PYTHON PROGRAMLAMA DİLİ

NESNE YÖNELİMLİ PROGRAMLAMA

Dr. Öğr. Üyesi M. Ozan AKI

NESNE YÖNELİMLİ PROGRAMLAMA

Nesne Yönelimli Programlama (NTP)
Object Oriented Programming (OOP)

Bilgisayar programlamadaki öğelerin gerçek hayattaki nesneler gibi tüm özelliklerini tek bir yapı altında ele alınmasını sağlayan bir programlama yaklaşımıdır.

Bir nesnede, o nesne ile ilgili olan tüm veriler ve metotlar bulunur.

Veri ve metotlar açık ya da özel olarak kontrol edilebilir.

Nesneler birbirinden türetilebilir ve kalıtım yoluyla birbirlerinin nitelik ve metotlarını kullanabilir.

NESNE YÖNELİMLİ PROGRAMLAMA

Gerçek hayat problemleri sınıf şablonları kullanılarak bilgisayar ortamına daha kolay ve anlaşılabilir bir biçimde aktarılabilir.

Sınıflar ile kodlar düzenli bir biçimde saklanarak zaman kaybı yaşanmaz.

Nesne yönelimli programlamada herhangi bir projede kullanılmak üzere yaratılan bir sınıf başka projelerde tekrar kullanılabilir.

Nesne Yönelimli Programlamanın Özellikleri:

- Bilgi saklama (Encapsulation)
- Veri Soyutlama (Data Abstraction)
- Çok Şekillilik (Polymorphizm)
- Kalıtım (Inheridance)

NESNE YÖNELİMLİ PROGRAMLAMA

Class (Sınıf):

Nesnelerin niteliklerini, davranışlarını ve başlangıç durumlarını tanımlamak için kullanılan şablonlara sınıf denilir.

(Kumdan kale kalıbı)

Instance (Object):

Sınıflardan türetilen her bir gerçek kopyaya Instance ya da Object (nesne) adı verilir.

(Aynı kum kalıbından bir çok kumdan kale yapabiliriz)

Attribute (Nitelik):

Nesne altında tanımlanan değişkenlerdir. Bu değişkenlere nesnenin nitelikleri (attributes) adı verilir. Nesnenin niteliklerini tanımlar.

Method (Yöntem):

Nesne altında tanımlanan fonksiyonlardır. Bu fonksiyonlara nesnenin metotları (methods) adı verilir. Nesnenin davranışlarını belirler.

NESNE YÖNELİMLİ PROGRAMLAMA

Bir sınıf tanımlamak için;

class

ifadesi kullanılır.

Nesneye ait nitelik ve metotlar bu sınıf tanımı altında tanımlanır.

Metotların ilk parametreleri, nesnenin örneğini gösteren bir referans olmak zorundadır.

```
class sinif_adi():
    attributes
    attributes
    ...
    methods
    methods
    ...
```

NESNE YÖNELİMLİ PROGRAMLAMA

Örneğin,

İki boyutlu kartezyen koordinat sisteminde bir noktayı ifade eden sınıf tanımlamak istersek;

Nokta isimli sınıf tanımını yaptıktan sonra koordinat bilgisi, tanımlanan **x** ve **y** değişkenlerinde saklanabilir.

Burada **x** ve **y** değişkenleri **nokta** sınıfının **nitelikleridir** (**attribute**).

```
class nokta():
    x=0.0
    y=0.0
```

NESNE YÖNELİMLİ PROGRAMLAMA

Örneğin,

Daire isimli bir sınıf oluşturmak istersek;

Dairenin yarıçapını belirten **r** değişkeni bir niteliktir (attribute).

Aynı şekilde çevre ve alan değişkenleri de birer niteliktir (attribute).

Çevresini hesaplayan **calc_cevre()** ve alanını hesaplayan **calc_alan()** fonksiyonları ise sınıfın metotlarıdır.

```
class daire():
    r=0.0
    cevre=0.0
    alan=0.0

    def calc_cevre(self):
        cevre=2*3.14*self.r

    def calc_alan(self):
        alan=3.14*self.r**2
```

NESNE YÖNELİMLİ PROGRAMLAMA

Constructor (Yapıcı Metot):
Nesneleri oluştururken, ilk değerleri atama gibi işlemleri yerine getirmek için, her nesne oluşturulduğunda otomatik olarak çağırılan bir yapıcı fonksiyon vardır.

Bu yapıcı fonksiyon;

__init__()

olarak yazılır.

```
class sinif_adi():
    attributes
    ...
    def __init__(self):
        yapıcı kod bloğu
    methods
    methods
    ...
```

NESNE YÖNELİMLİ PROGRAMLAMA

Destructor (Yıkıcı Metot):
Nesneler yok edilirken, yok edilmeden önce yapılması gereken işler için yıkıcı metot otomatik çağırılır.

Bu yıkıcı fonksiyon;

__del__()

olarak yazılır.

```
class sinif_adi():
    attributes
    ...
    def __init__(self):
        yapıcı kod bloğu
    def __del__(self):
        yıkıcı kod bloğu
    methods
    methods
    ...
```

NESNE YÖNELİMLİ PROGRAMLAMA

Bir nesne oluştururken bazı nitelikleri belirtilmek istenebilir.

Bu durumda, yapıcı (constructor) metot parametreleri ile yapıcı metot içinde nitelikler tanımlanabilir.

Böylece nesne oluşturulurken bu nitelikler girilmesi gerekir.

(bazı niteliklere atama operatörü ile varsayılan değer verilerek seçimlik bırakılabilir.)

```
class daire():
    cevre=0.0
    alan=0.0

    def __init__(self, yaricap):
        self.r = yaricap

    def calc_cevre(self):
        self.cevre=2*3.14*self.r

    def calc_alan(self):
        self.alan=3.14*self.r**2
```

NESNE YÖNELİMLİ PROGRAMLAMA

Kalıtım (inheritance), bir sınıftan miras alarak yeni bir sınıf oluşturmaya kalıtım denir.

Yeni tanımlanan sınıf, ata (selef) sınıfının tüm niteliklerini ve metotlarını miras alır.

Örneğin, şekil ata sınıftan türetilen daire sınıfı, **cevre** ve **alan** niteliklerini miras alır ve kendisinde tanımlamasına gerek kalmaz.

Sadece kendine özgü yarıçap r niteliğini tanımlar.

```
class sekil():
    cevre=0.0
    alan=0.0


class daire(sekil):
    r=0.0
    def calc_cevre(self):
        self.cevre=2*3.14*self.r


    def calc_alan(self):
        self.alan=3.14*self.r**2
```

NESNE YÖNELİMLİ PROGRAMLAMA

Diğer bir örnek olarak, şekil ata sınıftan türetilen dikdörtgen sınıfı, **cevre** ve **alan** niteliklerini miras alır ve kendisinde tanımlamasına gerek kalmaz.

Sadece kendine özgü kenar uzunlukları olan **a** ve **b** niteliklerini tanımlar.

dikdortgen sınıfının alan ve **cevre** metotlarının da kendine özgü olduklarına dikkat ediniz.

```
class sekil():
    cevre=0.0
    alan=0.0


class dikdortgen(sekil):
    a=0.0
    b=0.0
    def calc_cevre(self):
        self.cevre=2*(self.a+self.b)


    def calc_alan(self):
        self.alan=self.a*self.b
```

NESNE YÖNELİMLİ PROGRAMLAMA

Eğer ata sınıf, yapıcı metodunda bir niteliğin girilmesini gerektiriyorsa, bu sınıftan türetilen halef sınıflar bu netliği sağlamalıdır.

Örneğin, **sekil** isimli sınıf **isim** niteliğini girilmesini gerektiriyorsa, bu sınıftan türetilen **daire** sınıfının yapıcı fonksiyonunun içinde bu nitelik ata sınıfın yapıcı fonksiyonuna gönderilir.

```
class sekil():
    cevre=0.0
    alan=0.0
    def __init_(self, isim)
        self.isim=isim

class daire(sekil):
    r=0.0
    def __init__(self,isim)
        sekil.__init__(self, isim)
    def calc_cevre(self):
        self.cevre=2*(self.a+self.b)
    def calc_alan(self):
        self.alan=self.a*self.b
```

NESNE YÖNELİMLİ PROGRAMLAMA

Ata sınıfın nitelik ve metotlarına ulaşmak için alternatif olarak;

super()

Fonksiyonu kullanılabilir.

super() fonksiyonundan sonra self referansı belirtmeden ata sınıfın nitelik ve metotlarına erişilebilir.

```
class sekil():
    cevre=0.0
    alan=0.0
    def __init_(self, isim)
        self.isim=isim


class daire(sekil):
    r=0.0
    def __init__(self,isim)
        super().__init__(isim)
    def calc_cevre(self):
        self.cevre=2*(self.a+self.b)
    def calc_alan(self):
        self.alan=self.a*self.b
```

NESNE YÖNELİMLİ PROGRAMLAMA

Bir ata sınıftan türetilen halef sınıf eğer ata sınıftaki bir metodu aynı isimle yeniden tanımlarsa, ata sınıfın metodu türetilen sınıfın metodu tarafından ezilir (**override**) ve artık yeni tanımlanan metot geçerli olur.

```
class ata_sinif():
    def yazdir(self):
        print("ata sinif mesaji.")

class turetilen_sinif(ata_sinif):
    def yazdir(self):
        print("turetilen sinif mesaji.")
```