Laboratuvar Çalışması

Yapıcı Metotlar (Constructor) ve Static Anahtar Sözcüğü

Arş. Gör. Ömer Miraç KÖKÇAM
TF Yazılım Mühendisliği – YMT1102

28 Nisan 2026

Yönerge

- Bu lab çalışması **2 sorudan** oluşmaktadır. Her iki soruyu da çözmeniz beklenmektedir.
- Kodlarınızı bir IDE'de (IntelliJ IDEA, Eclipse veya VS Code) yazıp **derleyip çalıştırın**.
- Soru 1 için `Kitap.java` ve `TestKitap.java`; Soru 2 için `Arac.java` ve `TestArac.java` dosyalarını oluşturun.

Soru 1: Constructor Overloading – Kütüphane Sistemi (50 puan)

Bir kütüphane yönetim sistemi için `Kitap` sınıfı tasarlayacaksınız. Kullanıcılar kitabı sisteme *üç farklı şekilde* kaydedebilmelidir:

1. Sadece kitabın **adı** biliniyor (yeni gelen kitap).
2. Kitabın **adı ve yazarı** biliniyor.
3. Kitabın **adı, yazarı ve sayfa sayısı** biliniyor (tam kayıt).

Yapılacaklar:

1. `Kitap` sınıfını aşağıdaki alanlarla oluşturun: `String kitapAdi`, `String yazar`, `int sayfaSayisi`.
2. Yukarıdaki üç senaryoya karşılık gelen **üç ayrı constructor** yazın (constructor overloading).
3. Atanmayan alanların varsayılan değerleri korunacaktır (`String` → `null`, `int` → `0`).
4. `void bilgileriGoster()` adında bir metot yazın; bu metot kitabın tüm bilgilerini ekrana yazsın.
5. `TestKitap` sınıfında `main` metodu içinde **üç farklı kitap nesnesi** oluşturun (her constructor için bir tane) ve bilgilerini ekrana yazdırın.

Beklenen Çıktı (örnek)

```
Kitap: Suç ve Ceza | Yazar: null | Sayfa: 0
Kitap: Beyaz şDi | Yazar: Jack London | Sayfa: 0
Kitap: Sefiller | Yazar: Victor Hugo | Sayfa: 1488
```

İpucu

Yalnızca `kitapAdi` parametresi alan constructor'da `yazar` alanına *hiçbir şey atamayın* – otomatik olarak `null` kalacaktır. Aynı şekilde `sayfaSayisi` otomatik `0` olur. Bu *varsayılan değer* davranışıdır, bug değildir.

Soru 2: Static Değişken ve Otomatik ID Üretimi (50 puan)

Bir araç kiralama firmasının sistemini tasarlıyorsunuz. Sisteme her yeni araç eklendiğinde:

- Araca **otomatik bir plaka numarası** atanmalıdır (1'den başlayarak artan).
- Sistemdeki **toplam araç sayısı** her an bilinmelidir.
- Tüm araçlar aynı **firma adını** paylaşır (her araçta ayrı kopya tutulmamalı!).

Yapılacaklar:

1. `Arac` sınıfını oluşturun ve aşağıdaki alanları doğru *static / instance* ayrımıyla tanımlayın:
   - Tüm araçlarda paylaşılan: `firmaAdi` (String), `toplamAracSayisi` (int), `sonPlakaNo` (int).
   - Her araca özel: `plakaNo` (int), `marka` (String), `model` (String).

2. `Arac(String marka, String model)` constructor'ı yazın. Bu constructor:
   - `sonPlakaNo` değişkenini bir artırsın ve bunu `plakaNo` olarak atasın.
   - `toplamAracSayisi` değişkenini bir artırsın.
   - `marka` ve `model` alanlarını parametrelerden alsın.
3. `void goster()` metodu yazın; aracın firma adı, plaka, marka ve modelini yazdırsın.
4. `TestArac` sınıfında:
   - `Arac.firmaAdi` alanına `"FrtCar Kiralama"` değerini atayın.
   - En az **3 farklı araç** oluşturun.
   - Her aracın bilgisini `goster()` ile yazdırın.
   - En sonda **toplam araç sayısını** ekrana yazdırın (`Arac.toplamAracSayisi` – *sınıf adı üzerinden* erişin, nesne üzerinden değil!).

Beklenen Çıktı (örnek)

```
FrtCar Kiralama | Plaka: 1 | Toyota Corolla
FrtCar Kiralama | Plaka: 2 | Renault Megane
FrtCar Kiralama | Plaka: 3 | Ford Focus
Toplam araç ıısays: 3
```

Düşündürücü Sorular (rapor için kısaca yanıtlayın)

1. `firmaAdi` alanını `static` yapmasaydık ne olurdu? 1000 araç için bellekte ne değişirdi?
2. `sonPlakaNo` alanını `static` yerine *instance* olarak tanımlasaydık plaka numaraları nasıl olurdu? Neden?
3. `Arac.toplamAracSayisi` ile `a1.toplamAracSayisi` arasında *çalışma açısından* fark var mıdır? Hangisi *best practice*'tir, neden?