Nesneye Yönelimli Programlama – BLM 205

Hafta 5: Objeler ve Sınıflar

Fenerbahçe Üniversitesi

Öğretim Elemanları

Öğretim Üyesi: Dr. Vecdi Emre Levent

Ofis: 311

Email: emre.levent@fbu.edu.tr

Asistan: Arş. Gör. Uğur Özbalkan

Ofis: 307

Email: ugur.ozbalkan@fbu.edu.tr

Asistan: Arş. Gör. Ecenur Alioğulları

Ofis: 307

Email: ecenur.aliogullari@fbu.edu.tr

Ders Planı

- Objeler ve Sınıflar
  - Sınıf işlemleri
  - Constructor
  - Destructor

Objeler ve Sınıflar

Sınıflar

Değişkenler ve fonksiyonların bir araya gelerek oluşturulan, kodun yeniden kullanılılabilirliğini arttıran bir geliştirme yaklaşımıdır.

Gruplanarak oluşturulmuş olan sınıflardan, objeler yaratılır.

Dolayısıyla aynı fonksiyon ve değişkenlere sahip birden çok obje oluşturulabilir.

Objeler ve Sınıflar

Sınıflar

Örneğin, sınıflar konsepti ile programlama yapmadan,

5 kişinin adı, soyadı, yaşı bilgileri alınarak her birini ekrana bastıran bir uygulama geliştirilmesi istendiğinde;

Objeler ve Sınıflar

Sınıflar

Her bir kişi'nin tüm özellikleri için değişkenler oluşturarak bilgiler alınıp ekrana yazılabilir.

Yani

kisi1_isim, kisi2_isim, kisi1_yas … gibi

Objeler ve Sınıflar

Sınıflar

Kişi sayısı 100'e çıkartılmak istenirse, veya yeni bir özellik (Kişi'nin isminde hata var mı yok mu kontrol eden bir fonksiyon, içerisinde sayı olup olmadığını kontrol edebilir) getirilmek istenirse, önceden yazılmış kodun yeniden kullanılabilirliği az olacaktır.

Tekrar geliştirmeye oldukça yoğun efor göstermek gerekecektir.

Objeler ve Sınıflar

Sınıflar

Bunun için, gerekli olan objenin prototipini içerecek fonksiyon ve değişkenleri barındıran bir sınıf tanımlanır.

class keyword'ü ile tanımlanır.

Objeler ve Sınıflar

Sınıflar

Sınıf tanımı:

Örnek Kod Parçacığı

```
class sinifAdi:
    # Sinif Degiskenleri ve Fonksiyonlar
```

Objeler ve Sınıflar

Sınıflar

Sınıf tanımı:

Örnek Kod Parçacığı

```
class insanlar:
    isim = "";
    soyisim = "Test"
    yas = -1

    def isimDuzelt(self, isimArg):
        self.isim = isimArg

insanObjesi1 = insanlar()
insanObjesi1.isimDuzelt("Deneme")
print(insanObjesi1.isim)
```

Çıktı

```
Deneme
```

İlk argüman her zaman self'tir.
C++'taki this pointer'i gibidir.
Sınıf'ın içerisindeki değişkene erişmek için kullanılır

Objeler ve Sınıflar

Sınıflar

Init metodu, C++'taki constructor'a benzer. Sınıf oluşturulurken ilk değişken atamaları yapılma ihtiyacı var ise, bu fonksiyon aracılığı ile yapılabilir.

Örnek Kod Parçacığı

```
class kisi:

    isim = ""

    def __init__(self, isimArg):
        self.isim = isimArg

    def testMesaji(self):
        print('Merhaba benim adim ', self.isim)

obje1 = kisi('Ali')
obje1.testMesaji()
```

Çıktı

```
Merhaba benim adim  Ali
```

Objeler ve Sınıflar

Sınıflar

Örnek Kod Parçacığı

```
class kopek:

    def __init__(self, cinsi, rengi):

        self.cins = cinsi
        self.renk = rengi

kopek1 = kopek("Coban", "Siyah")
kopek2 = kopek("Bulldog", "Kahverengi")

print('Kopek 1 Bilgileri:')
print('Cinsi: ', kopek1.cins)
print('Rengi: ', kopek1.renk)

print('\nKopek 2 Bilgileri:')
print('Cinsi: ', kopek2.cins)
print('Rengi: ', kopek2.renk)
```

Çıktı

```
Kopek 1 Bilgileri:
Cinsi:  Coban
Rengi:  Siyah

Kopek 2 Bilgileri:
Cinsi:  Bulldog
Rengi:  Kahverengi
```

Objeler ve Sınıflar

Sınıflar

Örnek Kod Parçacığı

```
class kopek:

    def __init__(self, cinsi, rengi):

        self.cins = cinsi
        self.renk = rengi

    def renkAyarla(self, rengi):
        self.renk = rengi

    def renkAl(self):
        return self.renk

kopek1 = kopek("Coban", "Siyah")

kopek1.renkAyarla("Beyaz")

print('Kopek 1 Bilgileri:')
print('Cinsi: ', kopek1.cins)
print('Rengi: ', kopek1.renk)
```

Çıktı

```
Kopek 1 Bilgileri:
Cinsi:  Coban
Rengi:  Beyaz
```

Sınıf içindeki renk değişkenini, ayarlama ve alma fonksiyonları

Objeler ve Sınıflar

Sınıflar

Örnek Kod Parçacığı

```
class toplama:
    birinci = 0
    ikinci = 0
    sonuc = 0

    def __init__(self, f, s):
        self.birinci = f
        self.ikinci = s

    def goster(self):
        print("Birinci Sayi = " + str(self.birinci))
        print("Ikinci Sayi = " + str(self.ikinci))
        print("Toplam = " + str(self.sonuc))

    def hesapla(self):
        self.sonuc = self.birinci + self.ikinci

obj = toplama(1000, 2000)
obj.hesapla()
obj.goster()
```

Çıktı

```
Birinci Sayi = 1000
Ikinci Sayi = 2000
Toplam = 3000
```

Objeler ve Sınıflar

Sınıflar

Destructor’lar:

Objeler artık kullanılmayacağı zaman silinirler, silinme işlemi gerçekleşmeden hemen önce gerçekleştirilmesi istenen işlemler var ise, o işlemler destructor bloğu içerisine kodlanır.

Obje silinmeden önce destructor bloğu çalıştırılarak, gerekli işlemlerin yapılması sağlanır.

Sınıf içerisinde __del__ fonksiyonu ile tanımlanır

Objeler ve Sınıflar

Sınıflar

Destructor'lar:

Örnek Kod Parçacığı

```
class calisanlar:

    def __init__(self):
        print('Calisan olusturuldu.')

    def __del__(self):
        print('Destructor cagrildi. Calisan objesi
silindi')

obj = calisanlar()
del obj
```

Çıktı

```
Calisan olusturuldu.
Destructor cagrildi. Calisan objesi silindi
```

Objeler ve Sınıflar

Sınıflar

Destructor'lar:

Örnek Kod Parçacığı

```
class calisanlar:

    def __init__(self):
        print('Calisan Olusturuldu')

    def __del__(self):
        print("Destructor cagrildi")

def testFonksiyonu():
    print('Obje olusturulacak...')
    obj = calisanlar()
    print('Fonksiyon bitimi...')

print('Test fonksiyonu cagrilacak...')
testFonksiyonu()
print('Program bitimi...')
```

Çıktı

```
Test fonksiyonu cagrilacak...
Obje olusturulacak...
Calisan Olusturuldu
Fonksiyon bitimi...
Destructor cagrildi
Program bitimi...
```

Fonksiyon bitiminde obje artık kullanılmıyor. Dolayısıyla otomatik olarak silindi ve destructor cağrıldı

Objeler ve Sınıflar

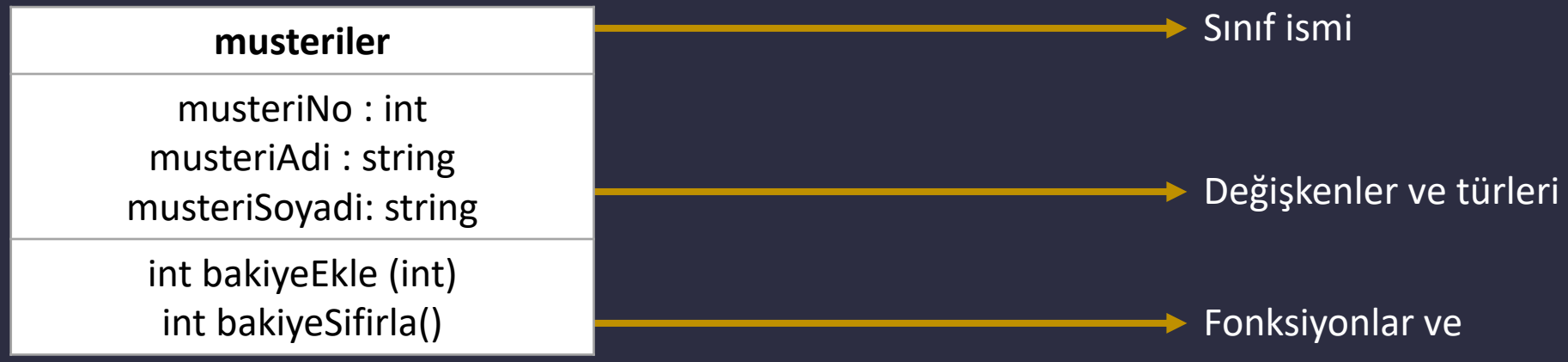
Sınıflar

Sınıflarda tanımlanmış değişken ve fonksiyonlar UML (Unified Modeling Language) isminde bir dilde ifade edilmesi, sınıf hakkında bilgi edinmek için oldukça faydalıdır.

Objeler ve Sınıflar

Sınıflar

UML örneği:



Objeler ve Sınıflar

Sınıflar Private Değişken ve Fonksiyonlar

Örnek Kod Parçacığı

```
class calisanlar:

    def __isimAyarla(isimArg):
        self.isim=isimArg

    def __init__(self, isimArg, maasArg):
        self.isim=isimArg
        self.__maas=maasArg

calisan1=calisanlar("Ahmet",5000)

print(calisan1.isim)
print(calisan1.__maas)
```

Çıktı

```
Ahmet
```

__isimAyarla fonksiyonu private tanımlanmış. Yani sadece sınıf içerisindeki diğer fonksiyonlar tarafından çağrılabilir.

__maas değişkenine erişilmeye çalışıldığında, **"'calisanlar' object has no attribute '__maas'"** hatası alınacaktır.

Objeler ve Sınıflar

Örnek Kod Parçacığı

```
class Laptop:
    isim = ''
    islemci = ''
    frekans = ''

    def baslat():
        print('Laptop ', isim, ' baslatiliyor...')

    def yenidenBaslat(self):
        print('Laptop ', isim, ' yeniden baslatiliyor...')

    def detaylarGetir(self):
        print('Laptop adi:', self.isim)
        print(self.islemci,' islemcisine sahiptir.')
        islemciGucu = self.frekans * 10
        print('Islemci gucu ', islemciGucu, ' sahiptir.')
```

Örnek Kod Parçacığı

```
laptop1 = Laptop()
laptop1.isim = 'HP'
laptop1.islemci = 'Intel Core i7'
laptop1.frekans = 2.5;
laptop1.detaylarGetir()

laptop2 = Laptop()
laptop2.isim = 'DELL'
laptop2.islemci = 'Intel Core i5'
laptop2.frekans = 1.5;
laptop2.detaylarGetir()
```

Çıktı

```
Laptop adi: HP
Intel Core i7  islemcisine sahiptir.
Islemci gucu  25.0  sahiptir.
Laptop adi: DELL
Intel Core i5  islemcisine sahiptir.
Islemci gucu  15.0  sahiptir.
```

Objeler ve Sınıflar

Örnek Kod Parçacığı

```
class Kargo:
    kargoNo = ''
    kargoDesi = ''
    kargoKg = ''

    def __init__(self, kargoNoArg, kargoDesiArg,
kargoKgArg):
        self.kargoNo=kargoNoArg
        self.kargoDesi=kargoDesiArg
        self.kargoKg=kargoKgArg

    def kargoUcretHesaplama(self):
        ucret = self.kargoDesi / 5 * self.kargoKg
        print(self.kargoNo, ' numarali kargo icin
hesaplanan ucret ', ucret)

kargo1 = Kargo(1234, 5, 10)
kargo2 = Kargo(1235, 3, 2)
kargo3 = Kargo(1236, 4, 5)
kargo4 = Kargo(1237, 6, 7)
kargo5 = Kargo(1238, 8, 9)

kargoListesi = [kargo1, kargo2, kargo3, kargo4, kargo5]

for x in kargoListesi:
    x.kargoUcretHesaplama();
```

Çıktı

```
1234  numarali kargo icin hesaplanan ucret  10.0
1235  numarali kargo icin hesaplanan ucret  1.2
1236  numarali kargo icin hesaplanan ucret  4.0
1237  numarali kargo icin hesaplanan ucret  8.4
1238  numarali kargo icin hesaplanan ucret  14.4
```

Objeler ve Sınıflar

Örnek Kod Parçacığı

```
class Kargo:
    kargoNo = ''
    kargoDesi = ''
    kargoKg = ''

    def __init__(self, kargoNoArg, kargoDesiArg,
kargoKgArg):
        self.kargoNo=kargoNoArg
        self.kargoDesi=kargoDesiArg
        self.kargoKg=kargoKgArg

    def kargoUcretHesaplama(self):
        ucret = self.kargoDesi / 3 * self.kargoKg * 2
        print(self.kargoNo, ' numarali kargo icin
hesaplanan ucret ', ucret)

kargo1 = Kargo(1234, 5, 10)
kargo2 = Kargo(1235, 3, 2)
kargo3 = Kargo(1236, 4, 5)
kargo4 = Kargo(1237, 6, 7)
kargo5 = Kargo(1238, 8, 9)

kargoListesi = [kargo1, kargo2, kargo3, kargo4, kargo5]

for x in kargoListesi:
    x.kargoUcretHesaplama();
```

Çıktı

```
1234  numarali kargo icin hesaplanan ucret  33.33
1235  numarali kargo icin hesaplanan ucret  4.0
1236  numarali kargo icin hesaplanan ucret  13.33
1237  numarali kargo icin hesaplanan ucret  28.0
1238  numarali kargo icin hesaplanan ucret  48.0
```

Sınıf tanımındaki ücret hesaplama yöntemi değiştiğinde tüm objelerdeki sonuçlar değişti.