

MUDANYA
UNIVERSITY



BMB 502 Algoritma ve Programlama

16.03.2024

Dr. Öğr. Üyesi Işıl Güzey



MUDANYA
UNIVERSITY



Nesne Tabanlı Programlama



Nesne Tabanlı Programlama

Temel Programlama Metotları

1-Prosedürel Programlama: Bir veya daha fazla prosedür; belli görevleri yerine getiren fonksiyonlara dayalı yaklaşım

!!! Program boyut ve karmaşıklığı büyüdüğünde veri ve veri üzerinde işlem yapan fonksiyonların ayrı yapıda olması sorunla oluşturmuştur.

Nesne (Object): Kendisi ile ilgili olan hem veri, hem de prosedürleri kapsayan bir yazılım birimidir

2-Nesne Tabanlı Programlama (NTP): Nesneleri gerçek hayattaki gibi; özellikleri(attribute) ve bu özellikler üzerinde yapılması söz konusu işlemler(metotlar) kapsamında ele alan yaklaşım

Nesne Tabanlı Programlama

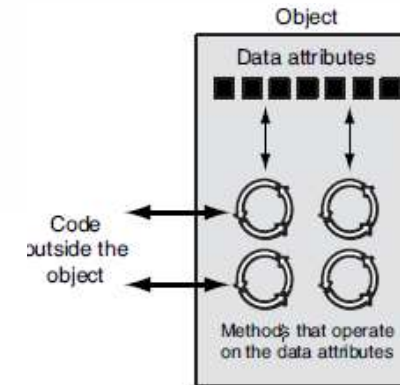
NTP, kapsülleme (encapsulation) ve veri saklama ile kod ve veri ayrımını hedefler

Kapsülleme (Encapsulation): Veri ve kodun (özellikler ve metotlar) bir nesne kapsamına birleştirilmesi

Veri Saklama (Data hiding): Nesnenin özelliklerini dışarısındaki bir programdan saklamasıdır.

!!!Sadece nesnenin metotları özelliklere erişebilir

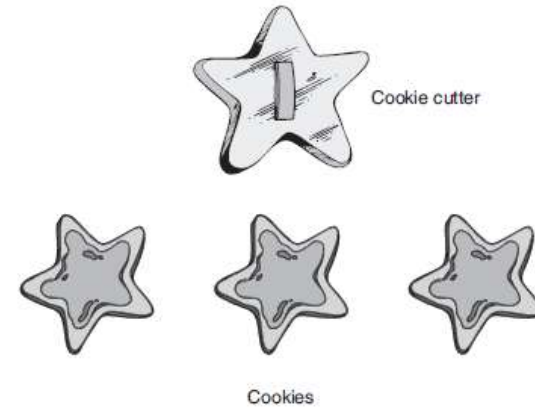
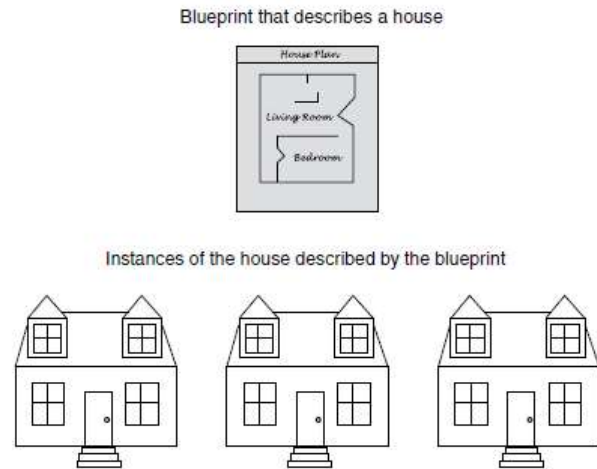
* NTP kapsamında nesnelerin yeniden kullanılabilir olmaları avantaj sağlar



Nesne Tabanlı Programlama

Nesneler yaratılmadan önce programcı tarafından, veri özellikleri ve metotlarının bir sınıf olarak tanımlanmaları gerekir.

Sınıf, nesnelerin yaratılırken kullanılacak bir yapı/kalıp/plan olarak düşünülebilir. Bu yapı kullanılarak benzer pek çok örnek/nesne yaratılabilir.



Nesne Tabanlı Programlama

Özellikler

- Saniye (0–59 arasında değer)
- Dakika (0–59 arasında değer)
- Saat (1–12 arasında değer)
- Alarm saati (Saat ve dakika değeri)
- Alarm Kurulu (True or False)

public

- saat_kur
- alarm_kur
- alarmı_aç
- alarmı_kapa

Nesnenin dışından
erişilebilir

Metotlar

private

- saniye artır
- dakika artır
- saat artır
- alarmı çal

Nesnenin iç işleyişi ile ilgili,
dışarıdan erişilemez

Nesne Tabanlı Programlama – Sınıf ve Nesne Tanımlama

- **init** metodu, Python'da varsayılan ve nesneler oluşturulurken otomatik olarak çağrılan **yapıcı(constructor) fonksiyondur**
- **self**, nesneyi gösteren referanstır

```
class Araba():  
  
    def __init__(self, model, renk, beygir_gücü, silindir):  
        self.model = model  
        self.renk = renk  
        self.beygir_gücü = beygir_gücü  
        self.silindir = silindir
```

```
araba1 = Araba("Peugeot 301", "Beyaz", 90, 4)
```

```
print(araba1.model)
```

Peugeot 301

Sınıftan nesne tanımlama

nesne_ismi = Sınıf_İsmi(parametreler(opsiyonel))

Nesnenin özelliklerine erişebilmek için

nesne_ismi.özellik_ismi

Nesne Tabanlı Programlama - Metodlar

```
class Yazılımcı():  
  
    def __init__(self, isim, soyisim, numara, maaş, diller):  
        self.isim = isim  
        self.soyisim = soyisim  
        self.numara = numara # Yazılımcı objelerinin özellikleri  
        self.maaş = maaş  
        self.diller = diller  
  
    def bilgilerigöster(self):  
        print("""  
            Çalışan Bilgisi:  
            İsim : {}  
            Soyisim : {}  
            Şirket Numarası: {}  
            Maaş : {}  
            Diller: {}  
            """.format(self.isim, self.soyisim, self.numara, self.maaş, self.diller))
```

Yeni sınıf

Metod

Nesne Tabanlı Programlama - Metodlar

```
def dil_ekle(self, yeni_dil):  
    print("Dil ekleniyor.")  
    self.diller.append(yeni_dil)
```

```
def maas_yukselt(self, zam_miktari):  
    print("Maaş yükseliyor...")  
    self.maas += 250
```

Metodlar

```
yazilimci1 = Yazilimci("Işıl", "Güzey", 12345, 3000, ["Python", "C", "R"])
```

```
print(yazilimci1.diller)
```

```
yazilimci1.bilgilerigoster()
```

```
yazilimci1.maas_yukselt(500)
```

Metodların Kullanımı

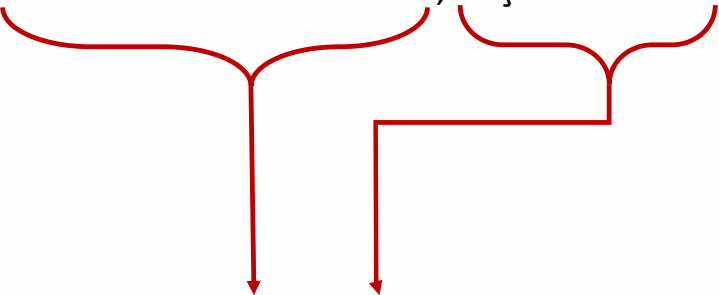
```
class Çalışan():
    def __init__(self, isim, maaş, departman):
        print("Çalışan sınıfının init fonksiyonu")
        self.isim = isim
        self.maaş = maaş
        self.departman = departman

    def bilgilerigoster(self):
        print("Çalışan sınıfının bilgileri.....")

        print("İsim : {} \nMaaş: {} \nDepartman: {} \n".format(self.isim, self.maaş, self.departman))

    def departman_degistir(self, yeni_departman):
        print("Departman değişiyor...")
        self.departman = yeni_departman
```

Kalıtım: Yeni tanımlanan bir sınıfın, başka bir sınıfın özellik ve metodlarını alması



```
class Yönetici(Çalışan): # Çalışan sınıfından miras alıyoruz.  
    Pass  
  
yönetici1 = Yönetici("Tülay Sazak",3000,"İnsan Kaynakları") # yönetici objesi
```

```
yönetici1.bilgilerigoster()
```

Önümüzdeki hafta: Overriding, özel metodlar, polymorphism, diğer uygulamalar



THANK YOU

Çağrıışan Mah. 2029 Sk. No:2 16265 Mudanya/BURSA

bilgi@mudanya.edu.tr +90(224) 224 2022 www.mudanya.edu.tr

Mudanya Üniversitesi mudanyauniversity @mudanyaedu