Sınıf ve Nesne - I

- “*Köpek dediğin sadık olmalı.*” ya da “*Köpekler çok sadık hayvanlardır.*” cümlelerinde, kavramsal, soyut olarak köpek sınıfı, cinsi kastedilir ve söylenen, var olmuş ve olacak bütün köpekler için geçerlidir.
- “*Komşunun köpeği çok tatlı.*” cümlesinde ise köpek sınıfının bir örneği ya da nesnesi, somut olan bir canlıdan bahsediyoruz demektir.
- Köpek, ilk durumda bir sınıfı, ikinci durumda ise bir nesneyi temsil eder.

Sınıf ve Nesne - II

- Bu anlamda sınıf (class), soyut olanı (abstract), tümeli (universal), örneklerinin hepsine genel (general) olanı temsil eder.

- Nesne ise, somut olanı (concrete), yani soyut olanın ete kemiğe bürünmüş halini, tikeli (particular), bir örneği (instance) temsil eder.

Nesne (Object) - I

- **Nesne** (**object**), insan zihninin algıladığı ve özellikleri olan herhangi bir kavramsal ya da fiziksel şeydir:
  - Öğrenciler, derslere devam ediyorlar.
  - Öğretmen, sınıfta öğrencileri dinliyor.
  - Dersler yarın başlıyor.

Nesne (Object) - II

- Nesnelerin özellikleri vardır ve bu özellikler, nesnelerin **durum**unu (**state**) tarif ederler.
- Ayrıca nesnelerin **davranış**ları (**behavior**) vardır:
  - Sarı boyalı sınıfta öğrenci şiir okuyor.
  - Kırmızı top suya yuvarlandı.
- Dolayısıyla nesneler, bilirler ve yaparlar.

Sınıf (Class) - I

- Sınıf, nesneler için bir kalıptır, şablondur, yani kendisinden üretilecek olan nesnelerin sahip olacağı özellikler ile davranışları tarif eder.
- Sınıf, kendisinden türetilecek olan nesnelerinin
  - özelliklerini değişik tiplerde değişkenlerle (variables),
  - davranışlarını ise metotlarla (method) (fonksiyon (function)/prosedür (procedure))

  ifade eder.

Sınıf (Class) - II

- Nesnenin özelliklerinin bütününe **durum** (**state**), metotların bütününe de **arayüz** (**interface**) denir.

- Durum bilgisi ve arayüz, sınıfta tanımlanır, sınıftan üretilen nesneler ise durum bilgisine sahip olurlar, arayüzle tarif edilen davranışları yerine getirirler.

Sınıf (Class) - III

- Aynı sınıftan üretilen nesneler aynı tipte olurlar:
  - Aynı özelliklere sahiptir ama özelliklerin değerleri değişebilir,
  - Aynı davranışlara sahiptir,
- Davranışlar genelde duruma bağlı olduğundan, farklı durumdaki nesnelerin davranışları da farklı olur.

Yazılımın Nesnesi

- Yazılımın nesnesi ise hayatımızdaki kavramsal ya da fiziksel bir nesneyi temsil etmek üzere, onun özelliklerini ve davranışlarını ifade eden yapıdır:
  - Yazılımın nesnesi, temsil ettiği gerçek dünyadaki nesnenin durumunu, sınıfında tanımlanan değişkenlerle, davranışlarını da metotlarla yerine getirir.

Durum (State)

- Nesnelerin **durum**u (**state**) ile daha çok durağan (static) özellikleri kastedilir ve programlama dillerinde farklı tiplerde bir grup değişken ile ifade edilir.
- Nesnenin durumunu oluşturan her bir ayrık bilgiye ise **özellik** (**attribute, property**) denir:
  - Öğrenci: No, isim, soy isim, doğum tarihi, cinsiyet, adres, bölüm, aldığı dersler, vs.
  - Ders: No, isim, bölüm, veren kişi, kredi sayısı, vs.

Davranış (Behavior)

- Nesneler davranırlar, belli işleri yerine getirirler.

- Yazılım nesnelerinin davranışlarına, yerine getirdiği **sorumluluk** (**responsibility**), verdiği **hizmet** (**service**) ya da aldığı **mesaj** (**message**) olarak bakmak, işimizi kolaylaştırır:

  - Öğrenci: Kayıt olur, ders alır, sınava girer, vs.

  - Ders: Öğrencinin kaydolmasına/bırakmasına izin verir, ön şart dersleri hakkında bilgi verir, vs.

Sınıflar ve Nesneleri

selsoft

build better, deliver faster

Sınıf - I

- Sınıf, kendisinden üretilecek nesnelerin kalıbıdır - şablonudur.
  - Aynı sınıftan üretilen nesnelerin tipi, aynıdır.
- Sınıf, nesnelerinin özelliklerini (attributes, properties) ve davranışlarını (behavior) tanımlar.
  - Nesnelerin özellikleri, değişkenlerle (variables),
  - Nesnelerin davranışları ise metotlarla (methods) tanımlanır.

Sınıf - II

- Nesnenin özelliklerinin bütününe **durum** (**state**), metotların bütününe de **arayüz** (**interface**) denir.
- Durum ve arayüz, sınıfta tanımlanır,
- Sınıf, Java'ın en geniş bloğudur.

Sınıf ve Tip

- Sınıf, bir tip tanımlama yapısıdır.
  - Ama Java'daki tek tip tanımlama yapısı sınıf değildir.
- Java'da **soyut sınıflar** (**abstract classes**) ve **iç sınıflar** (**nested classes**) gibi farklı yapıda ve özellikte sınıflar olduğu gibi, sınıfın dışında, **sıralama** (**enum**) ve **arayüz** (**interface**) isminde farklı tip oluşturma yapıları da mevcuttur.
- java'ya yeni sürümlerinde **kayıt** (**record**) isimli yeni bir tip daha gelecektir.

Sınıf Tanımlama - I

- Java'da sınıf tanımlamak için **`class`** anahtar kelimesi kullanılır:

```
<niteleyici>* class <SınıfİsmI>{
        <değişken>*
        <kurucu>*
        <metot>*
}
```

- Sınıfın, sıfır ya da daha fazla niteleyicisi (modifier) olabilir.
- Sınıfın, geçerli ve anlamlı bir ismi olmalıdır.

Sınıf Tanımlama - II

- **Kurucu** (ya da **yapılandırıcı)** (**constructor**), nesne yaratılırken çağrılan ve nesnenin ilk durumunu belirlemek (initialization) için kullanılan özel bir metotdur.

- Zorunlu olmamakla birlikte sınıfın öğeleri, sınıf içinde yukarıdan aşağıya doğru özellikler, kurucular ve metotlar olarak sıralanır.

Nesne Yaratmak - I

- Nesne yaratma ve daha sonra erişmek için bu nesneyi bir referansa atama, aslen bir değişken tanımlamadır.
  - Önce bir değişken olarak referans tanıtılmalı sonra da göstereceği nesne oluşturularak referensa atanmalıdır.
  - İlkel tipte değişken tanımladan en önemli farkı `new` anahtar kelimesi ve kurucu çağrısıdır.

```
Değişken Tanımlama = Değişken Tanıtma + İlk Değer Atama
```

```
Variable Definition = Variable Declaration + Initialization
```

```
Car carObject = new Car();
```

Nesne Yaratmak - II

- Nesne yaratmanın detayları şöyledir:
  - **Tanıtım** (**Declaration**): Yaratılacak nesneyi gösterecek referans değişkeni tanıtılır.
  - **Yaratma** (**Instantiation**): `new` anahtar kelimesi kurucu çağrısı yapmak için kullanılır.

```
Car carObject = new Car();
```

  - **Başlangıç durumuna getirme** (**Initialization**): Kurucu çağrısı ile nesne yaratılır ve başlangıç durumuna getirilir.
  - **Atama** (**Assignment**): Nesnenin adresi referansa atanır.

Nesne Yaratmak - III

- `new` işlemcisi, daima kurucu çağrısından önce ve sadece onunla kullanılır.
  - Java'da bir kaç istisna dışında nesneler daima `new` ve takip eden kurucu çağrısıyla oluşturulur.
    - Bunun en bilinen istisnaları `String` ve dizilerdir (arrays).
- Her nesne, bellekte (heap) ayrı bir alana sahiptir, nesne özelliklerinin değerleri, heapte, nesneye ayrılan bellek alanında tutulur.

Nesne Yaratmak - III

- Oluşturulan nesnenin tipi kurucu çağrısıyla belirlenir.
- Nesnenin tipi ile referansının tipi aynı olmalıdır.
  - Şimdilik bunu bu basitlikte kabul edelim.

Nesne Yaratmak - IV

- Nesne oluşturmak ve bir referansa atamak, daha önce ele alındığı gibi, tanıtma ve ilk değer atamadan oluşur.

- Bu iki adım, tek bir yerde yapılabileceği gibi önce referans tanıtımı bir yerde, daha sonra da nesneyi yaratma, başlangıç durumuna getirme ve atama topluca başka bir yerde yapılabilir.

```
Car carObject;              // Step 1
...
carObject = new Car(); // Step 2, 3 & 4
```

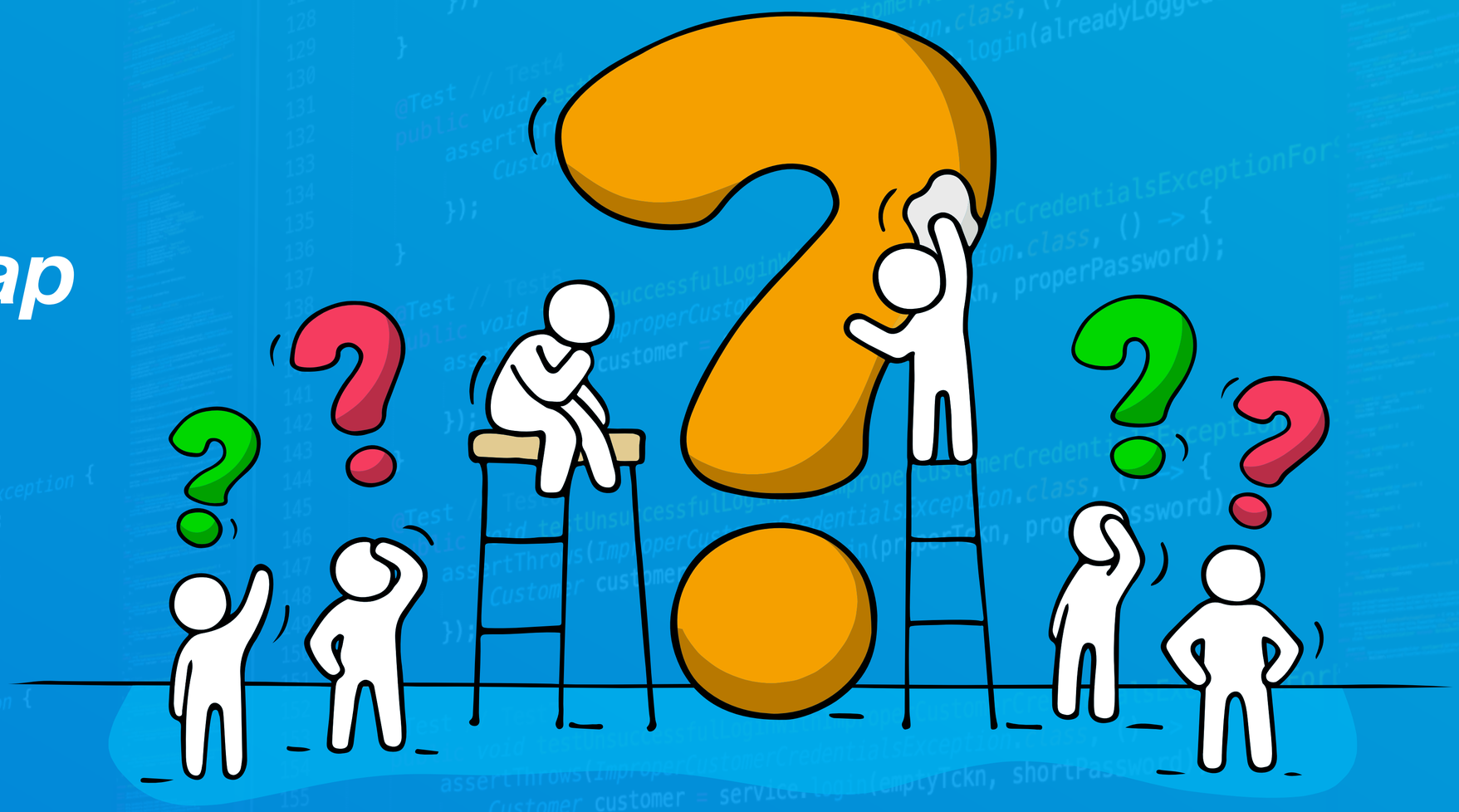
Heap

- Nesneler, **heap** adı verilen ve RAM'in JVM tarafından dinamik olarak kullanılabilen alanında oluşturulurlar.
  - Nesnelerin özellikleri olan değişkenler bu alanda yaşarlar.
- Java'da, bu bellek alanına doğrudan ulaşmamız ya da müdahale etmemiz mümkün değildir.

Referans

- Java’da, nesnelere sadece **referans**larıyla (**reference**) ulaşılır.
- Referansa aşağıdaki isimler de verilir:
  - **Tutaç** (**handle**)
  - **İşaretçi** (**pointer**)
- Java’daki referanslar, C++ pointerlarının soyut halidir.
  - C++’ta, pointerlarla nesnelerin fiziksel adreslerine ulaşılabilir ve pointer aritmetiği yapılabilir.

Soru ve Cevap Zamanı!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

Nesnenin Durumu: Alanlar

selsoft

build better, deliver faster

Değişkenlerin Rolleri (Tekrar)

- Java'da değişkenler, ilkel olsun referans olsun, fonksiyonellik ya da rol açısından üçe ayrılırlar:
  - **Nesne değişkenleri** (**instance** ya da **object variables**): Nesnenin durumunu (state) oluşturan değişkenlerdir.
  - **Sınıf değişkenleri** (**class variables**): Nesnelerin ortak durumunu ifade eden değişkenlerdir.
  - **Yerel değişkenler** (**local variables**): Geçici değişkenlerdir.
- İlk ikisine **üye değişkenler** (**member variables**) denir.

Üye Değişkenler

- Yerel değişkenler daha önce geniş bir şekilde ele alınmıştı.
- Burada üye değişkenleri ele alacağız:
  - Önce nesne sonra da sınıf değişkenlerini inceleyeceğiz.

Nesne Değişkenleri - I

- **Nesne değişkenleri** (**instance** ya da **object variables**), fonksiyonel olarak, nesnenin özelliklerini ifade ederler.
  - Nesne değişkenlerine **alan** (**field**) da denir.
  - Nesne değişkenlerine bazı durumlarda **özellik** (**property**) de denir.
- Nesne değişkenleri, yapısal olarak referans değişkeni olabildiği gibi basit değişken de olabilir.
- Eğer nesne değişkeni bir referans tip ise bu durumda o da başka bir sınıftan oluşturulmuş bir nesneyi gösterir.

Nesne Değişkenleri - II

- Nesne değişkenleri, sınıfın içinde ama metot ya da başlatma bloğu (initializer block) gibi herhangi bir alt blok dışında, herhangi bir yerde tanıtılmalıdır,
- Genelde sınıfın en başında tanıtılırlar.
- Nesne değişkenlerinin kapsamı (scope), tüm sınıftır.
  - Fakat nesne değişkenlerinde de ileri referans (forward reference) yapılamaz.

- **`Car`** sınıfında 5 tane nesne değişkeni, 4 tane metot vardır.
  - Hiç sınıf değişkeni yoktur.
- Nesne değişkenleri, sınıf bloğunda ve sınıfın başında tanıtılmışlardır.

```
public class Car{

  public String make;
  public String model;
  public String year;
  public int speed;
  public int distance;

   public void go(int newDistance) {
     …
  }

  public void accelerate(int newSpeed) {
     …
  }

  public void stop() {
     …
  }

  public String getInfo() {
     …
  }
}
```

Nesne Değişkenleri - III

- Nesne değişkenleri ilk değerlerini tanıtılırken alabildikleri gibi daha sonra bir metot içinde, genelde de kurucu metotta alabilirler.
- Tanımlanacak nesne değişkeni sayısında bir kısıtlama yoktur.
- Ama sağlıklı bir tasarım için onlarca (yüzlerce!) değişkene sahip sınıflardan kaçınılmalıdır.

Car Sınıfı - I

- Aşağıda sadece özelliklerden oluşan ve hiç bir davranışa sahip olmayan `Car` sınıfı tanımlanmıştır.

```
// Attribute only Car
public class Car {
   String make;
   String model;
   String year;
   int speed;
   int distance;
}
```

Car Sınıfı - II

- `Car`'ın özellikleri, basit ve karmaşık veri tiplerinden değişkenlerle betimlenmiştir.
- `Car`'ın nesne değişkenleri sadece tanıtılmışlardır, bir ilk değer atanmamıştır.

Özelliklere Erişim

- Nesnenin özelliklerine nesnenin referansı yoluyla erişilir.
- Erişim "." notasyonuyla olur:

```
reference.attribute
```

- Bu şekilde erişilen özelliğin değerine ulaşılabileceği gibi tipine uygun atamalar da yapılabilir:

```
Car carObject = new Car();
carObject.speed = 60;
System.out.println(carObject.speed);
```

```
public class Car{
  String make;
  String model;
  String year;
  int speed;
  int distance;
}
```

```
public class Test {
  public static void main(String[] args) {
     Car myCar = new Car();
     myCar.make = "Mercedes";
     myCar.model = "E200";
     myCar.year = "2011";
     myCar.speed = 80;
     myCar.distance = 37_650;
     System.out.println("My Car: " + myCar.year + " " +
                              myCar.make + " " + myCar.model)

     Car yourCar = new Car();
     yourCar.make = "Toyota";
     yourCar.model = "Camry";
     yourCar.year = "2011";
     yourCar.speed = 0;
     yourCar.distance = 60_000;
     System.out.println("Your Car: " + yourCar.year + " “
                          + yourCar.make + " " + yourCar.model);
  }
}
```

Car ve Test

- `org.javaturk.oopj.ch08.car.attribute.Car`
- Bu örnekteki `Car` sınıfında hiç metot yoktur; sadece nesne özellikleri yani alanlar vardır.

Nesne Değişkenlerinin İlk Değeri - I

- Az önceki örnekte, oluşturulan `Car` nesnesinin alanlarına (field) hiç bir ilk değer verilmediği halde hata alınmadığını hatta derleyicinin onlara varsayılan bir değer atadığını farkettiniz mi?
- Ama biz daha önce Java'da bir değişkene ilk değer atamadan, yani onu tanımlamadan, kullanamayacağımızı söylemiştik.

Nesne Değişkenlerinin İlk Değeri - II

- Değişkenler, bir ilk değer atanmadan kullanılamaz.
  - Fakat bu durum sadece yerel değişkenler için geçerlidir.
- Java'da nesne değişkenlerine bir ilk değer atamasak bile, derleyici onlara varsayılan bir ilk değer verir.
  - Bu ilk değerler, değişkenlerin tipleriyle uyumlu ve en az bilgi sağlayan değerlerdir.
- **Soru**: Sizce, Java, yerel değişkenlere bir ilk değer vermezken neden nesne değişkenlerine bir ilk değer veriyor?

Nesne Değişkenlerinin İlk Değeri - III

- Nesne değişkenlerine otomatik olarak atanan ve onlara tipleriyle uyumlu ve en az bilgi sağlayan ilk değerler şunlardır:

```
boolean   false
byte      0
char      ''  (int olarak 0)
short     0
int       0
long      0   (64 bit)
float     0.0 (32 bit)
double    0.0 (64 bit)

reference null
```

InitialValues

- `org.javaturk.oopj.ch08.init.InitialValues`

Nesne Durumu - I

- Java'nın nesne değişkenlerine varsayılan bir ilk değer vermesinin en temel sebebi, nesnenin daima anlamlı bir durumda olmasını sağlama isteğidir.
- Nesnelerin durumlarının anlamlı olması programcının sorumluluğundadır.
  - Java derleyicisi varsayılan değerlerle buna destek olur.

Nesne Durumu - II

- Nesnenin durumu, varsayılan ilk değerlere rağmen çoğunlukla anlamsız olacaktır.
  - Örneğin referansların varsayılan `null` ilk değerleri ya da cinsiyeti gösteren `char` tipinin `0` olan ilk değeri.
- Nesnenin oluşturulduğunda anlamlı halde olmasını sağlamanın yolu, akıllı kurucu (smart constructor) çağrısı yapmaktır.
  - Bunu ileride ele alacağız.