

Nesne Tabanlı Programlama

- Nesnenin özelliklerinin bütününe durum (state), metotların bütününe de arayüz (interface) denir.
- Durum ve arayüz sınıfta tanımlanır, sınıftan türetilen nesne durum bilgisine sahiptir ve arayüz ile tarif edilen davranışları yerine getirir.
- Nesneler, yazılım sisteminin yerine getireceği sorumlulukları paylaşır.

- Sınıf, Java'nın en geniş bloğudur.
- Java'da soyut sınıflar, iç sınıflar gibi farklı yapıda ve özellikte sınıflar olduğu gibi, sınıf hariç enum ve interface isminde farklı tip oluşturma yapıları vardır.
- Nesneler, heap alanında tutulur. Nesnelerin değişkenleri bu alanda saklanır. JVM tarafından bu alan dinamik olarak kullanılabilir.
- Java'da nesnelere sadece referansları ile ulaşılır.

- Java'da değişkenler ilk değer atamadan kullanılamaz. Fakat değişkenler eğer nesne değişkenleri ise bunlara varsayılan değer atanır.

Bileşik Nesne

- Farklı türden nesnelerden oluşan nesnelere, bileşik nesne denir.
- Sınıf diğer sınıfların nesne değişkenlerine sahiptir(sınıfta tanımlanan String tipli değişken) veya nesne diğer nesnelerin referansına (kendi oluşturduğumuz iki sınıf) sahiptir.

Örnek

- Öğrenci ve Ders isimli iki sınıf olsun. Öğrenci sınıfının öğrenciNo ve İsim isminde iki değişkeni, Ders sınıfında dersId ve dersAdı isminde iki değişkeni olsun. Bu sınıflar bileşik nesnede 1-1 ilişki olacak şekilde oluştursunlar. Bu sınıflardan gelen bilgiler ile main kısmında dersi alan ve ders bilgileri ekrana yazılsın.

Örnek

- Bir önceki örneğe bir de bölüm sınıfı oluşturup çoklu ilişki ile ilgili sınıfları kurgulayın.

İmza (Signature) ve Arayüz (Interface)

- Bir metodun isim ve parametrelerinden oluşan bilgisine imza denir.
- Bunlara ek dönüş değer tipi ve sıra dışı durumlardan oluşmasına arayüz denir.
 - `Public double hesapla (double arg) throws IllegalArgumentException`
- Overloading işleminde birden fazla alternatif üst tip varsa kendisine en yakın tipe sahip olan çağırılır.

Değişken Sayıda Parametre Alma

- Java'da değişken sayıda parametre ... ile verilir.
 - `void hesapla (int ... sayılar) { }`

Ekran Çıktısı?

```
class Hesapla
{
    public void kare(int i)
    {
        i = i * i;
    }
}
```

Main Kısmı:

```
public static void main(String[] args)
{
    Hesapla nesne = new Hesapla();
    int a = 20;
    System.out.println(a);
    nesne.kare(a);
    System.out.println(a);
}
```

Ekran Çıktısı?

```
class Hesapla
{
    int i=20;
    public void kare(Hesapla a)
    {
        a.i=a.i*a.i;
    }
}
```

Main Kısmı:

```
public static void main(String[] args)
{
    Hesapla nesne = new Hesapla();
    nesne.kare(nesne);
    System.out.println(nesne.i);
}
```

Durumlu ve Durumsuz Nesneler

- Yapılan iş neticesinde verileri tutan nesnelere durumlu (hafızalı), tutmayanlara durumsuz (hafızasız) nesne denir.

- Fonksiyon uygulamaya ait herhangi bir durum bilgisini tutmaz ve değiştirmez.
- Kullanmak kolaydır.
- Durumsuz nesnelerde durumla ilgili bilgi alıp veren set/get metotları olmaz.
- Bir fonksiyon, geçici olarak durumu tutan local değişkenlere sahip olabilir. Ama bu fonksiyon kesinlikle sınıf veya ait olduğu fonksiyonda tanımlı olan değişkenlere ve nesnelere referans etmemeli.
- Durumsuz nesnelerin uygulama içerisinde bulunan ve fonksiyonun dışındaki herhangi bir değişkenin durumunu değiştirmedeği için herhangi bir yan etkisi bulunmamaktadır.

- *public class Stateless*
 {
 public int topla(int a, int b) {
 return a + b;
 }
 }

- Durumlu nesneler bir süreç yönetir.
- Kullanmak zordur.
- Durumlu nesneler yaptıkları işi bir dizi metot çağrısından oluşan bir süreç ile yerine getirir. Metot süreç adımını yerine getirir ve sadece o adım için veriyi parametre şeklinde alır.
- Durumlu nesnelerde set/get metotları olur.
- Sonrası için bellek görevi görürler fakat çok hafıza kaplayabilirler.

```

public class Stateful
{
    int a;
    int b;
    int c;
    char işlem;
    public int setA (int x) {
        a=x;
    }
    public int getA ( ) {
        return a;
    }
    ,
    ,
    ,
    ,
    ,
    ,
    }

```

```

Public static void main (String [] args)
{
    Stateful nesne=new Stateful();
    nesne.setA=3;
    nesne.setB=5;
    nesne.setişlem='+';
    int sonuç=nesne.getSonuç( );
    ,
    ,
    ,
    ,
    }

```

- Çok istemcili durumlarda durumlu nesnelerde istemci sayısı kadar nesne gerekecektir ve bunlar birbirine karışabilir.
- Bundan kaçınmak için istemci sayısı kadar nesne oluşturulmalı ya da senkronizasyon gibi işlemler yapılmalıdır.
- Ölçeklenebilirlik önemli ise durumsuz nesne kullanılmalıdır.