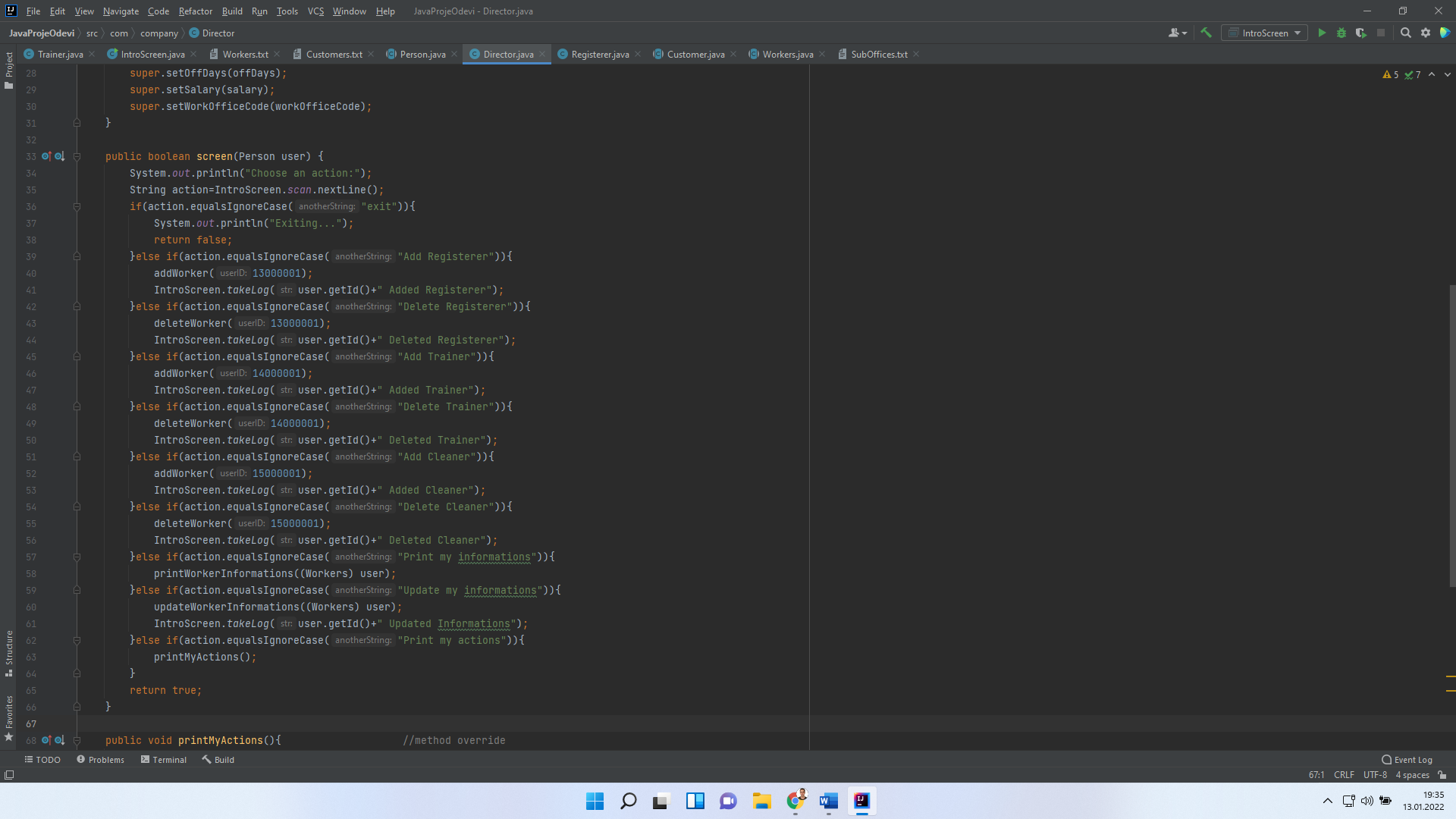
Başarılı bir giriş yapıldıktan sonra 27. Polimorfizm uygulanır. connectUser metodu bizden bir ID ister ve verilen ID bilgisine göre ne tür bir çalışan veya ne tür bir müşteri olduğunu seçer ve o tipte bir nesne oluşturur.

31. satırdaki screen(Person user) metodu ise kullanıcıyı kendi kullanacağı ekrana yönlendiren bir metoddur. Eğer bu metodu, her tipdeki kullanıcıların en üst class ı olan Person classında abtarct olarak tanımlamasaydık, “user.” dedikten sonra “screen” diyemezdik. Bu metod, kendi türüne göre kullanıcıya farklı işlemler/yekiler sunmak için, Person class ında abstact olarak tanımlanmış ve en alt classları(Claner, Trainer, Registerer, Director, DirectorGeneral, CustomerGold, CustomerStandart, CustomerEconomic) tarafından override edilmiştir.

Girilen ID kullanıcının nasıl bir ekran ile karşılaşacağını belirlediği için, bu ID lerin nasıl oluşturulduğu da önemli. Bunun için ID lerin bu programdaki anlamlarından bahsetmekte gerekir. ID leri 8 haneli yaptık ve son 2 hane, kullanıcıların türlerini belirliyor, ilk 6 hane ise kendi türündeki diğer kullanıcılarla karışmaması için.

11=>DirectorGeneral, 12=>Director, 13=>Registerer, 14=>Trainer, 15=>Cleaner.  
21=>CustomerGold, 22=>CustomerStandart, 23=>CustomerEconimic.



Person Class ındaki createId fonksiyonu bizden bir dosya ve bir tam sayi(userID) istemektedir. Bu argumanlar kullanıcının dolduracağı değerler değildir,  kullanıcının kullanabileceği metotlarda, doldurulmuş şekilde yazılıdır. add registerer dediği zaman Workers.txt dosyasındaki 13000000 ile 1400000 arasındaki en yüksek id yi bulur, bir fazlasını değer olarak döndürür. Kullanıcı add customer dediği zaman, (girilen customerType a göre, örneğin Gold) Customers.txt dosyasındaki 21000000 ile 22000000 arasındaki en yüksek id yi bulur ve bir fazlasını döndürür.   
Tüm kullanıcılar aynı dosyada yazılabilirdi veya tüm kullanıcıların hepsi ayrı dosyalarda saklanabilirdi, bu proje için boyle uygun düşündük. SubOffice eklerken de id ile ilgili durum benzerdir. Bunların dışında screen fonksiyonu boolen bir değer döndüren fonksiyondur. Kullanıcı exit yazana kadar çalışmaya devam eder çünkü (IntroScreen 31. satır da) main fonksiyonun içindeki kullanımı while döngüsüne koşul olarak yazılmıştır.

Miras durumları ve yetkinlikler hakkında: Örnek bir miras alma durumu:  
Bir genel müdür, kayıtçının yaptığı müşteri ekleme/çıkarma gibi işlerle uğraşıp uğraşmaması gerektiği konusunda kararsızlıklar yaşadığımız için, screen fonksiyonuna bu fonksiyonları yazmadık. Ancak miraslandırmayı:  
Person => Workers => Registerer => Director => DirectorGeneral   
şeklinde yaptık ki, istenildiği durumda kolaylıkla eklenip çıkarılabilir bu yetkiler. Ancak DirectorGeneral içerisinde tanımlanmış olan bir fonksiyonu (add subOffice vb.) diğer classlar kullanamaz. Kullanılabilir olması için kullanmak istenen classta veya daha üst classında (Workers ya da Person ) tanımlanması gerekir. Bu durum bizim istediğimiz bir durum olmadığı için miraslandırmayı bu şekilde yapmayı uygun gördük.

Bunların dışındaki tasarımlar UML diyagramından daha rahat anlaşılmaktadır.

* 1. **PROJE ÖN TASARIM VE GELİŞTİRME TASARIMI**

Projenin tasarım amaacından önce saha çalışması ile birlikte veri toplama işlemini gerçekleştirdik. Fitness merkezlerinde çalışan ve Fitness merkezine giden insanlardan bilgi toplayıp projemizin sınırlarını belirledik. Ardından “Bu bilgileri koda nasıl dökeriz?” i tartıştık bu süre zarfında genel hatlarıyla oluşturduğumuz diyagramımız bittikten sonra kod yazım aşamasına geçtik. Kod yazarken yaptığımız şema işlerimizi daha da hızlandırdı. Kod yazdıkça projemizdeki eksiklikleri fark ettik ve onlara çözümler getirdik

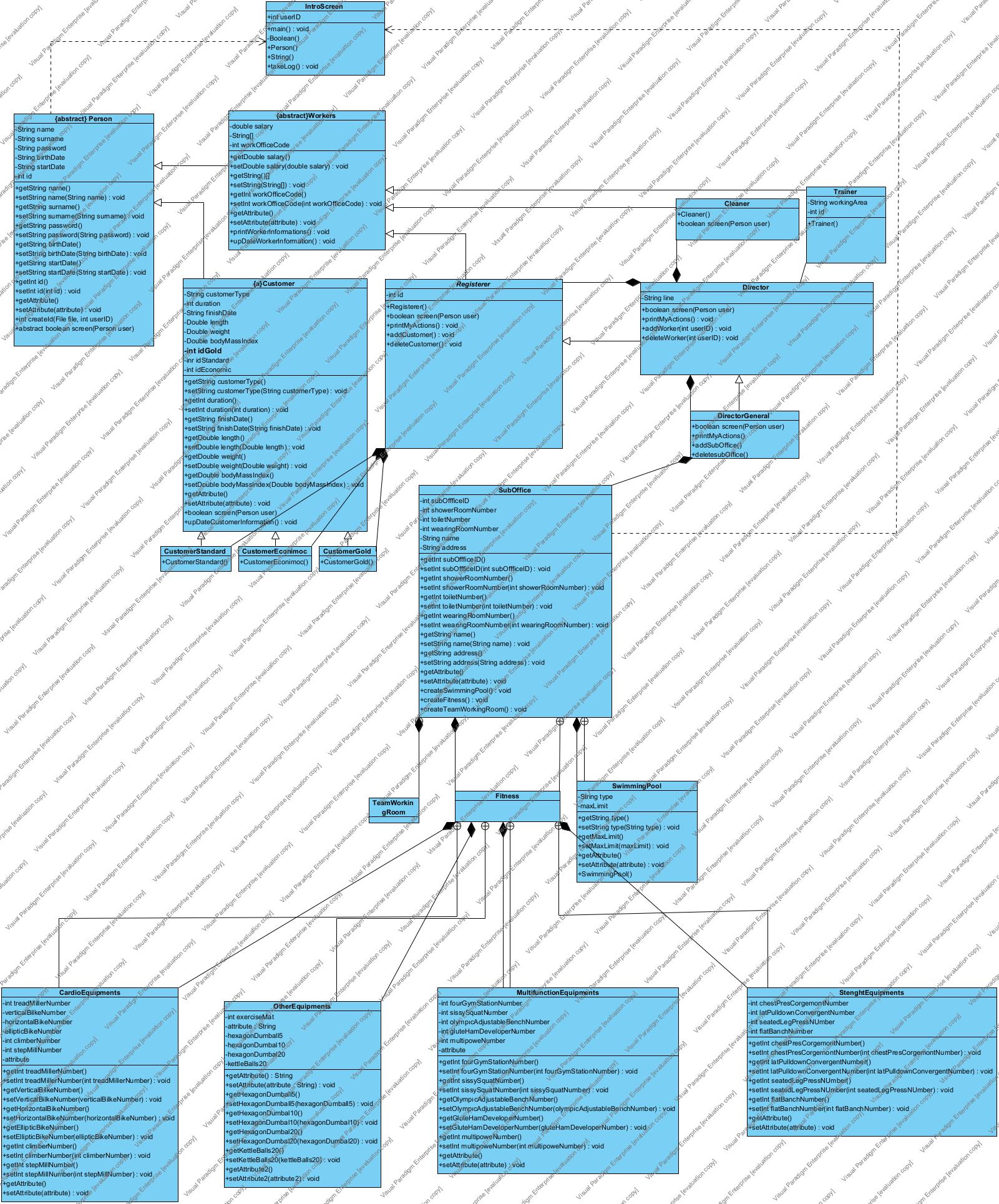
* 1. **KULLANILMASI PLANLANAN TEKNOLOJİLER**

Programlama dili olarak Java Standart Edition (SE) kullanıldı.

Geliştirme ortamı (IDE) olarak IntelliJ Idea kullanıldı.

UML diyagramı için Visual Paradigm masaüstü uygulaması kullanıldı.

* 1. **PROJENİN CLASS DİYAGRAMI**



1. **EK**



1. **KAYNAKÇA**

* <https://realleadergym.en.made-in-china.com/?keyword=gym%20fitness%20equipment&gclid=CjwKCAiA24SPBhB0EiwAjBgkhuiOr55rUBl5anaJOvvflBQHB4UEtoMi0iDoF6GB9ovbRCLP-7gd-hoCTbEQAvD_BwE>





## DirectorGeneral



## Director



## Registerer



## Trainer



## Cleaner



## Customer