

Nesne Tabanlı Programlama

Var

- Java 10 ile ortaya çıkmış bir yapıdır.
- Var olarak tanımlanan değişkene tip bilgisi vermeye gerek yoktur. Java tip bilgisini ortamdan kendi çıkarır (tip çıkarımı).
- Yerel değişken oluşturmada kullanılır. Üye değişken olarak sınıfta tanımlanamaz.

`var i=5; //doğru`

`var j; //yanlış`

Unicode

- Java Unicode bir dil olduğu için Türkçe karakter kullanımına olanak vermektedir. Fakat yine de evrensel olabilmesi için İngilizce alfabe kullanılması tavsiye edilmektedir.

Compile Time ve Run Time arasında Sabit Tanımlama Farkı

- Integer bir değişkenin max değeri 2147483647'dir. Buna göre compile time ve run time arasında fark meydana gelmektedir.

Sınıflar

- Java'da her şey sınıf içinde olmalıdır. Çalışacak kodlar muhakkak sınıf içinde olur.
- Metot blokları, inner class blokları, ilk değer atama blokları sınıf bloğu içinde yer alır.
- Sınıftan nesne oluşturulunca, bellekte nesneye yer ayrılır ve bu yerin adresi nesnenin referansına atanır. (sınıf referans=new sınıf()).

Sınıf ve Nesne

- Sınıf, kendinden üretilecek nesneler için bir şablondur.
- Nesneler, sınıfların çalışma zamanında bellekte oluşan instance'lardır.
- Sınıflar ile tipler oluşturulur. Aynı sınıftan üretilen nesneler aynı tiptedir.
- Java'da nesnelere sadece referansları ile ulaşabiliriz.

Referans

- Referans nesnelere soyut erişim mekanizmasıdır.
- Bir referans bir anda sadece bir nesne gösterebilir.
- Bir nesnenin birden fazla referansı olabilir.
- Bir referans farklı zamanlarda aynı tipten farklı nesneleri gösterebilir.

Örnek

- Ev isminde bir sınıfımız olsun ve bu sınıfın sokak ve numara isimli değişkenleri ve yaz isminde metodu olsun. Daha sonra bu sınıftan yazlıkEv ve kışlıkEv isminde iki nesne türetelim. Daha sonra bu nesnelerin yerlerini değiştirerek sabit değişkenler ile referans değişkenler arasındaki farka bakalım.
- Benzer şekilde Ev sınıfından oluşturacağımız tmp ve yazlıkEv referansının değişken değerlerinin değerlerini gözlemleyelim.

Ekran Çıktısı?

```
public static void main(String[] args)
```

```
{  
    Ev yazlikEv = new Ev();  
    yazlikEv.yaz();  
  
    yazlikEv.kapiNo = 334;  
    yazlikEv.sokakAdi = "Tufan";  
    yazlikEv.yaz();  
  
    Ev kislikEv = new Ev();  
    kislikEv.yaz();  
}
```

```
public class Ev
```

```
{  
    public int kapiNo = 444;  
    public String sokakAdi = "Meltem";  
    public void yaz()  
    {  
        System.out.println ("Adres:" + sokakAdi + "No:" + kapiNo);  
    }  
}
```

- Sınıf değişkenlerinin değeri nesnelere aynen iletilmekte. Program çalışırken o değeri nesne ile değiştirmek mümkün, ama nesne içindeki değişiklik, sınıf tanımındaki değerleri etkilemez.

Null ifadeler

- Hiçbir referansı olmayan nesneye kayıp nesne denir.
- Kayıp nesneler garbage collector (çöp toplayıcı) tarafından temizlenir.

İÇ İÇE SINIFLAR (NESTED CLASS)

- Nesneye yönelik programlama dilleri kullanıcıya bir sınıf içerisinde başka bir sınıf tanımlama olanağı sağlar. Bu tür sınıflara Nested Class(Gömülü Sınıf) adı verilir. Gömülü sınıflar statik ve statik olmayan gömülü sınıflar olmak üzere ikiye ayrılırlar. Statik olmayan gömülü sınıflara Inner Class(İç Sınıf) adı verilir.
- Aynı isim uzayında olması gerekir.

Inner Class Haricinde;

- public sınıf, kendi ismini taşıyan bir dosyada yer almak zorundadır.
- Yani her Java kaynak kodu dosyasında en fazla bir tane public sınıf olabilir.
- Java kaynak kodu dosyasında public olmayan pek çok sınıf olabilir fakat mümkünse her biri ayrı kendi dosyasında kurgulanmalıdır.
- Java kaynak dosyasının ismi ne olursa olsun derlendiğinde içinde bulunan tüm sınıflar için .class dosyası oluşur.

- Gömülü sınıfların üç temel faydası vardır. Bunlar:
 - **Sınıfların Mantıksal Gruplanması** : Ortak iş yapan sınıfların bir arada bulunmasını sağlayarak daha sağlıklı yazılım programlama arayüzleri oluşturulmasını sağlar.
 - **Daha İyi Kapsülleme** : Sınıf değişkenlerinin private(özel) kalmasını sağlayarak daha iyi kapsülleme sağlar.
 - **Kod Okunabilirliği** : Üst seviye sınıflar altında iş gören gömülü sınıflar kod okunabilirliğini arttırır.

This anahtarı

- Aynı kapsamda aynı isimde bir üye ve yerel değişken olabilir. Bu durumda yerel değişken üye değişkeni gölgeler.
- Java bu durumda olan bir değişkene ulaşımda, en yakın tanıtilan yerel değişkene erişir. Üye değişken için ayrı bir yapı kullanılır.

Ekran çıktısı?

```
public static void main(String[] args)
{
    yazarlar a = new yazarlar();
    a.yaz();
}
```

```
class yazarlar
{
    public String ad= "George R.R. Martin";
    public void yaz()
    {
        String ad = "Tolkien";
        System.out.println(ad + "," + this.ad);
    }
}
```