Adı: Selim Eren

Soyadı: KAYA

Okul Numarası: 220601052

Bölüm: Bilgisayar Mühendisliği

Ders Adı: Bilgisayar Programlama 2

Konu: Python Final Projesi

Öğretim Görevlisi: Zekeriya Anıl Güven

MENEMEN, 2023

İÇİNDEKİLER

A. Ödev

1. Ödevin Amacı

Çalışanların performans ölçümü ve zam almasına, çalışmayanların uygun statüye önerilmesine dayalı bir proje geliştirilecektir. Projede insan, çalışan, işsiz, mavi yaka, beyaz yaka sınıfları oluşturulacaktır. Sınıflar arasında kalıtım olmasına dikkat edilmelidir.

- Çalışan ve İşsiz sınıfları İnsan sınıfından kalıtım yoluyla üretilmelidir.
- Mavi yaka, beyaz yaka sınıfları Çalışan sınıfından kalıtım yoluyla üretilmelidir.

2. Ödev için Gereksinimler

- Proje içinde main.py ve classlar'ın py dosyaları olmalıdır.
- İnsan sınıfında (Insan.py); tc_no, ad, soyad, yaş, cinsiyet, uyruk bilgileri private değişkenleri olarak bulunmalıdır.
    - Değişkenlere göre Initializer metot olmalıdır.
    - Tüm değişkenler için get/set metotları tanımlanmalıdır.
    - __str__ metodu ile kullanıcı bilgileri yazdırılmalıdır.
- İşsiz sınıfı için (Issiz.py) statüsü ("mavi yaka, beyaz yaka, yönetici") olan ve bu statülere ait geçmiş tecrübelerinin (yıl değeri) tutulduğu bir dictionary private değişkeni bulunmalıdır.
    - Değişkene göre Initializer metot olmalıdır.
    - Tüm gerekli değişkenler için get/set metotları tanımlanmalıdır.
    - En uygun statünün bulunması için statu_bul metodu yazınız (Dictionaryde girilen değerlere göre; yıl değerinde mavi yakanın etkisi %20, beyaz yakanın etkisi %35, yöneticinin etkisi %45 olarak hesaplayınız ve en yüksek çıkan değere ait statüyü ilgili değişkeninize atayınız. Bu değişkene farklı bir class'tan erişim sağlanabilmelidir.)
    - İlgili yerlerde try/except kullanılmalıdır.
    - __str__ metodu içinde kullanıcının ad, soyadı ve dictionary ile hesaplanan kişiye en uygun statü (public değişken ile yazdırılmamalı) yazdırılmalıdır.
- Çalışan sınıfı için (Calisan.py) sektör (kullanıcının "teknoloji, muhasebe, inşaat, diğer" seçenekleri girmesi sağlanmalı ve doğru girdiği kontrol edilmelidir), tecrübe (ay değeri) ve maaş değişkenleri private olarak tanımlanmalıdır.
    - Değişkenlere göre Initializer metot olmalıdır.
    - Tüm gerekli değişkenler için get/set metotları tanımlanmalıdır.
    - Çalışanın zam hakkını hesaplayan zam_hakki metodu yazılacaktır (2 sene öncesi tecrübesi olanın zam oranı önerisi 0'dır. 2-4 sene arası çalışan ise ve maaş 15000TL altıysa "maaş%tecrübe" sonucu zam oranı önerilecektir. 4 seneden fazla tecrübe varsa ve maaş 25000 altıysa "(maaş%tecrübe)/2" zam oranı

önerilecektir). Yeni maaş, eski maaş ile aynıysa eski maaş, yeni maaşa atanmalıdır.

- İlgili yerlerde try/except kullanılmalıdır.
- __str__ metodunda ad, soyad, tecrübe ve yeni maaşı (public değişken ile yazdırılmamalı) yazılmalıdır.

- Mavi yaka sınıfı için (MaviYaka.py) yıpranma payı (float: 0.2, 0.5 gibi değer almalıdır) değişkeni private olarak bulunmalıdır.
  - Değişkenlere göre Initializer metot olmalıdır.
  - Tüm gerekli değişkenler için get/set metotları tanımlanmalıdır.
  - Çalışanın zam hakkını hesaplayan zam_hakki metodu yazılacaktır (2 sene öncesi tecrübesi olanın zam oranı önerisi "yıpranma_payi*10" olacaktır. 2-4 sene arası çalışan ise ve maaş 15000TL altıysa "(maaş%tecrübe)/2 + (yıpranma_payi*10)" sonucu zam oranı önerilecektir. 4 seneden fazla tecrübe varsa ve maaş 25000 altıysa "(maaş%tecrübe)/3+ (yıpranma_payi*10)" zam oranı önerilecektir). Yeni maaş, eski maaş ile aynıysa eski maaş, yeni maaşa atanmalıdır.
  - İlgili yerlerde try/except kullanılmalıdır.
  - __str__ metodunda ad, soyad, tecrübe ve yeni maaşı (public değişken ile yazdırılmamalı) yazılmalıdır.
- Beyaz yaka sınıfı için (BeyazYaka.py) teşvik primi (500, 2500 gibi değer almalıdır) değişkeni private olarak bulunmalıdır.
  - Değişkenlere göre Initializer metot olmalıdır.
  - Tüm gerekli değişkenler için get/set metotları tanımlanmalıdır.
  - Çalışanın zam hakkını hesaplayan zam_hakki metodu yazılacaktır (2 sene öncesi tecrübesi olanın zam önerisi "teşvik_primi" olacaktır. 2-4 sene arası çalışan ise ve maaş 15000TL altıysa "(maaş%tecrübe)*5 + teşvik_primi" sonucu, zam olarak önerilecektir (önceki sınıflar gibi oran değil, bu sınıf zam miktarı). 4 seneden fazla tecrübe varsa ve maaş 25000 altıysa "(maaş%tecrübe)*4 + teşvik_primi" zam olarak önerilecektir). Yeni maaş, eski maaş ile aynıysa eski maaş, yeni maaşa atanmalıdır.
  - İlgili yerlerde try/except kullanılmalıdır.
  - __str__ metodunda ad, soyad, tecrübe ve yeni maaşı (public değişken ile yazdırılmamalı) yazılmalıdır.
- Main.py için;
  - İlgili yerlerde try/except kullanılmalıdır.
  - Sadece insan sınıfı için 2 nesne üretilmelidir ve bilgiler __str__ metotu aracılığıyla yazdırılmalıdır.
  - İşsiz sınıfı için 3 nesne üretilmelidir ve __str__ metotu ile ilgili bilgiler ekrana yazdırılmalıdır.
  - Çalışan sınıfı için 3 nesne üretilmelidir ve __str__ metotu ile ilgili bilgiler ekrana yazdırılmalıdır.

- Mavi yaka sınıfı için 3 nesne üretilmelidir ve __str__ metotu ile ilgili bilgiler ekrana yazdırılmalıdır.
- Beyaz yaka sınıfı için 3 nesne üretilmelidir ve __str__ metotu ile ilgili bilgiler ekrana yazdırılmalıdır.
- Çalışan, mavi yaka ve beyaz yaka nesnelerinin tüm değerlerinden ("çalışan, mavi yaka, beyaz yaka" nesne değeri ,tc_no, ad, soyad, yas, cinsiyet, uyruk, sektör, tecrübe (kaydederken yıla çeviriniz), maaş, yıpranma payı, teşvik primi, yeni maaş) bir pandas DataFrame oluşturunuz (excel, csv veya dictionary ile). Oluşturduğunuz DataFrame ile şu işlemleri gerçekleştiriniz:
  a) Bazı değişken değerleri diğer nesneler için boş olabilir, DataFrame için bu değerleri 0 atayınız.
  b) Çalışan, mavi yaka ve beyaz yaka için gruplandırarak tecrübe ve yeni maaş ortalamalarını her grup için hesaplayınız ve yazdırınız.
  c) Maaşı 15000TL üzerinde olanların toplam sayısını bulunuz.
  d) Yeni maaşa göre DataFrame'i küçükten büyüğe sıralayınız ve yazdırınız.
  e) Tecrübesi 3 seneden fazla olan Beyaz yakalıları bulunuz ve yazdırınız.
  f) Yeni maaşı 10000 TL üzerinde olanlar için; 2-5 satır arası olanları, tc_no ve yeni_maaş sütunlarını seçerek gösteriniz ve yazdırınız.
  g) Var olan DataFrame'den ad, soyad, sektör ve yeni maaşı içeren yeni bir DataFrame elde ediniz ve yazdırınız.

- Kodu her yazdığınızda GitHub üzerinden işlemeniz (commit etme) gerekmektedir. En az projeniz için iki branch (main, test gibi) oluşturmanız istenmektedir. GitHub'da yaptığınız işlemlerin geçmişini de rapora eklemelisiniz. Raporda GitHub profil linkinizi ve public oluşturulan projenizin linkini paylaşınız.

B. Kod Açıklamaları

1. Main.py

İnsan, İşsiz, Çalışan, Mavi Yaka ve Beyaz Yaka sınıflarına ait nesnelerin oluşturulup, bu nesnelere ait bilgilerin bir DataFrame'de tutulmasını sağlayan ve bu DataFrame üzerinden ödev gereksinimleri başlığı altında istenilen işlemlerin gerçekleşmesini sağlayan koddur.

2. Insan.py

Main.py dosyası içinde kullanılacak olan İnsan sınıfını içinde barındıran koddur. Bu sınıfta bireylerin tc_no, ad, soyad, yas, cinsiyet ve uyruk bilgileri tutulur. İstendiğinde çağırmak ve değiştirebilmek adına da her bir değişken için Get/Set metotları bulunur. Bilgilerin hepsini ekrana yazdırmak için sınıf içinde ayrıca __str__ metodu bulunur.

3. Issiz.py

Main.py dosyası içinde kullanılacak olan, İnsan sınıfının alt sınıfı olan İşsiz sınıfını içinde barındıran koddur. İnsan sınıfının alt sınıfı olmasına karşın içinde bireylerin mavi yaka, beyaz yaka ve yönetici statüsündeki tecrübe sürelerini farklı değişkenlere atayıp bir sözlükte tutar ve bu sözlüğü kullanarak kişiye en uygun statüyü hesaplayan statu_bul metodunu içinde barındırır. Bütün bu değişkenler için Get/Set metotları bulunur. __str__ metodu İnsan sınıfındakinin aksine bu sınıfta sırasıyla sadece ad, soyad ve uygun statü değerlerini yazdırır.

4. Calisan.py

Main.py dosyası içinde kullanılacak olan, İnsan sınıfının alt sınıfı olan Çalışan sınıfını içinde barındıran koddur. İnsan sınıfındaki değerlerle beraber ekstra olarak sektör, tecrübe ve maaş değerlerini de tutar. Bu ekstra değerlerden tecrübe ve maaş değerini kullanarak hak edilen zam oranını hesaplayan zam_hakki metoduna sahiptir. Bütün bu değerleri döndüren Get/Set metotlarına sahiptir. __str__ metodu ad, soyad, tecrübe ve zam sonrası maaş yani yeni maaş değerini ekrana yazdırır.

5. MaviYaka.py

Main.py dosyasının içinde kullanılacak olan, Çalışan sınıfının alt sınıfı olan Mavi Yaka sınıfını içinde barındıran koddur. Çalışan sınıfındaki değerlerle birlikte ekstra olarak yıpranma payı değerini de tutar. Tuttuğu bütün değerlere ait Get/Set metotları vardır. Çalışan sınıfındaki zam_hakki metodunun kullandığı tecrübe ve maaş değerlerine ekstra olarak yıpranma payı değerini de kullanarak hak edilen zam oranını hesaplayan zam_hakki metoduna sahiptir. __str__ metodu, Çalışan sınıfında yazdırılan değerlerin aynısını ekrana yazdırır.

6. BeyazYaka.py

Main.py dosyasının içinde kullanılacak olan, Çalışan sınıfının alt sınıfı olan Beyaz Yaka sınıfını içinde barındıran koddur. Çalışan sınıfındaki değerlerle birlikte ekstra olarak teşvik primi değerini de tutar. Tuttuğu bütün değerlere ait Get/Set metotları vardır. Çalışan sınıfındaki zam_hakki metodunun kullandığı tecrübe ve maaş değerlerine ekstra olarak teşvik primi değerini de kullanarak hak edilen zam miktarını hesaplayan zam_hakki metoduna sahiptir. __str__ metodu, Çalışan sınıfında yazdırılan değerlerin aynısını ekrana yazdırır.

C. Kod Çıktıları

1. İnsan Sınıfına ait nesne bilgilerinin yazdırılması

```
* * * İnsan Sınıfına Ait Nesne Bilgileri * * *
TC No = 10013168314
Ad = Selim Eren
Soyad = White
Yaş = 18
Cinsiyet = Erkek
Uyruk = Türk
------------------------
TC No = 23791307912
Ad = Umut Şeref
Soyad = Varga
Yaş = 20
Cinsiyet = Erkek
Uyruk = Türk
------------------------
```

2. İşsiz Sınıfına ait nesne bilgilerinin yazdırılması

```
* * * İşsiz Sınıfına Ait Nesne Bilgileri * * *
Ad = Yusuf
Soyad = Salamanca
En Uygun Statü = Beyaz Yaka
------------------------
Ad = Mehmet Igor
Soyad = Dobarov
En Uygun Statü = Yönetici
------------------------
Ad = Beyza
Soyad = Karakoç
En Uygun Statü = Mavi Yaka
------------------------
```

3. Çalışan Sınıfına ait nesne bilgilerinin yazdırılması

```
* * * Çalışan Sınıfına Ait Nesne Bilgileri * * *
Ad = Hüseyin
Soyad = Garcia
Tecrübe = 29 Ay
Yeni Maaş = 9900.0
------------------------
Ad = Sofia
Soyad = Vetra
Tecrübe = 35 Ay
Yeni Maaş = 15600.0
------------------------
Ad = Vittoria
Soyad = Morel
Tecrübe = 45 Ay
Yeni Maaş = 14700.0
------------------------
```

4. Mavi Yaka Sınıfına ait nesne bilgilerinin yazdırılması

```
* * * Mavi Yaka Sınıfına Ait Nesne Bilgileri * * *
Ad = İlker
Soyad = Tunç
Tecrübe = 57 Ay
Yeni Maaş = 26143.33
------------------------
Ad = Nur
Soyad = Güler
Tecrübe = 32 Ay
Yeni Maaş = 17000.00
------------------------
Ad = Sinem
Soyad = Ayaz
Tecrübe = 27 Ay
Yeni Maaş = 13080.00
------------------------
```

5. Beyaz Yaka Sınıfına ait nesne bilgilerinin yazdırılması

```
* * * Beyaz Yaka Sınıfına Ait Nesne Bilgileri * * *
Ad = Gustavo
Soyad = Fring
Tecrübe = 80 Ay
Yeni Maaş = 27000.00
------------------------
Ad = Michael
Soyad = Ehrmantraut
Tecrübe = 56 Ay
Yeni Maaş = 22032.00
------------------------
Ad = Jessie
Soyad = Pinkman
Tecrübe = 30 Ay
Yeni Maaş = 13500.00
------------------------
```

6. DataFrame Çıktıları

a. Statü ortalamaları

```
* * * STATÜ ORTALAMALARI * * *

----Çalışan Statülü Bireylerin----
Ortalama Tecrübesi = 2.33 Yıl
Ortalama Yeni Maaşı = 13400.00 TL

----Mavi Yaka Statülü Bireylerin----
Ortalama Tecrübesi = 2.67 Yıl
Ortalama Yeni Maaşı = 18741.00 TL

----Beyaz Yaka Statülü Bireylerin----
Ortalama Tecrübesi = 4.00 Yıl
Ortalama Yeni Maaşı = 20844.00 TL
```

b. Maaşı 15000 TL üzeri olan birey sayısı

```
MAAŞI 15000TL ÜZERİ OLAN BİREY SAYISI = 4
```

c. DataFrame'in yeni maaşa göre küçükten büyüğe sıralanmış hali

```
----------DATAFRAME'İN YENİ MAAŞA GÖRE KÜÇÜKTEN BÜYÜĞE DOĞRU SIRALANMIŞ HALİ----------

  Nesne Değeri       TC No       Ad        Soyad  Yaş Cinsiyet     Uyruk     Sektör  Tecrübe(Yıl)   Maaş  Yıpranma Payı  Teşvik Primi  Yeni Maaş
0      Çalışan  15928406635  Hüseyin       Garcia   18    Erkek   Fransız     inşaat             2   9000            0.0             0       9900
5    Mavi Yaka  64153045957    Sinem         Ayaz   22    Kadın      Türk   Muhasebe             2  12000            0.3             0      13080
8   Beyaz Yaka  52587415063   Jessie      Pinkman   25    Erkek  Amerikan   Muhasebe             2  12000            0.0          1500      13500
2      Çalışan  15823947025 Vittoria        Morel   24    Kadın   Fransız  Teknoloji             3  14000            0.0             0      14700
1      Çalışan  73025904169    Sofia        Vetra   20    Kadın  İspanyol      Diğer             2  12000            0.0             0      15600
4    Mavi Yaka  49177620539      Nur        Güler   25    Kadın      Türk  Teknoloji             2  17000            0.2             0      17000
7   Beyaz Yaka  29063742415  Michael  Ehrmantraut   60    Erkek  Amerikan      Diğer             4  20000            0.0          2000      22032
3    Mavi Yaka  75331020462     İlker         Tunç   32    Erkek      Türk   Muhasebe             4  23000            0.4             0      26143
6   Beyaz Yaka  38234957268   Gustavo        Fring   40    Erkek    Şilili      Diğer             6  24000            0.0          3000      27000
```

d. Tecrübesi 3 seneden fazla olan Beyaz Yakalılar

```
--------TECRÜBESİ 3 SENEDEN FAZLA OLAN BEYAZ YAKALILAR--------
1. Gustavo Fring
2. Michael Ehrmantraut
```

e. Yeni maaşı 10000 TL üzeri ve sırası 2-5 arası olanlar

```
--------YENİ MAAŞI 10000TL ÜZERİ VE SIRASI 2-5 ARASI OLANLAR--------
Sıra No = 2
TC No         15823947025
Yeni Maaş           14700

Sıra No = 3
TC No         75331020462
Yeni Maaş           26143

Sıra No = 4
TC No         49177620539
Yeni Maaş           17000

Sıra No = 5
TC No         64153045957
Yeni Maaş           13080
```

f. Sadece ad, soyad, sektör ve yeni maaş içeren DataFrame

```
--------SADECE AD, SOYAD, SEKTÖR VE YENİ MAAŞI İÇEREN YENİ DATAFRAME--------
          Ad        Soyad     Sektör  Yeni Maaş
0   Hüseyin       Garcia     inşaat       9900
1     Sofia        Vetra      Diğer      15600
2  Vittoria        Morel  Teknoloji      14700
3     İlker         Tunç   Muhasebe      26143
4       Nur         Güler  Teknoloji      17000
5     Sinem         Ayaz   Muhasebe      13080
6   Gustavo        Fring      Diğer      27000
7   Michael  Ehrmantraut      Diğer      22032
8    Jessie      Pinkman   Muhasebe      13500

Process finished with exit code 0
```

g. DataFrame'in orijinal hali

	Nesne Değeri	TC No	Ad	Soyad	Yaş	Cinsiyet	Uyruk	Sektör	Tecrübe(Yıl)	Maaş	Yıpranma Payı	Teşvik Primi	Yeni Maaş
0	Çalışan	15928406635	Hüseyin	Garcia	18	Erkek	Fransız	inşaat	2	9000	0.0	0	9900
1	Çalışan	73025904169	Sofia	Vetra	20	Kadın	İspanyol	Diğer	2	12000	0.0	0	15600
2	Çalışan	15823947025	Vittoria	Morel	24	Kadın	Fransız	Teknoloji	3	14000	0.0	0	14700
3	Mavi Yaka	75331020462	İlker	Tunç	32	Erkek	Türk	Muhasebe	4	23000	0.4	0	26143
4	Mavi Yaka	49177620539	Nur	Güler	25	Kadın	Türk	Teknoloji	2	17000	0.2	0	17000
5	Mavi Yaka	64153045957	Sinem	Ayaz	22	Kadın	Türk	Muhasebe	2	12000	0.3	0	13080
6	Beyaz Yaka	38234957268	Gustavo	Fring	40	Erkek	Şilili	Diğer	6	24000	0.0	3000	27000
7	Beyaz Yaka	29063742415	Michael	Ehrmantraut	60	Erkek	Amerikan	Diğer	4	20000	0.0	2000	22032
8	Beyaz Yaka	52587415063	Jessie	Pinkman	25	Erkek	Amerikan	Muhasebe	2	12000	0.0	1500	13500

D. Sözde Kodlar

1. Main Fonksiyonu

1. BAŞLA
2. ÇAĞIR Insan KÜTÜPHANESİ
3. ÇAĞIR Issiz KÜTÜPHANESİ
4. ÇAĞIR Calisan KÜTÜPHANESİ
5. ÇAĞIR MaviYaka KÜTÜPHANESİ
6. ÇAĞIR BeyazYaka KÜTÜPHANESİ
7. ÇAĞIR pandas KÜTÜPHANESİ
8. insan_nesne_1 = Insan.Insan(10013168314, "Selim Eren", "White", 18, "Erkek", "Türk")
9. insan_nesne_2 = Insan.Insan(23791307912, "Umut seref", "Varga", 20, "Erkek", "Türk")
10. issiz_nesne_1 = Issiz.Issiz(24697482015, "Yusuf", "Salamanca", 21, "Erkek", "İspanyol", 5, 3, 0)
11. issiz_nesne_2 = Issiz.Issiz(48201672483, "Mehmet Igor", "Dobarov", 23, "Erkek", "Rus", 5, 0, 3)
12. issiz_nesne_3 = Issiz.Issiz(28604627451, "Beyza", "Karakoç", 19, "Kadın", "Türk", 4, 1, 0)
13. calisan_nesne_1 = Calisan.Calisan(15928406635, "Hüseyin", "Garcia", 18, "Erkek", "Fransız", "inşaat", 29, 9000)
14. calisan_nesne_2 = Calisan.Calisan(73025904169, "Sofia", "Vetra", 20, "Kadın", "İspanyol", "Diğer", 35, 12000)
15. calisan_nesne_3 = Calisan.Calisan(15823947025, "Vittoria", "Morel", 24, "Kadın", "Fransız", "Teknoloji", 45, 14000)
16. mavi_yaka_nesne_1 = MaviYaka.MaviYaka(75331020462, "İlker", "Tunç", 32, "Erkek", "Türk", "Muhasebe", 57, 23000, 0.4)
17. mavi_yaka_nesne_2 = MaviYaka.MaviYaka(49177620539, "Nur", "Güler", 25, "Kadın", "Türk", "Teknoloji", 32, 17000, 0.2)
18. mavi_yaka_nesne_3 = MaviYaka.MaviYaka(64153045957, "Sinem", "Ayaz", 22, "Kadın", "Türk", "Muhasebe", 27, 12000, 0.3)
19. beyaz_yaka_nesne_1 = BeyazYaka.BeyazYaka(38234957268, "Gustavo", "Fring", 40, "Erkek", "Şilili", "Diğer", 80, 24000, 3000)
20. beyaz_yaka_nesne_2 = BeyazYaka.BeyazYaka(29063742415, "Michael", "Ehrmantraut", 60, "Erkek", "Amerikan", "Diğer", 56, 20000, 2000)
21. beyaz_yaka_nesne_3 = BeyazYaka.BeyazYaka(52587415063, "Jessie", "Pinkman", 25, "Erkek", "Amerikan", "Muhasebe", 30, 12000, 1500)
22. YAZDIR("\n* * * İnsan Sınıfına Ait Nesne Bilgileri * * *")
23. YAZDIR(insan_nesne_1, end="\n----------\n")
24. YAZDIR(insan_nesne_2, end="\n----------\n")
25. YAZDIR("\n* * * İşsiz Sınıfına Ait Nesne Bilgileri * * *")
26. YAZDIR(issiz_nesne_1, end="\n----------\n")
27. YAZDIR(issiz_nesne_2, end="\n----------\n")

```
28. YAZDIR(issiz_nesne_3, end="\n-----------------------\n")
29. YAZDIR("\n* * * Çalışan Sınıfına Ait Nesne Bilgileri * * *")
30. YAZDIR(calisan_nesne_1, end="\n-----------------------\n")
31. YAZDIR(calisan_nesne_2, end="\n-----------------------\n")
32. YAZDIR(calisan_nesne_3, end="\n-----------------------\n")
33. YAZDIR("\n* * * Mavi Yaka Sınıfına Ait Nesne Bilgileri * * *")
34. YAZDIR(mavi_yaka_nesne_1, end="\n-----------------------\n")
35. YAZDIR(mavi_yaka_nesne_2, end="\n-----------------------\n")
36. YAZDIR(mavi_yaka_nesne_3, end="\n-----------------------\n")
37. YAZDIR("\n* * * Beyaz Yaka Sınıfına Ait Nesne Bilgileri * * *")
38. YAZDIR(beyaz_yaka_nesne_1, end="\n-----------------------\n")
39. YAZDIR(beyaz_yaka_nesne_2, end="\n-----------------------\n")
40. YAZDIR(beyaz_yaka_nesne_3, end="\n-----------------------\n")
41. dictionary = {
                    "Nesne Değeri": ["Çalışan", " Çalışan ", " Çalışan”,
                    "Mavi Yaka", "Mavi Yaka", "Mavi Yaka",
                    "Beyaz Yaka", "Beyaz Yaka", "Beyaz Yaka"],
                    "TC No": [calisan_nesne_1.get_tc_no(),
                    calisan_nesne_2.get_tc_no(),          calisan_nesne_3.get_tc_no(),
                    mavi_yaka_nesne_1.get_tc_no(), mavi_yaka_nesne_2.get_tc_no(),
                    mavi_yaka_nesne_3.get_tc_no(),
                    beyaz_yaka_nesne_1.get_tc_no(), beyaz_yaka_nesne_2.get_tc_no(),
                    beyaz_yaka_nesne_3.get_tc_no()],
                    "Ad": [calisan_nesne_1.get_ad(),    calisan_nesne_2.get_ad(),
                            calisan_nesne_3.get_ad(),
                    mavi_yaka_nesne_1.get_ad(),    mavi_yaka_nesne_2.get_ad(),
                    mavi_yaka_nesne_3.get_ad(), beyaz_yaka_nesne_1.get_ad(),
                    beyaz_yaka_nesne_2.get_ad(), beyaz_yaka_nesne_3.get_ad()],
                    "Soyad": [calisan_nesne_1.get_soyad(), calisan_nesne_2.get_soyad(),
                    calisan_nesne_3.get_soyad(), mavi_yaka_nesne_1.get_soyad(),
                    mavi_yaka_nesne_2.get_soyad(), mavi_yaka_nesne_3.get_soyad(),
                    beyaz_yaka_nesne_1.get_soyad(), beyaz_yaka_nesne_2.get_soyad(),
                    beyaz_yaka_nesne_3.get_soyad()],
                    "Yaş": [calisan_nesne_1.get_yas(), calisan_nesne_2.get_yas(),
                    calisan_nesne_3.get_yas(),     mavi_yaka_nesne_1.get_yas(),
                    mavi_yaka_nesne_2.get_yas(), mavi_yaka_nesne_3.get_yas(),
                    beyaz_yaka_nesne_1.get_yas(), beyaz_yaka_nesne_2.get_yas(),
                    beyaz_yaka_nesne_3.get_yas()],
                    "Cinsiyet": [calisan_nesne_1.get_cinsiyet(),
                    calisan_nesne_2.get_cinsiyet(), calisan_nesne_3.get_cinsiyet(),
                    mavi_yaka_nesne_1.get_cinsiyet(), mavi_yaka_nesne_2.get_cinsiyet(),
                    mavi_yaka_nesne_3.get_cinsiyet(), beyaz_yaka_nesne_1.get_cinsiyet(),
                    beyaz_yaka_nesne_2.get_cinsiyet(), beyaz_yaka_nesne_3.get_cinsiyet()],
```

```
                    "Uyruk": [calisan_nesne_1.get_uyruk(), calisan_nesne_2.get_uyruk(),
                    calisan_nesne_3.get_uyruk(), mavi_yaka_nesne_1.get_uyruk(),
                    mavi_yaka_nesne_2.get_uyruk(), mavi_yaka_nesne_3.get_uyruk(),
                    beyaz_yaka_nesne_1.get_uyruk(), beyaz_yaka_nesne_2.get_uyruk(),
                    beyaz_yaka_nesne_3.get_uyruk()],

                    "Sektör": [calisan_nesne_1.get_sektor(), calisan_nesne_2.get_sektor(),
                    calisan_nesne_3.get_sektor(), mavi_yaka_nesne_1.get_sektor(),
                    mavi_yaka_nesne_2.get_sektor(), mavi_yaka_nesne_3.get_sektor(),
                    beyaz_yaka_nesne_1.get_sektor(), beyaz_yaka_nesne_2.get_sektor(),
                    beyaz_yaka_nesne_3.get_sektor()],

                    "Tecrübe(Yıl)": [int(calisan_nesne_1.get_tecrube_ay()/12),
                    int(calisan_nesne_2.get_tecrube_ay()/12),
                    int(calisan_nesne_3.get_tecrube_ay()/12),
                    int(mavi_yaka_nesne_1.get_tecrube_ay()/12),
                    int(mavi_yaka_nesne_2.get_tecrube_ay()/12),
                    int(mavi_yaka_nesne_3.get_tecrube_ay()/12),
                    int(beyaz_yaka_nesne_1.get_tecrube_ay()/12),
                    int(beyaz_yaka_nesne_2.get_tecrube_ay()/12),
                    int(beyaz_yaka_nesne_3.get_tecrube_ay()/12)],

                    "Maaş": [calisan_nesne_1.get_maas(), calisan_nesne_2.get_maas(),
                    calisan_nesne_3.get_maas(), mavi_yaka_nesne_1.get_maas(),
                    mavi_yaka_nesne_2.get_maas(), mavi_yaka_nesne_3.get_maas(),
                    beyaz_yaka_nesne_1.get_maas(), beyaz_yaka_nesne_2.get_maas(),
                    beyaz_yaka_nesne_3.get_maas()],

                    "Yıpranma Payı": [0, 0, 0, mavi_yaka_nesne_1.get_yipranma_payi(),
                    mavi_yaka_nesne_2.get_yipranma_payi(),
                    mavi_yaka_nesne_3.get_yipranma_payi(),                    0, 0, 0],

                    "Teşvik Primi": [0, 0, 0, 0, 0, 0, beyaz_yaka_nesne_1.get_tesvik_primi(),
                    beyaz_yaka_nesne_2.get_tesvik_primi(),
                    beyaz_yaka_nesne_3.get_tesvik_primi()],

                    "Yeni Maaş": [int(calisan_nesne_1.get_yeni_maas()),
                    int(calisan_nesne_2.get_yeni_maas()),
                    int(calisan_nesne_3.get_yeni_maas()),
                    int(mavi_yaka_nesne_1.get_yeni_maas()),
                    int(mavi_yaka_nesne_2.get_yeni_maas()),
                    int(mavi_yaka_nesne_3.get_yeni_maas()),
                    int(beyaz_yaka_nesne_1.get_yeni_maas()),
                    int(beyaz_yaka_nesne_2.get_yeni_maas()),
                    int(beyaz_yaka_nesne_3.get_yeni_maas())],

                }
```

42. dataframe = pandas.DataFrame(data=dictionary)
43. ortalama_tecrube = {"Çalışan": 0, "Mavi Yaka": 0, "Beyaz Yaka": 0}
44. ortalama_yeni_maas = {"Çalışan": 0, "Mavi Yaka": 0, "Beyaz Yaka": 0}
45. birey_sayisi = {"Çalışan": 0, "Mavi Yaka": 0, "Beyaz Yaka": 0}
46. DENE:
    - 46.1. FOR i IN range(len(dataframe)):
        - 46.1.1. EĞER dataframe.iloc[i, 0] == "Çalışan" İSE:
            - 46.1.1.1. ortalama_tecrube["Çalışan"] += dataframe.iloc[i, 8]

46.1.1.2. ortalama_yeni_maas["Çalışan"] += dataframe.iloc[i,12]

46.1.1.3. birey_sayisi["Çalışan"] += 1

46.1.2. DEĞİLSE, EĞER dataframe.iloc[i, 0] == "Mavi Yaka" İSE:

46.1.2.1. ortalama_tecrube["Mavi Yaka"] += dataframe.iloc[i, 8]

46.1.2.2. ortalama_yeni_maas["Mavi Yaka"] += dataframe.iloc[i,12]

46.1.2.3. birey_sayisi["Mavi Yaka"] += 1

46.1.3. DEĞİLSE, EĞER dataframe.iloc[i, 0] == "Beyaz Yaka" İSE:

46.1.3.1. ortalama_tecrube["Beyaz Yaka"] += dataframe.iloc[i, 8]

46.1.3.2. ortalama_yeni_maas["Beyaz Yaka"] += dataframe.iloc[i,12]

46.1.3.3. birey_sayisi["Beyaz Yaka"] += 1

46.1.4. GİT 45.1. ADIM

47. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:

47.1. YAZDIR(f"HATA! Hata sebebi: {exception}")

48. DENE:

48.1. ortalama_tecrube["Çalışan"] = ortalama_tecrube["Çalışan"] / birey_sayisi["Çalışan"]

48.2. ortalama_yeni_maas["Çalışan"] = ortalama_yeni_maas["Çalışan"] / birey_sayisi["Çalışan"]

48.3. ortalama_tecrube["Mavi Yaka"] = ortalama_tecrube["Mavi Yaka"] / birey_sayisi["Mavi Yaka"]

48.4. ortalama_yeni_maas["Mavi Yaka"] = ortalama_yeni_maas["Mavi Yaka"] / birey_sayisi["Mavi Yaka"]

48.5. ortalama_tecrube["Beyaz Yaka"] = ortalama_tecrube["Beyaz Yaka"] / birey_sayisi["Beyaz Yaka"]

48.6. ortalama_yeni_maas["Beyaz Yaka"] = ortalama_yeni_maas["Beyaz Yaka"] / birey_sayisi["Beyaz Yaka"]

49. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:

49.1. YAZDIR(f"HATA! Hata sebebi: {exception}")

50. statu_listesi = ("Çalışan", "Mavi Yaka", "Beyaz Yaka")

51. YAZDIR("\n\n* * * STATÜ ORTALAMALARI * * *")

52. DENE:

52.1. FOR i IN statu_listesi:

52.1.1. YAZDIR(f"\n----{i} Statülü Bireylerin----"

f"\nOrtalama Tecrübesi = {ortalama_tecrube[i]:.2f} Yil"

f"\nOrtalama Yeni Maaşı = {ortalama_yeni_maas[i]:.2f} TL")

52.1.2. GİT 51.1. ADIM

53. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:

53.1. YAZDIR(f"HATA! Hata sebebi: {exception}")

54. on_bes_bin_uzeri_maas_alan_sayisi = 0

55. DENE:

55.1. FOR i IN range(len(dataframe)):

55.1.1. EĞER dataframe.iloc[i, 9] > 15000 İSE:

55.1.1.1. on_bes_bin_uzeri_maas_alan_sayisi += 1

55.1.2. GİT 54.1. ADIM

55.2. YAZDIR(f"\n\nMAAŞI 15000TL ÜZERİ OLAN BİREY SAYISI = {on_bes_bin_uzeri_maas_alan_sayisi}")

56. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:

56.1. YAZDIR(f"HATA! Hata sebebi: {exception}")

57. DENE:

57.1. sirali_dataframe = dataframe.sort_values("Yeni Maaş", ascending=True)

57.2. YAZDIR("\n\n----------DATAFRAME'iN YENİ MAAŞA GÖRE KÜÇÜKTEN BÜYÜĞE DOĞRU SIRALANMIŞ HALİ----------\n")

57.3. YAZDIR(sirali_dataframe.to_string())

58. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:

58.1. YAZDIR(f"HATA! Hata sebebi: {exception}")

59. uc_sene_ve_beyaz_yaka = []

60. DENE:

60.1. FOR i IN range(len(dataframe)):

60.1.1. EĞER dataframe.iloc[i, 8] > 3 and dataframe.iloc[i, 0] == "Beyaz Yaka" İSE:

60.1.1.1. uc_sene_ve_beyaz_yaka.append(i)

60.1.2. GİT 59.1. ADIM

60.2. YAZDIR("\n\n--------TECRÜBESİ 3 SENEDEN FAZLA OLAN BEYAZ YAKALILAR--------")

60.3. sira = 1

60.4. FOR i IN uc_sene_ve_beyaz_yaka:

60.4.1. YAZDIR(f"{sira}. "

f"{dataframe.loc[i][2]} "

f"{dataframe.loc[i][3]}")

60.4.2. sira += 1

60.4.3. GİT 59.4. ADIM

61. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:

61.1. YAZDIR(f"HATA! Hata sebebi: {exception}")

62. YAZDIR("\n\n--------YENİ MAAŞI 10000TL ÜZERİ VE SIRASI 2-5 ARASI OLANLAR--------")
63. DENE:
    63.1. FOR i IN range(len(dataframe)):
        63.1.1. EĞER dataframe.iloc[i, 12] > 10000 VE 2 <= i <= 5 İSE:
            63.1.1.1. bilgiler = dataframe.iloc[i, 1:13:11]
            63.1.1.2. (f"SIra No = {i}\n"
                f"{bilgiler.to_string()}\n")
        63.1.2. GİT 62.1. ADIM
64. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:
    64.1. YAZDIR(f"HATA! Hata sebebi: {exception}")
65. YAZDIR("\n\n--------SADECE AD, SOYAD, SEKTÖR VE YENİ MAAŞI İÇEREN YENİ DATAFRAME--------")
66. DENE:
    66.1. yeni_dataframe = dataframe[["Ad", "Soyad", "Sektör", "Yeni Maaş"]].copy()
    66.2. YAZDIR(yeni_dataframe.to_string())
67. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:
    67.1. YAZDIR(f"HATA! Hata sebebi: {exception}")
68. BİTİR

2. İnsan Sınıfı

a. __init__ Metodu

__init__(self, tc_no, ad, soyad, yas, cinsiyet, uyruk):

1. BAŞLA
2. self.__tc_no = tc_no
3. self.__ad = ad
4. self.__soyad = soyad
5. self.__yas = yas
6. self.__cinsiyet = cinsiyet
7. self.__uyruk = uyruk
8. BİTİR

b. Private değişkenlerin Get/Set Metodları

get_tc_no(self):

1. BAŞLA
2. DÖNDÜR self.__tc_no
3. BİTİR

set_tc_no(self, tc_no):

1. BAŞLA
2. self.__tc_no = tc_no
3. BİTİR

get_ad(self):

1. BAŞLA
2. DÖNDÜR self.__ad
3. BİTİR

set_ad(self, ad):

1. BAŞLA
2. self.__ad = ad
3. BİTİR

get_soyad(self):

1. BAŞLA
2. DÖNDÜR self.__soyad
3. BİTİR

set_soyad(self, soyad):

1. BAŞLA
2. self.__soyad = soyad
3. BİTİR

get_yas(self):

1. BAŞLA
2. DÖNDÜR self.__yas
3. BİTİR

set_yas(self, yas):

1. BAŞLA
2. self.__yas = yas
3. BİTİR

get_cinsiyet(self):

1. BAŞLA
2. DÖNDÜR self.__cinsiyet
3. BİTİR

set_cinsiyet(self, cinsiyet):

1. BAŞLA
2. self.__cinsiyet = cinsiyet
3. BİTİR

get_uyruk(self):

1. BAŞLA
2. DÖNDÜR self.__uyruk
3. BİTİR

set_uyruk(self, uyruk):

1. BAŞLA
2. self.__uyruk = uyruk
3. BİTİR

c.__str__ Metodu

__str__(self):

1. BAŞLA
2. DÖNDÜR f"TC No = {self.__tc_no}" \
   f"\nAd = {self.__ad}" \
   f"\nSoyad = {self.__soyad}" \
   f"\nYaş = {self.__yas}" \
   f"\nCinsiyet = {self.__cinsiyet}" \
   f"\nUyruk = {self.__uyruk}"
3. BİTİR

3. İşsiz Sınıfı

a. __init_ Metodu

__init__(self, tc_no, ad, soyad, yas, cinsiyet, uyruk, mavi_yaka_yil, beyaz_yaka_yil, yonetici_yil):

1. BAŞLA
2. super().__init__(tc_no, ad, soyad, yas, cinsiyet, uyruk)
3. self.__mavi_yaka_yil = mavi_yaka_yil
4. self.__beyaz_yaka_yil = beyaz_yaka_yil
5. self.__yonetici_yil = yonetici_yil
6. self.__uygun_statu = "Belirlenmedi"
7. self.__statu_dictionary = {"mavi_yaka": self.__mavi_yaka_yil,
   "beyaz_yaka": self.__beyaz_yaka_yil,
   "yonetici": self.__yonetici_yil}
8. BİTİR

b. Private değişkenlerin Get/Set Metodları

get_mavi_yaka_yil(self):

1. BAŞLA
2. DÖNDÜR self.__mavi_yaka_yil
3. BİTİR

set_mavi_yaka_yil(self, mavi_yaka_yil):

1. BAŞLA
2. self.__mavi_yaka_yil = mavi_yaka_yil
3. self.__statu_dictionary["mavi_yaka"] = mavi_yaka_yil
4. BİTİR

get_beyaz_yaka_yil(self):

1. BAŞLA
2. DÖNDÜR self.__beyaz_yaka_yil
3. BİTİR

set_beyaz_yaka_yil(self, beyaz_yaka_yil):

1. BAŞLA
2. self.__beyaz_yaka_yil = beyaz_yaka_yil
3. self.__statu_dictionary["beyaz_yaka"] = beyaz_yaka_yil
4. BİTİR

get_yonetici_yil(self):

1. BAŞLA
2. DÖNDÜR self.__yonetici_yil
3. BİTİR

set_yonetici_yil(self, yonetici_yil):

1. BAŞLA
2. self.__yonetici_yil = yonetici_yil
3. self.__statu_dictionary["yonetici"] = yonetici_yil
4. BİTİR

get_uygun_statu(self):

1. BAŞLA
2. DÖNDÜR self.__uygun_statu
3. BİTİR

set_uygun_statu(self, uygun_statu):

1. BAŞLA
2. self.__uygun_statu = uygun_statu
3. BİTİR

c. statu_bul Metodu

statu_bul(self):

1. BAŞLA
2. TRY:
   2.1. mavi_yaka_deger = (float(self.__statu_dictionary["mavi_yaka"]) / 100) * 20
   2.2. beyaz_yaka_deger = (float(self.__statu_dictionary["beyaz_yaka"]) / 100) * 35
   2.3. yonetici_deger = (float(self.__statu_dictionary["yonetici"]) / 100) * 45
   2.4. EĞER mavi_yaka_deger > beyaz_yaka_deger and mavi_yaka_deger > yonetici_deger İSE:
      2.4.1. self.__uygun_statu = "Mavi Yaka"
   2.5. DEĞİLSE, EĞER beyaz_yaka_deger >= mavi_yaka_deger and beyaz_yaka_deger > yonetici_deger İSE:
      2.5.1. self.__uygun_statu = "Beyaz Yaka"
   2.6. DEĞİLSE, EĞER yonetici_deger >= mavi_yaka_deger and yonetici_deger >= beyaz_yaka_deger İSE:
      2.6.1. self.__uygun_statu = "Yönetici"
3. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:
   3.1. YAZDIR(f"HATA! Hata sebebi: {exception}")
4. BİTİR

d. __str__ Metodu

__str__(self):

1. BAŞLA
2. self.statu_bul()
3. DÖNDÜR f"Ad = {self.get_ad()}" \
   f"\nSoyad = {self.get_soyad()}" \
   f"\nEn Uygun Statü = {self.get_uygun_statu()}"
4. BİTİR

4. Çalışan Sınıfı

a. __init__ Metodu

__init__(self, tc_no, ad, soyad, yas, cinsiyet, uyruk, sektor, tecrube_ay, maas):

1. BAŞLA
2. super().__init__(tc_no, ad, soyad, yas, cinsiyet, uyruk)
3. sektor_liste = ["teknoloji", "muhasebe", "inşaat", "diğer"]
4. EĞER sektor.lower() DEĞİLSE sektor_liste İÇİNDE :
   4.1. YAZDIR("Geçersiz bir sektör girdiniz.\n")
   4.2. sektor = GİRDİ("Geçerli bir sektör giriniz.\n"
   "Seçenekler = teknoloji, muhasebe, inşaat, diğer\n"
   "Sektör = ")
   4.3. GİT 4. ADIM
5. self.__sektor = sektor
6. self.__tecrube_ay = tecrube_ay
7. self.__maas = maas
8. self.__yeni_maas = "Belirlenmedi"
9. self.__zam_orani_oneri = 0
10. self.__hakedilen_zam = 0
11. BİTİR

b. Private değişkenlerin Get/Set Metodları

get_sektor(self):

1. BAŞLA
2. DÖNDÜR self.__sektor
3. BİTİR

set_sektor(self, sektor):

1. BAŞLA
2. self.__sektor = sektor
3. BİTİR

get_tecrube_ay(self):

1. BAŞLA
2. DÖNDÜR self.__tecrube_ay
3. BİTİR

set_tecrube_ay(self, tecrube_ay):

1. BAŞLA
2. self.__tecrube_ay = tecrube_ay
3. BİTİR

get_maas(self):

1. BAŞLA
2. DÖNDÜR self.__maas
3. BİTİR

set_maas(self, maas):

1. BAŞLA
2. self.__maas = maas
3. BİTİR

get_yeni_maas(self):

1. BAŞLA
2. DÖNDÜR self.__yeni_maas
3. BİTİR

set_yeni_maas(self, yeni_maas):

1. BAŞLA
2. self.__yeni_maas = yeni_maas
3. BİTİR

get_zam_orani_oneri(self):

1. BAŞLA
2. DÖNDÜR self.__zam_orani_oneri
3. BİTİR

set_zam_orani_oneri(self, zam_orani_oneri):

1. BAŞLA
2. self.__zam_orani_oneri = zam_orani_oneri
3. BİTİR

get_hakedilen_zam(self):

1. BAŞLA
2. DÖNDÜR self.__hakedilen_zam
3. BİTİR

set_hakedilen_zam(self, hakedilen_zam):

1. BAŞLA
2. self.__hakedilen_zam = hakedilen_zam
3. BİTİR

c. zam_hakki Metodu

zam_hakki(self):

1. BAŞLA
2. DENE:
   2.1. EĞER float(self.__tecrube_ay) / 12 < 2 İSE:
      2.1.1. self.__zam_orani_oneri = 0
   2.2. DEĞİLSE, EĞER 2 <= float(self.__tecrube_ay) / 12 < 4 VE self.__maas < 15000 İSE:
      2.2.1. self.__zam_orani_oneri = self.__maas % self.__tecrube_ay
   2.3. DEĞİLSE, EĞER float(self.__tecrube_ay) >= 4 VE self.__maas < 25000 İSE:
      2.3.1. self.__zam_orani_oneri = (self.__maas % self.__tecrube_ay) / 2
   2.4. self.__hakedilen_zam = (float(self.__maas) / 100) * self.__zam_orani_oneri
   2.5. self.__yeni_maas = self.__maas + self.__hakedilen_zam
   2.6. EĞER self.__yeni_maas == self.__maas İSE:
      2.6.1. self.__yeni_maas = self.__maas
3. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:
   3.1. YAZDIR(f"HATA! Hata sebebi: {exception}")
4. BİTİR

d. __str__ Metodu

__str__(self):

1. BAŞLA
2. self.zam_hakki()
3. DÖNDÜR f"Ad = {self.get_ad()}" \
   f"\nSoyad = {self.get_soyad()}" \
   f"\nTecrübe = {self.__tecrube_ay} Ay" \
   f"\nYeni Maaş = {self.get_yeni_maas()}"
4. BİTİR

5. Mavi Yaka Sınıfı

a. __init__ Metodu

__init__(self, tc_no, ad, soyad, yas, cinsiyet, uyruk, sektor, tecrube_ay, maas, yipranma_payi):

1. BAŞLA
2. super().__init__(tc_no, ad, soyad, yas, cinsiyet, uyruk, sektor, tecrube_ay, maas)
3. self.__yipranma_payi = yipranma_payi
4. BİTİR

b. Private değişkenlerin Get/Set Metodları

get_yipranma_payi(self):

1. BAŞLA
2. DÖNDÜR self.__yipranma_payi
3. BİTİR

set_yipranma_payi(self, yipranma_payi):

1. BAŞLA
2. self.__yipranma_payi = yipranma_payi
3. BİTİR

c. zam_hakki Metodu

zam_hakki(self):

1. BAŞLA
2. DENE:
    2.1. zam_orani_oneri = 0
    2.2. EĞER float(self.get_tecrube_ay()) / 12 < 2 İSE:
        2.2.1. zam_orani_oneri = self.__yipranma_payi * 10
    2.3. DEĞİLSE, EĞER 2 <= float(self.get_tecrube_ay()) / 12 < 4 VE self.get_maas() < 15000 İSE:
        2.3.1. zam_orani_oneri = (float(self.get_maas() % self.get_tecrube_ay()) / 2) + (self.__yipranma_payi * 10)
    2.4. DEĞİLSE, EĞER float(self.get_tecrube_ay()) / 12 >= 4 VE self.get_maas() < 25000 İSE:
        2.4.1. zam_orani_oneri = (float(self.get_maas() % self.get_tecrube_ay()) / 3) + (self.__yipranma_payi * 10)
    2.5. self.set_zam_orani_oneri(zam_orani_oneri)
    2.6. hakedilen_zam = ((float(self.get_maas())) / 100) * zam_orani_oneri
    2.7. self.set_hakedilen_zam(hakedilen_zam)
    2.8. yeni_maas = self.get_maas() + self.get_hakedilen_zam()
    2.9. self.set_yeni_maas(yeni_maas)
    2.10. EĞER self.get_yeni_maas() == self.get_maas() İSE:
        2.10.1. self.set_yeni_maas(self.get_maas())
3. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:
    3.1. YAZDIR(f"HATA! Hata sebebi: {exception}")
4. BİTİR

d. __str__ Metodu

__str__(self):

1. BAŞLA
2. self.zam_hakki()
3. DÖNDÜR f"Ad = {self.get_ad()}" \
        f"\nSoyad = {self.get_soyad()}" \
        f"\nTecrübe = {self.get_tecrube_ay()} Ay" \
        f"\nYeni Maaş = {self.get_yeni_maas():.2f}"
4. BİTİR

6. Beyaz Yaka Sınıfı

a. __init__ Metodu

__init__(self, tc_no, ad, soyad, yas, cinsiyet, uyruk, sektor, tecrube_ay, maas, tesvik_primi):

1. BAŞLA
2. super().__init__(tc_no, ad, soyad, yas, cinsiyet, uyruk, sektor, tecrube_ay, maas)
3. self.__tesvik_primi = tesvik_primi
4. BİTİR

b. Private değişkenlerin Get/Set Metodları

get_tesvik_primi(self):

1. BAŞLA
2. DÖNDÜR self.__tesvik_primi
3. BİTİR

set_tesvik_primi(self, tesvik_primi):

1. BAŞLA
2. self.__tesvik_primi = tesvik_primi
3. BİTİR

c. zam_hakki Metodu

zam_hakki(self):

1. BAŞLA
2. DENE:

2.1. zam_onerisi = 0

2.2. EĞER float(self.get_tecrube_ay()) / 12 < 2 İSE:

2.2.1. zam_onerisi = self.__tesvik_primi

2.3. DEĞİLSE, EĞER 2 <= float(self.get_tecrube_ay()) / 12 < 4 VE self.get_maas() < 15000 İSE:

2.3.1. zam_onerisi = ((self.get_maas() % self.get_tecrube_ay()) * 5) + self.__tesvik_primi

2.4. DEĞİLSE, EĞER float(self.get_tecrube_ay()) / 12 >= 4 VE self.get_maas() < 25000 İSE:

2.4.1. zam_onerisi = ((self.get_maas() % self.get_tecrube_ay()) * 4) + self.__tesvik_primi

2.5. hakedilen_zam = zam_onerisi

2.6. self.set_hakedilen_zam(hakedilen_zam)

2.7. yeni_maas = self.get_maas() + self.get_hakedilen_zam()

2.8. self.set_yeni_maas(yeni_maas)

2.9. EĞER self.get_yeni_maas() == self.get_maas() İSE:

2.9.1. self.set_yeni_maas(self.get_maas())

3. ÇALIŞMAZSA, Exception DEĞERİNİ ATA exception:

3.1. YAZDIR(f"HATA! Hata sebebi: {exception}")

4. BİTİR

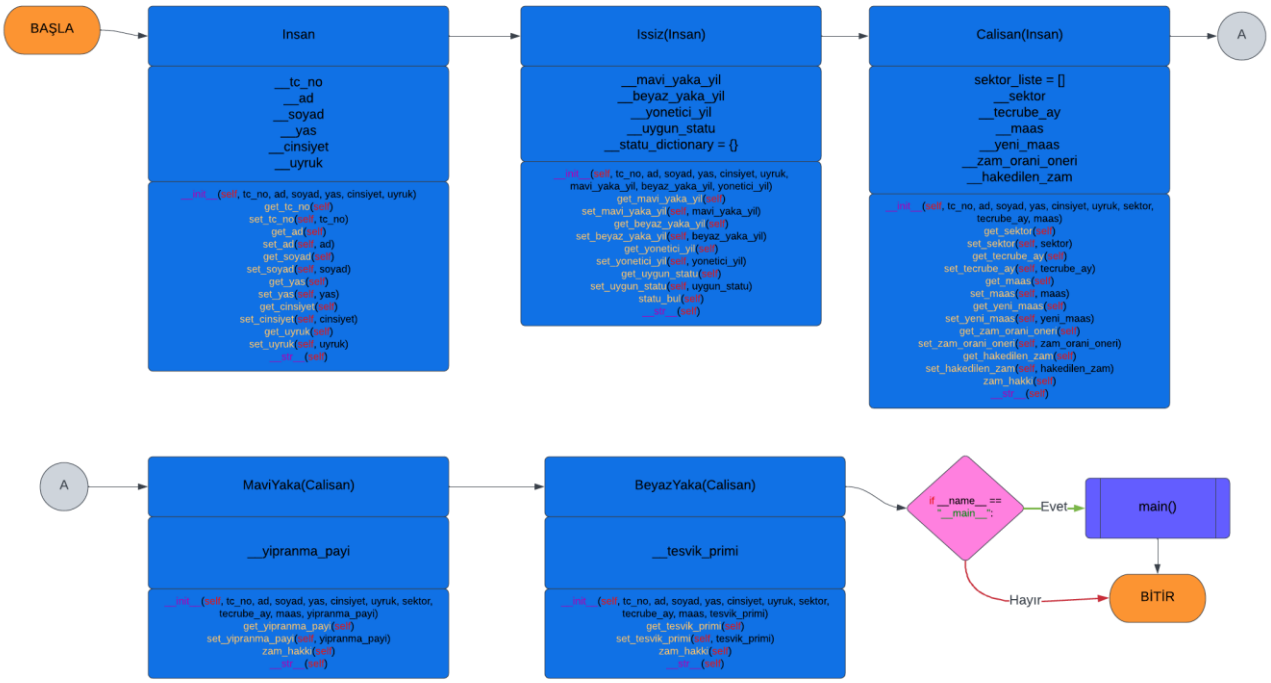
d. __str__ Metodu

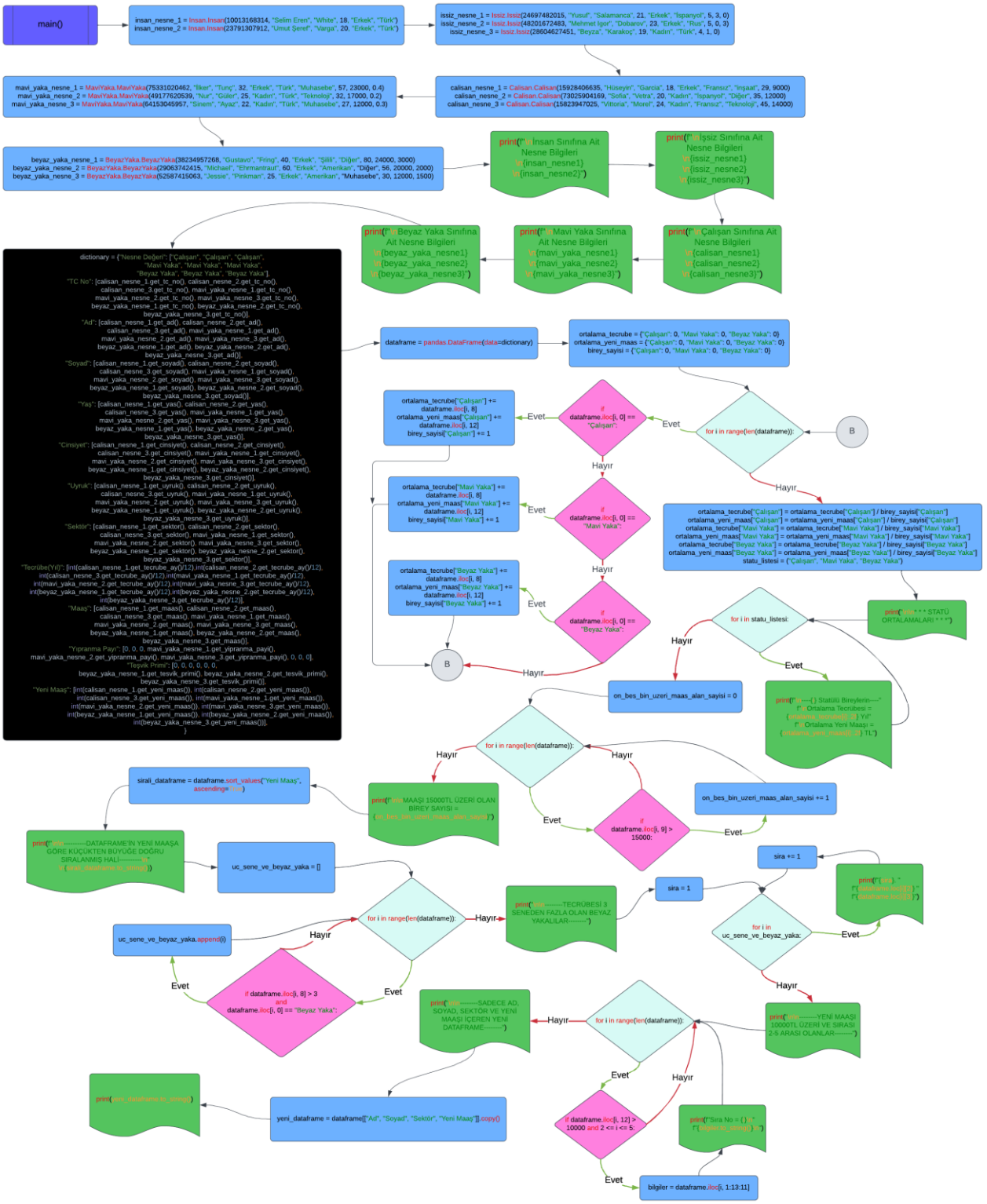
__str__(self):

1. BAŞLA
2. self.zam_hakki()
3. DÖNDÜR f"Ad = {self.get_ad()}" \
   f"\nSoyad = {self.get_soyad()}" \
   f"\nTecrübe = {self.get_tecrube_ay()} Ay" \
   f"\nYeni Maaş = {self.get_yeni_maas():.2f}"
4. BİTİR

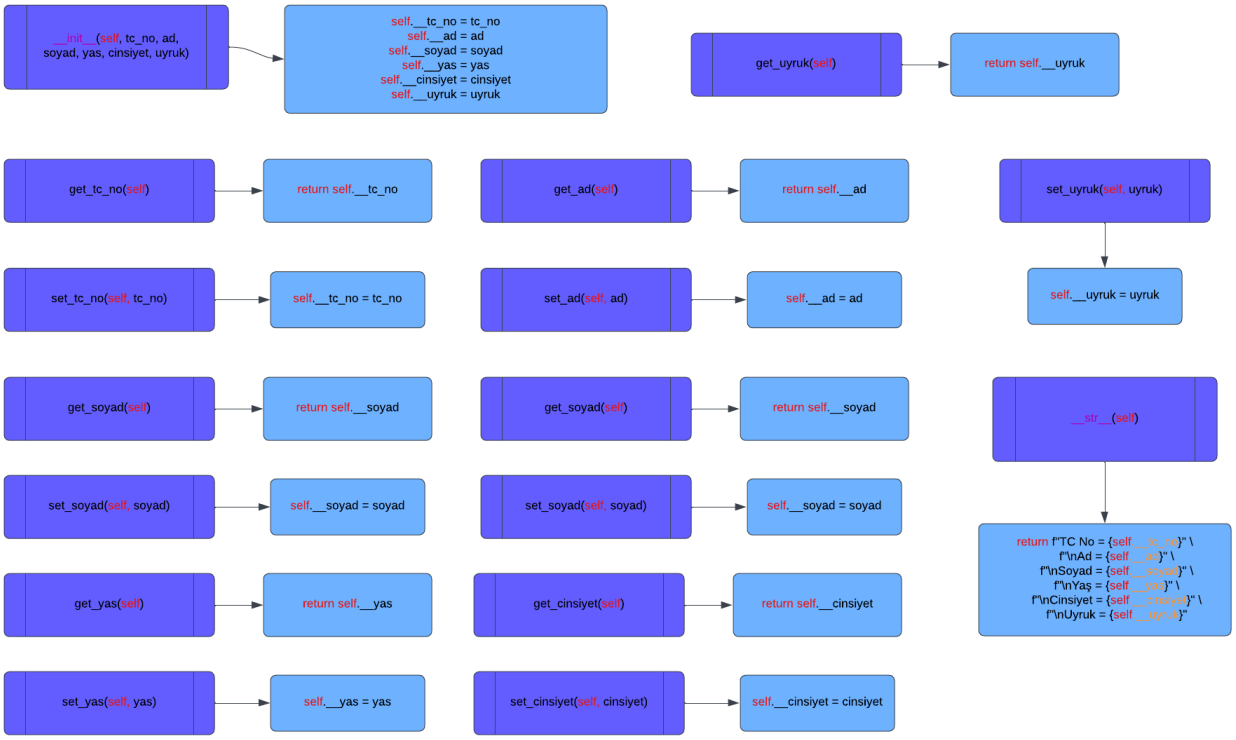
E. Akış Diyagramları

1. Main.py



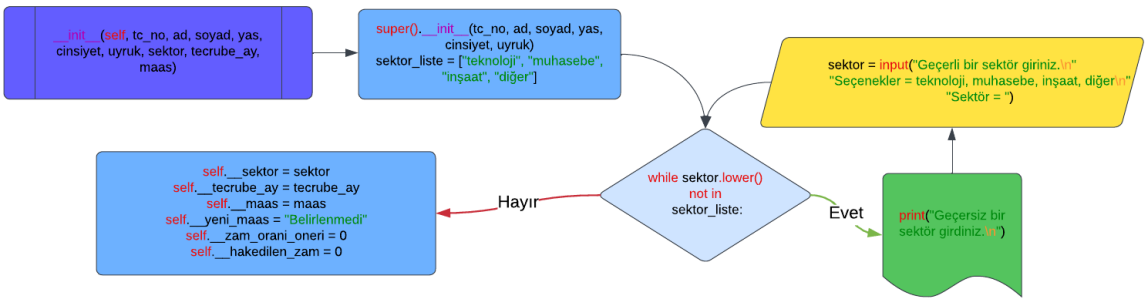

main()
insan_nesne_1 = Insan.Insan(10013168314, "Selim Eren", "White", 18, "Erkek", "Türk")
insan_nesne_2 = Insan.Insan(23791307912, "Umut Şeref", "Varga", 20, "Erkek", "Türk")
issiz_nesne_1 = Issiz.Issiz(24697482015, "Yusuf", "Salamanca", 21, "Erkek", "İspanyol", 5, 3, 0)
issiz_nesne_2 = Issiz.Issiz(48201672483, "Mehmet Igor", "Dobarov", 23, "Erkek", "Rus", 5, 0, 3)
issiz_nesne_3 = Issiz.Issiz(28604627451, "Beyza", "Karakoç", 19, "Kadın", "Türk", 4, 1, 0)
calisan_nesne_1 = Calisan.Calisan(15928406635, "Hüseyin", "Garcia", 18, "Erkek", "Fransız", "inşaat", 29, 9000)
calisan_nesne_2 = Calisan.Calisan(73025904169, "Sofia", "Vetra", 20, "Kadın", "İspanyol", "Diğer", 35, 12000)
calisan_nesne_3 = Calisan.Calisan(15823947025, "Vittoria", "Morel", 24, "Kadın", "Fransız", "Teknoloji", 45, 14000)
mavi_yaka_nesne_1 = MaviYaka.MaviYaka(75331020462, "İlker", "Tunç", 32, "Erkek", "Türk", "Muhasebe", 57, 23000, 0.4)
mavi_yaka_nesne_2 = MaviYaka.MaviYaka(49177620539, "Nur", "Güler", 25, "Kadın", "Türk", "Teknoloji", 32, 17000, 0.2)
mavi_yaka_nesne_3 = MaviYaka.MaviYaka(64153045957, "Sinem", "Ayaz", 22, "Kadın", "Türk", "Muhasebe", 27, 12000, 0.3)
beyaz_yaka_nesne_1 = BeyazYaka.BeyazYaka(38234957268, "Gustavo", "Fring", 40, "Erkek", "Sili", "Diğer", 80, 24000, 3000)
beyaz_yaka_nesne_2 = BeyazYaka.BeyazYaka(29063742415, "Michael", "Ehrmantraut", 60, "Erkek", "Amerikan", "Diğer", 56, 20000, 2000)
beyaz_yaka_nesne_3 = BeyazYaka.BeyazYaka(52587415063, "Jessie", "Pinkman", 25, "Erkek", "Amerikan", "Muhasebe", 30, 12000, 1500)
print(f"\n İnsan Sınıfına Ait Nesne Bilgileri \n {insan_nesne1} \n {insan_nesne2}")
print(f"\n İşsiz Sınıfına Ait Nesne Bilgileri \n {issiz_nesne1} \n {issiz_nesne2} \n {issiz_nesne3}")
print(f"\n Çalışan Sınıfına Ait Nesne Bilgileri \n {calisan_nesne1} \n {calisan_nesne2} \n {calisan_nesne3}")
print(f"\n Mavi Yaka Sınıfına Ait Nesne Bilgileri \n {mavi_yaka_nesne1} \n {mavi_yaka_nesne2} \n {mavi_yaka_nesne3}")
print(f"\n Beyaz Yaka Sınıfına Ait Nesne Bilgileri \n {beyaz_yaka_nesne1} \n {beyaz_yaka_nesne2} \n {beyaz_yaka_nesne3}")
dictionary = {"Nesne Değeri": ["Çalışan", "Çalışan", "Çalışan", "Mavi Yaka", "Mavi Yaka", "Mavi Yaka", "Beyaz Yaka", "Beyaz Yaka", "Beyaz Yaka"], "TC No": [calisan_nesne_1.get_tc_no(), calisan_nesne_2.get_tc_no(), calisan_nesne_3.get_tc_no(), mavi_yaka_nesne_1.get_tc_no(), mavi_yaka_nesne_2.get_tc_no(), mavi_yaka_nesne_3.get_tc_no(), beyaz_yaka_nesne_1.get_tc_no(), beyaz_yaka_nesne_2.get_tc_no(), beyaz_yaka_nesne_3.get_tc_no()], "Ad": [calisan_nesne_1.get_ad(), calisan_nesne_2.get_ad(), calisan_nesne_3.get_ad(), mavi_yaka_nesne_1.get_ad(), mavi_yaka_nesne_2.get_ad(), mavi_yaka_nesne_3.get_ad(), beyaz_yaka_nesne_1.get_ad(), beyaz_yaka_nesne_2.get_ad(), beyaz_yaka_nesne_3.get_ad()], "Soyad": [calisan_nesne_1.get_soyad(), calisan_nesne_2.get_soyad(), calisan_nesne_3.get_soyad(), mavi_yaka_nesne_1.get_soyad(), mavi_yaka_nesne_2.get_soyad(), mavi_yaka_nesne_3.get_soyad(), beyaz_yaka_nesne_1.get_soyad(), beyaz_yaka_nesne_2.get_soyad(), beyaz_yaka_nesne_3.get_soyad()], "Yaş": [calisan_nesne_1.get_yas(), calisan_nesne_2.get_yas(), calisan_nesne_3.get_yas(), mavi_yaka_nesne_1.get_yas(), mavi_yaka_nesne_2.get_yas(), mavi_yaka_nesne_3.get_yas(), beyaz_yaka_nesne_1.get_yas(), beyaz_yaka_nesne_2.get_yas(), beyaz_yaka_nesne_3.get_yas()], "Cinsiyet": [calisan_nesne_1.get_cinsiyet(), calisan_nesne_2.get_cinsiyet(), calisan_nesne_3.get_cinsiyet(), mavi_yaka_nesne_1.get_cinsiyet(), mavi_yaka_nesne_2.get_cinsiyet(), mavi_yaka_nesne_3.get_cinsiyet(), beyaz_yaka_nesne_1.get_cinsiyet(), beyaz_yaka_nesne_2.get_cinsiyet(), beyaz_yaka_nesne_3.get_cinsiyet()], "Uyruk": [calisan_nesne_1.get_uyruk(), calisan_nesne_2.get_uyruk(), calisan_nesne_3.get_uyruk(), mavi_yaka_nesne_1.get_uyruk(), mavi_yaka_nesne_2.get_uyruk(), mavi_yaka_nesne_3.get_uyruk(), beyaz_yaka_nesne_1.get_uyruk(), beyaz_yaka_nesne_2.get_uyruk(), beyaz_yaka_nesne_3.get_uyruk()], "Sektör": [calisan_nesne_1.get_sektor(), calisan_nesne_2.get_sektor(), calisan_nesne_3.get_sektor(), mavi_yaka_nesne_1.get_sektor(), mavi_yaka_nesne_2.get_sektor(), mavi_yaka_nesne_3.get_sektor(), beyaz_yaka_nesne_1.get_sektor(), beyaz_yaka_nesne_2.get_sektor(), beyaz_yaka_nesne_3.get_sektor()], "Tecrübe(Yıl)": [int(calisan_nesne_1.get_tecrube_ay()/12),int(calisan_nesne_2.get_tecrube_ay()/12), int(calisan_nesne_3.get_tecrube_ay()/12),int(mavi_yaka_nesne_1.get_tecrube_ay()/12), int(mavi_yaka_nesne_2.get_tecrube_ay()/12),int(mavi_yaka_nesne_3.get_tecrube_ay()/12), int(beyaz_yaka_nesne_1.get_tecrube_ay()/12),int(beyaz_yaka_nesne_2.get_tecrube_ay()/12), int(beyaz_yaka_nesne_3.get_tecrube_ay()/12)], "Maaş": [calisan_nesne_1.get_maas(), calisan_nesne_2.get_maas(), calisan_nesne_3.get_maas(), mavi_yaka_nesne_1.get_maas(), mavi_yaka_nesne_2.get_maas(), mavi_yaka_nesne_3.get_maas(), beyaz_yaka_nesne_1.get_maas(), beyaz_yaka_nesne_2.get_maas(), beyaz_yaka_nesne_3.get_maas()], "Yıpranma Payı": [0, 0, 0, mavi_yaka_nesne_1.get_yipranma_payi(), mavi_yaka_nesne_2.get_yipranma_payi(), mavi_yaka_nesne_3.get_yipranma_payi(), 0, 0, 0], "Teşvik Primi": [0, 0, 0, 0, 0, 0, beyaz_yaka_nesne_1.get_tesvik_primi(), beyaz_yaka_nesne_2.get_tesvik_primi(), beyaz_yaka_nesne_3.get_tesvik_primi()], "Yeni Maaş": [int(calisan_nesne_1.get_yeni_maas()), int(calisan_nesne_2.get_yeni_maas()), int(calisan_nesne_3.get_yeni_maas()), int(mavi_yaka_nesne_1.get_yeni_maas()), int(mavi_yaka_nesne_2.get_yeni_maas()), int(mavi_yaka_nesne_3.get_yeni_maas()), int(beyaz_yaka_nesne_1.get_yeni_maas()), int(beyaz_yaka_nesne_2.get_yeni_maas()), int(beyaz_yaka_nesne_3.get_yeni_maas())]}
dataframe = pandas.DataFrame(data=dictionary)
ortalama_tecrube = {"Çalışan": 0, "Mavi Yaka": 0, "Beyaz Yaka": 0}
ortalama_yeni_maas = {"Çalışan": 0, "Mavi Yaka": 0, "Beyaz Yaka": 0}
birey_sayisi = {"Çalışan": 0, "Mavi Yaka": 0, "Beyaz Yaka": 0}
for i in range(len(dataframe)):
B
if dataframe.iloc[i, 0] == "Çalışan":
Evet
Hayır
ortalama_tecrube["Çalışan"] += dataframe.iloc[i, 8]
ortalama_yeni_maas["Çalışan"] += dataframe.iloc[i, 12]
birey_sayisi["Çalışan"] += 1
if dataframe.iloc[i, 0] == "Mavi Yaka":
ortalama_tecrube["Mavi Yaka"] += dataframe.iloc[i, 8]
ortalama_yeni_maas["Mavi Yaka"] += dataframe.iloc[i, 12]
birey_sayisi["Mavi Yaka"] += 1
if dataframe.iloc[i, 0] == "Beyaz Yaka":
ortalama_tecrube["Beyaz Yaka"] += dataframe.iloc[i, 8]
ortalama_yeni_maas["Beyaz Yaka"] += dataframe.iloc[i, 12]
birey_sayisi["Beyaz Yaka"] += 1
ortalama_tecrube["Çalışan"] = ortalama_tecrube["Çalışan"] / birey_sayisi["Çalışan"]
ortalama_yeni_maas["Çalışan"] = ortalama_yeni_maas["Çalışan"] / birey_sayisi["Çalışan"]
ortalama_tecrube["Mavi Yaka"] = ortalama_tecrube["Mavi Yaka"] / birey_sayisi["Mavi Yaka"]
ortalama_yeni_maas["Mavi Yaka"] = ortalama_yeni_maas["Mavi Yaka"] / birey_sayisi["Mavi Yaka"]
ortalama_tecrube["Beyaz Yaka"] = ortalama_tecrube["Beyaz Yaka"] / birey_sayisi["Beyaz Yaka"]
ortalama_yeni_maas["Beyaz Yaka"] = ortalama_yeni_maas["Beyaz Yaka"] / birey_sayisi["Beyaz Yaka"]
statu_listesi = ("Çalışan", "Mavi Yaka", "Beyaz Yaka")
print("\n\n * * * STATÜ ORTALAMALARI * * *")
for i in statu_listesi:
Hayır
Evet
print(f" \n\n ---{ } Statülü Bireylerin---"
f"\n Ortalama Tecrübesi = {ortalama_tecrube[i]:.2f} Yıl"
f"\n Ortalama Yeni Maaşı = {ortalama_yeni_maas[i]:.2f} TL")
on_bes_bin_uzeri_maas_alan_sayisi = 0
for i in range(len(dataframe)):
if dataframe.iloc[i, 9] > 15000:
on_bes_bin_uzeri_maas_alan_sayisi += 1
print(f"\n\n MAAŞI 15000TL ÜZERİ OLAN BİREY SAYISI = {on_bes_bin_uzeri_maas_alan_sayisi}")
sirali_dataframe = dataframe.sort_values("Yeni Maaş", ascending=True)
print(f"\n\n---------DATAFRAME'İN YENİ MAAŞA GÖRE KÜÇÜKTEN BÜYÜĞE DOĞRU SIRALANMIŞ HALİ---------: \n\n {sirali_dataframe.to_string()}")
uc_sene_ve_beyaz_yaka = []
for i in range(len(dataframe)):
if dataframe.iloc[i, 8] > 3 and dataframe.iloc[i, 0] == "Beyaz Yaka":
uc_sene_ve_beyaz_yaka.append(i)
print("\n\n---------TECRÜBESİ 3 SENEDEN FAZLA OLAN BEYAZ YAKALILAR---------")
sira = 1
for i in uc_sene_ve_beyaz_yaka:
print(f"{sira} - " f"{dataframe.iloc[i][1]} " f"{dataframe.iloc[i][2]}")
sira += 1
print("\n\n---------SADECE AD, SOYAD, SEKTÖR VE YENİ MAAŞI İÇEREN YENİ DATAFRAME---------")
yeni_dataframe = dataframe[["Ad", "Soyad", "Sektör", "Yeni Maaş"]].copy()
print(yeni_dataframe.to_string())
print("\n\n---------YENİ MAAŞI 10000TL ÜZERİ VE SIRASI 2-5 ARASI OLANLAR---------")
for i in range(len(dataframe)):
if dataframe.iloc[i, 12] > 10000 and 2 <= i <= 5:
print(f"Sıra No = {i} :" f"{register.to_string(i)}")
bilgiler = dataframe.iloc[i, 1:13:11]

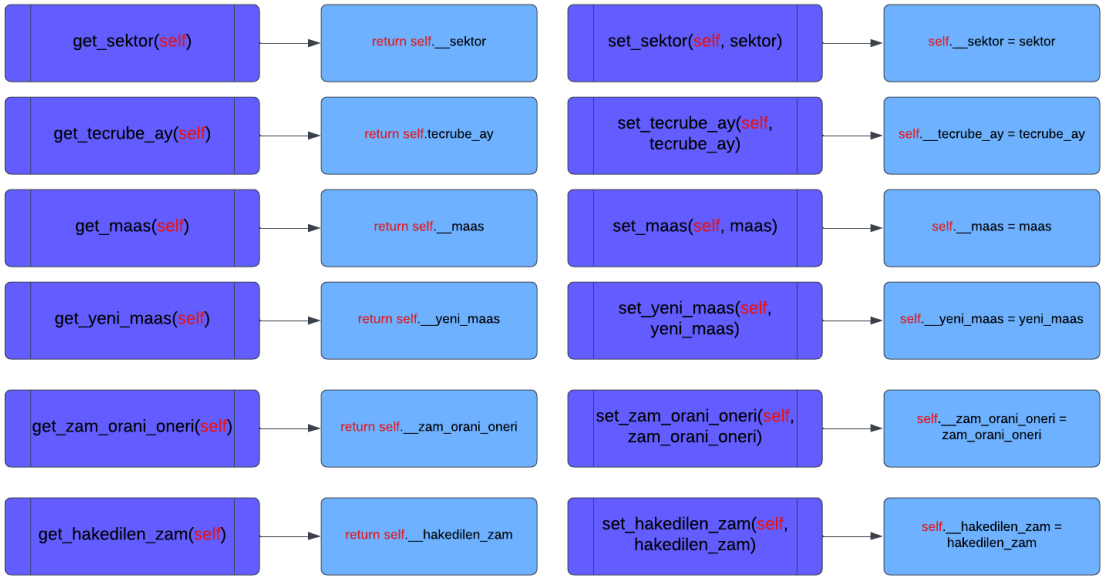
2. Insan.py

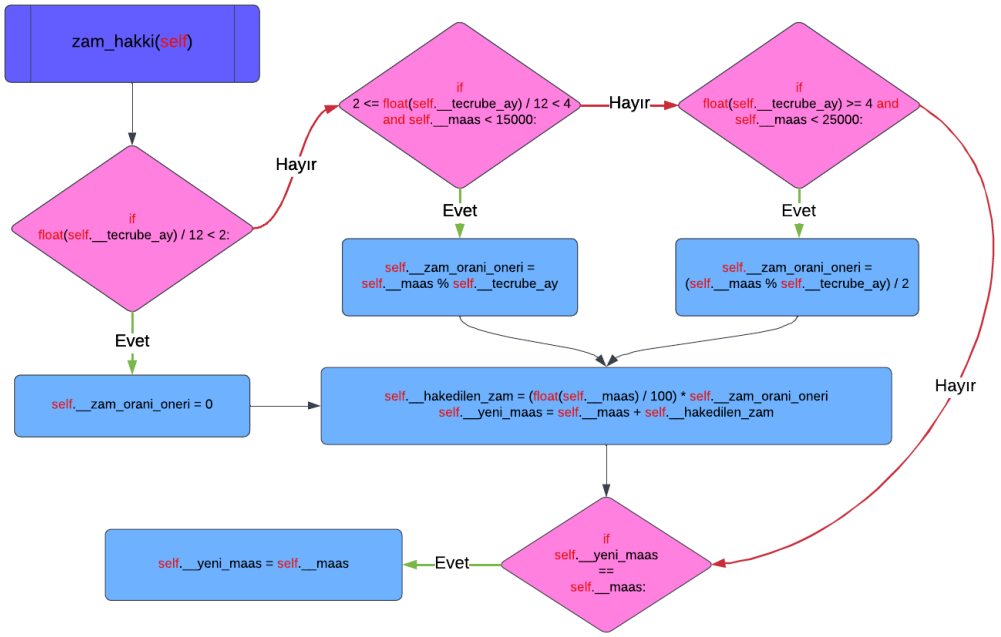


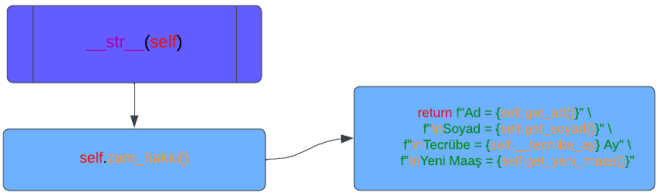
3. Issiz.py

4. Calisan.py

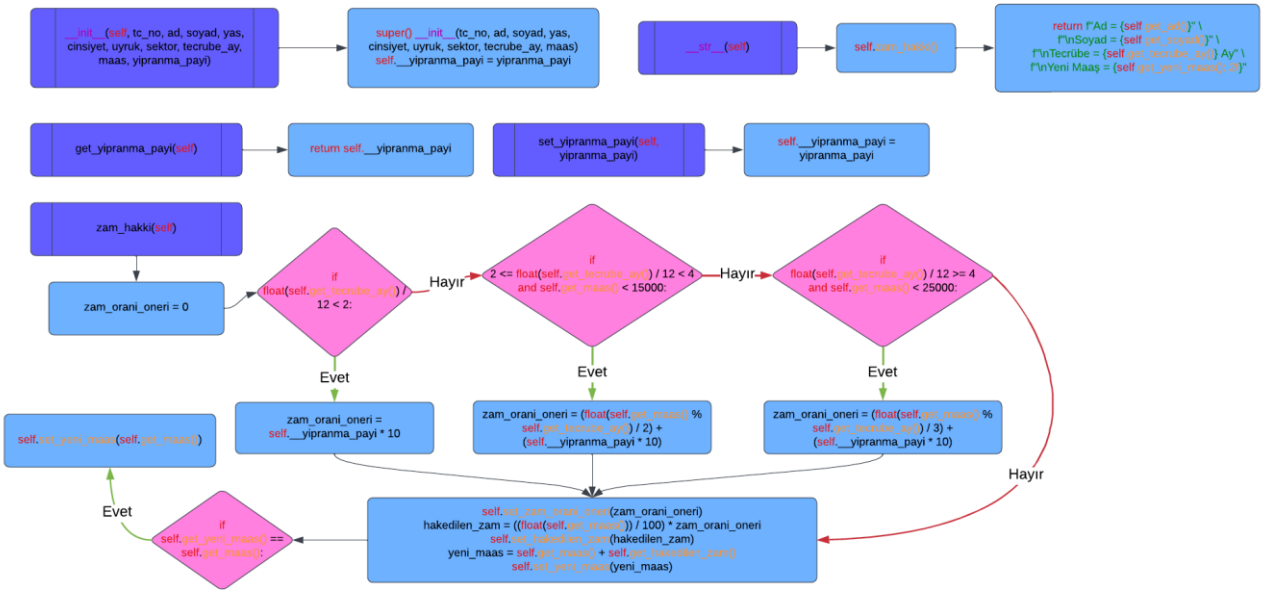

__init__(self, tc_no, ad, soyad, yas, cinsiyet, uyruk, sektor, tecrube_ay, maas)
super().__init__(tc_no, ad, soyad, yas, cinsiyet, uyruk)
sektor_liste = ["teknoloji", "muhasebe", "inşaat", "diğer"]
sektor = input("Geçerli bir sektör giriniz.\n" "Seçenekler = teknoloji, muhasebe, inşaat, diğer\n" "Sektör = ")
while sektor.lower() not in sektor_liste:
Evet
print("Geçersiz bir sektör girdiniz.\n")
Hayır
self.__sektor = sektor
self.__tecrube_ay = tecrube_ay
self.__maas = maas
self.__yeni_maas = "Belirlenmedi"
self.__zam_orani_oneri = 0
self.__hakedilen_zam = 0


get_sektor(self)
return self.__sektor
set_sektor(self, sektor)
self.__sektor = sektor
get_tecrube_ay(self)
return self.tecrube_ay
set_tecrube_ay(self, tecrube_ay)
self.__tecrube_ay = tecrube_ay
get_maas(self)
return self.__maas
set_maas(self, maas)
self.__maas = maas
get_yeni_maas(self)
return self.__yeni_maas
set_yeni_maas(self, yeni_maas)
self.__yeni_maas = yeni_maas
get_zam_orani_oneri(self)
return self.__zam_orani_oneri
set_zam_orani_oneri(self, zam_orani_oneri)
self.__zam_orani_oneri = zam_orani_oneri
get_hakedilen_zam(self)
return self.__hakedilen_zam
set_hakedilen_zam(self, hakedilen_zam)
self.__hakedilen_zam = hakedilen_zam

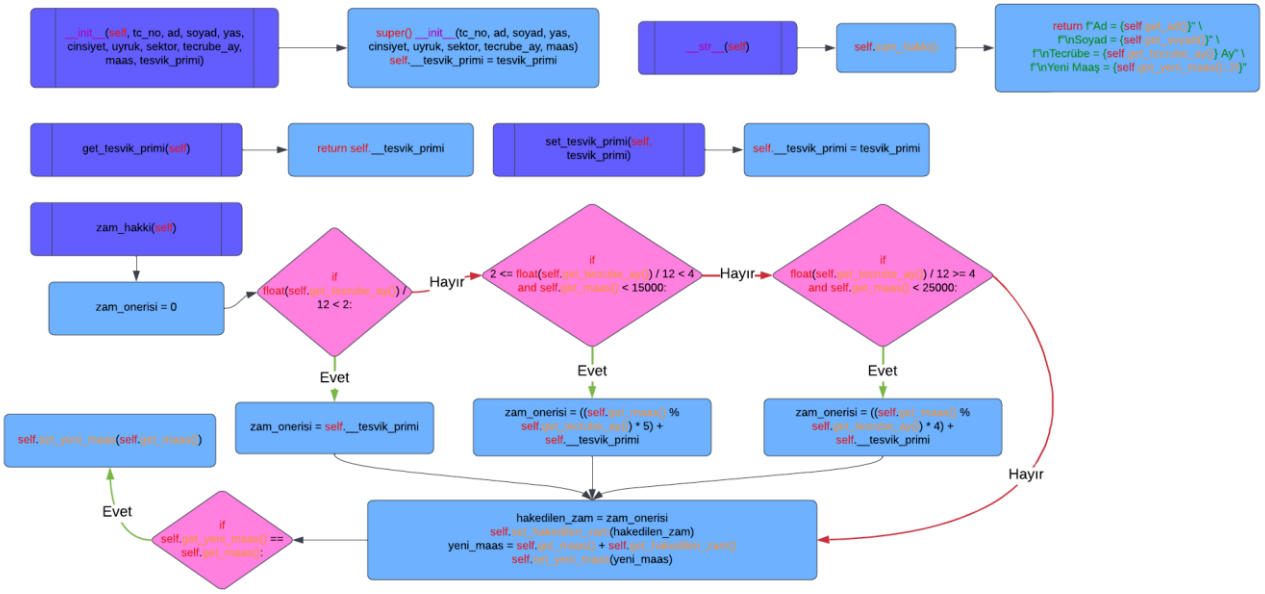

zam_hakki(self)
if float(self.__tecrube_ay) / 12 < 2:
Hayır
if 2 <= float(self.__tecrube_ay) / 12 < 4 and self.__maas < 15000:
Hayır
if float(self.__tecrube_ay) >= 4 and self.__maas < 25000:
Evet
self.__zam_orani_oneri = self.__maas % self.__tecrube_ay
Evet
self.__zam_orani_oneri = (self.__maas % self.__tecrube_ay) / 2
Evet
self.__zam_orani_oneri = 0
self.__hakedilen_zam = (float(self.__maas) / 100) * self.__zam_orani_oneri
self.__yeni_maas = self.__maas + self.__hakedilen_zam
Hayır
if self.__yeni_maas == self.__maas:
Evet
self.__yeni_maas = self.__maas


__str__(self)
self.zam_hakki()
return f"Ad = {self.get_ad()}" \
f"\nSoyad = {self.get_soyad()}" \
f"\nTecrübe = {self.__tecrube_ay} Ay" \
f"\nYeni Maaş = {self.get_yeni_maas()}"

5. MaviYaka.py

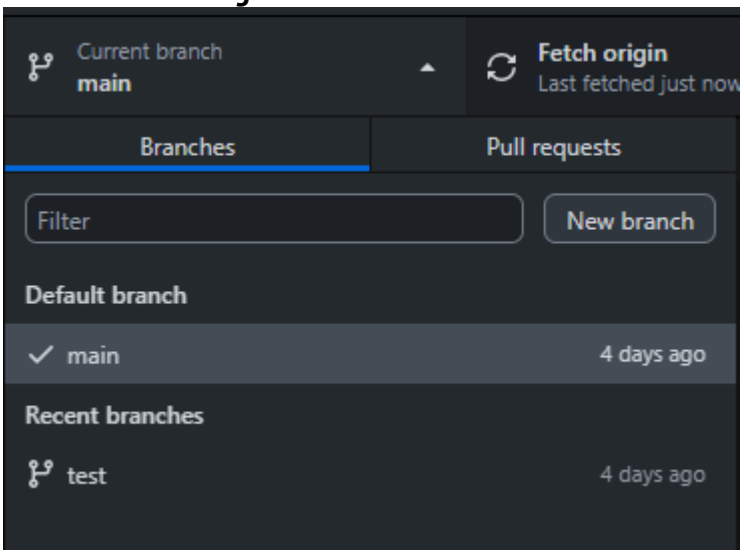


6. BeyazYaka.py

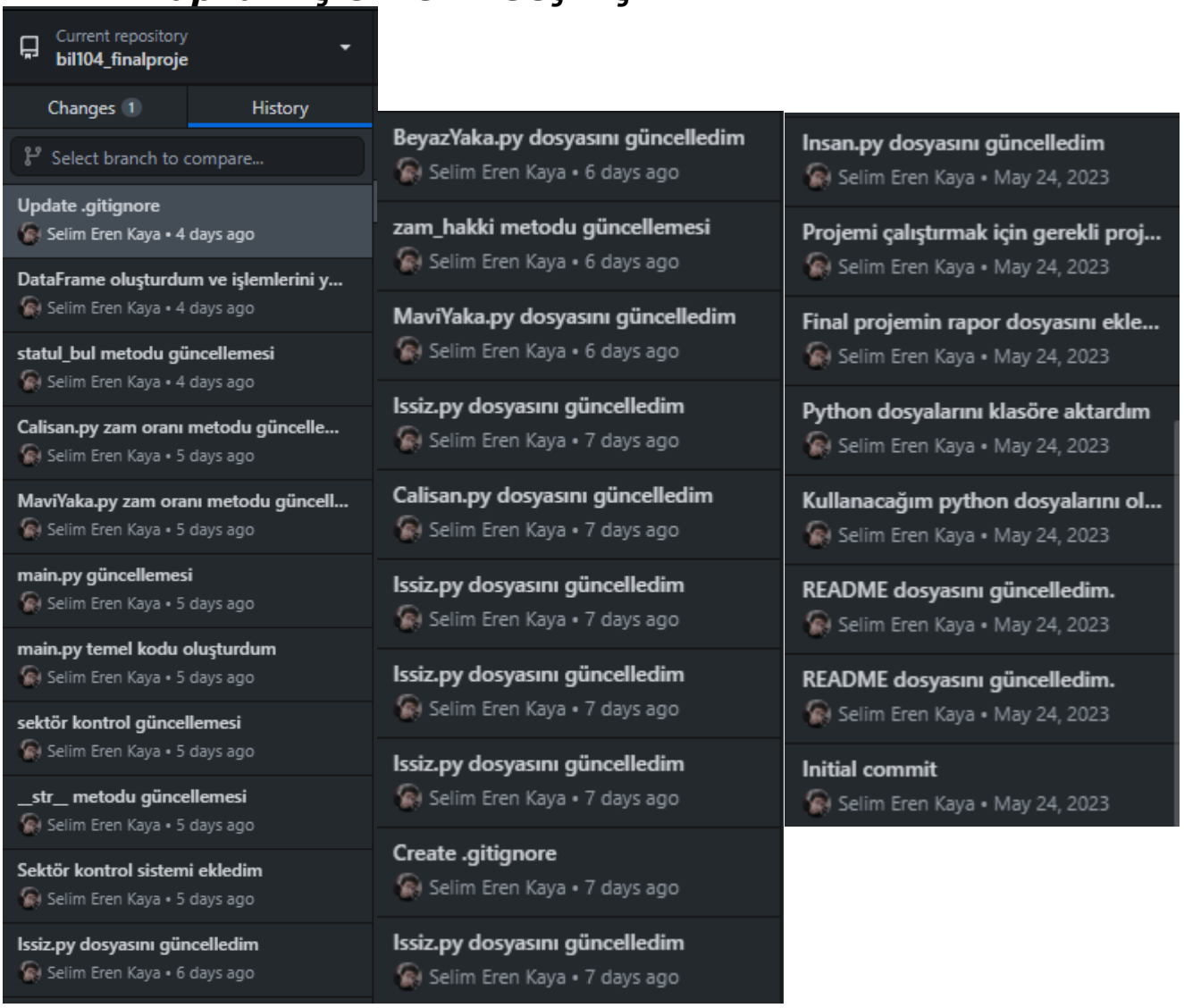


F. Github Kısmı

1. Oluşturulan Branchler



2. Yapılan İşlemlerin Geçmişi



3. Github Linkleri

Github Profil Linki: https://github.com/selimerenkaya

Github Final Proje Linki: https://github.com/selimerenkaya/bil104_finalproje