

FLUTTER

Sınıf ve Nesne Kullanımı

17. hafta • Modelleme ve Listeler

36 hafta, 72 bağımsız çalışan Flutter projesi, gerçek uygulama ekranları, uygulamalı dersler ve öğrenci geri bildiriminden sonra incelenebilen çözüm kaynakları.

Bu kitap güncel ders kaynaklarından otomatik üretilmiştir; özet ya da jenerik şablon değildir.

Öğrenme Haritası

Hafta	Ders	Projeler
17	Sınıf ve Nesne Kullanımı	Öğrenci Modeli • Kitaplık Modeli

17. Hafta — Sınıf ve Nesne Kullanımı

Bu hafta ne yapacağız?

Bir öğrencinin numarası, adı, sınıfı ve notları birbiriyle ilişkilidir. Bu bilgileri ayrı değişkenlerde tutmak yerine tek bir Öğrenci nesnesinde birleştireceğiz. Ardından nesneleri bir listede gösterecek, seçilen öğrencinin ayrıntısını açacak ve yeni not eklenince ortalamanın kendiliğinden güncellenmesini sağlayacağız.

Haftanın ikinci uygulamasında aynı düşüncüyü bir sınıf kitaplığına taşıyacağız. Kitapları modelleyecek, ödünç verme/iade işlemini gerçekleştirecek ve yalnız raftaki kitapları filtreleyeceğiz.

Kazanımlar

- Dart dilinde sınıf, nesne, alan, kurucu metot, metot ve getter kavramlarını kullanma
- Bir modele ait veriyi ve davranışı aynı sınıfta toplama
- Model nesnelerini List içinde saklama ve ListView ile gösterme
- Kullanıcı işlemi sonucunda modeli değiştirip arayüzü setState ile yenileme
- Hesaplanan bir değeri getter ile her zaman güncel üretme

1. Ders — Öğrenci modelini oluşturma

Önce uygulamanın veri yapısını kuruyoruz. required anahtar sözcüğü, bir öğrenci oluştururken temel bilgilerin unutulmasını engeller. ortalama ve durum ayrıca saklanmaz; notlardan hesaplanır. Böylece yeni not eklendiğinde eski bir ortalama değerini güncelleme zorunluluğu doğmaz.

```
class Öğrenci {
  Öğrenci({
    required this.numara,
    required this.ad,
    required this.sinif,
    required this.notlar,
  });

  final int numara;
  final String ad;
  final String sınıf;
  final List<int> notlar;

  double get ortalama =>
    notlar.fold<int>(0, (toplama, not) => toplama + not) / notlar.length;

  String get durum => ortalama >= 85
    ? 'Çok iyi'
    : ortalama >= 70
      ? 'Başarılı'
      : ortalama >= 50
        ? 'Gelişiyor'
        : 'Destek gerekli';

  void notEkle(int not) => notlar.add(not.clamp(0, 100));
}
```

Burada:

- final alanlar nesnenin kimlik bilgilerini korur.

- notlar listesinin içine yeni değer eklenebilir.
- fold, listedeki notları toplayıp ortalamayı hesaplar.
- clamp(0, 100), eklenecek notu geçerli aralıkta tutar.

2. Ders — Nesne listesini hazırlama

Her satır aynı veri yapısını kullandığı için üç ayrı değişken grubu yerine üç Öğrenci nesnesi oluşturuyoruz.

```
final ogrenciler = [
  Öğrenci(
    numara: 1124,
    ad: 'Ece Yılmaz',
    sınıf: '11/B',
    notlar: [78, 92, 86],
  ),
  Öğrenci(
    numara: 1131,
    ad: 'Mert Kaya',
    sınıf: '11/B',
    notlar: [66, 74, 81],
  ),
  Öğrenci(
    numara: 1140,
    ad: 'Zeynep Ak',
    sınıf: '11/B',
    notlar: [91, 88, 95],
  ),
];
```

3. Ders — Listeyi ekrana bağlama

ListView.builder, listedeki her model için bir öğrenci kartı üretir. Kartın onTap olayı seçilen nesneyi secili değişkenine aktarır.

```
ListView.builder(
  shrinkWrap: true,
  itemCount: ogrenciler.length,
  itemBuilder: (context, index) {
    final ogrenci = ogrenciler[index];
    return ListTile(
      title: Text(ogrenci.ad),
      subtitle: Text('${ogrenci.sınıf} • ${ogrenci.notlar.length} değerlendirme'),
      trailing: Text(ogrenci.ortalama.toStringAsFixed(1)),
      onTap: () => setState(() => secili = ogrenci),
    );
  },
)
```

4. Ders — Modeli değiştirme ve sonucu yenileme

Seçilen öğrenciye 100 puan eklendiğinde notEkle metodu modeli değiştirir. setState, kartı yeniden oluşturur; ortalama ve durum getter üzerinden güncel notlarla tekrar hesaplanır.

```
FilledButton.icon(
  onPressed: () => setState(() => secili!.notEkle(100)),
  icon: const Icon(Icons.playlist_add_check),
  label: const Text('100 puan ekle'),
)
```

Ece Yılmaz'ın ilk ortalaması $(78 + 92 + 86) / 3 = 85,3$ olur. 100 puan eklendiğinde sonuç $(78 + 92 + 86 + 100) / 4 = 89,0$ olarak yenilenir.

Çalışan uygulamanın gerçek ekranı

Aşağıdaki görüntü, Flutter widget testi sırasında uygulama çalıştırılıp Ece Yılmaz seçildikten ve 100 puan eklendikten sonra alınmıştır.

11/B Öğrenci Paneli

Sınıf listesi

Bir öğrenciyi seçerek model ayrıntılarını görüntüle.

24	Ece Yılmaz 11/B • 4 değerlendirme	89.0
31	Mert Kaya 11/B • 3 değerlendirme	73.7
40	Zeynep Ak 11/B • 3 değerlendirme	91.3
Ece Yılmaz No: 1124 • 11/B Ortalama: 89.0 • Çok iyi  100 puan ekle		

Öğrenci Modeli gerçek Flutter ekranı

Özgün uygulama — Sınıf Kitaplığı

Şimdi bilgiyi farklı bir günlük yaşam problemine uyguluyoruz. Her kitap ISBN, ad, yazar, sayfa sayısı ve ödünç durumu taşır.

```

class Kitap {
  Kitap({
    required this.isbn,
    required this.ad,
    required this.yazar,
    required this.sayfa,
    this.odunc = false,
  });

  final String isbn;
  final String ad;
  final String yazar;
  final int sayfa;
  bool odunc;

  String get durum => odunc ? 'Ödünç verildi' : 'Rafta';
  void oduncDurumunuDegistir() => odunc = !odunc;
}

```

Ödünç verme düğmesi modeldeki odunc değerini değiştirir. Filtre açıkken görünür liste, yalnız odunc == false olan kitaplardan oluşturulur.

```

final gorunenKitaplar = sadeceRafta
  ? kitaplar.where((kitap) => !kitap.odunc).toList()
  : kitaplar;

```

Sınıf Kitaplığı

3 / 3 kitap rafta

Yalnız raftaki kitapları göster



Küçük Prens

Antoine de Saint-Exupéry
112 sayfa • Rafta

Ödünç ver



Şeker Portakalı

José Mauro de
Vasconcelos
182 sayfa • Rafta

Ödünç ver



İnsan Ne ile Yaşar?

Lev Tolstoy
96 sayfa • Rafta

Ödünç ver

Sınıf Kitaplığı gerçek Flutter ekranı

Uygulama görevleri

Kolay — Ders modeli

Ders adında bir sınıf oluşturun. Ders adı, haftalık saat ve öğretmen alanlarını tanımlayın. Üç ders nesnesini kartlar hâlinde gösterin.

Orta — Servis yolcu listesi

Yolcu modelinde ad, durak ve araca binip binmediğini tutun. Düğmeyle durumu değiştirin; araçtaki öğrenci sayısını ekranda gösterin.

İleri — Kantin stok sistemi

Urun modelinde ad, fiyat ve stok bilgisi tutun. Satış metodunda stok sıfırda işlem yapılmasını engelleyin. Toplam stok değeri ve satış tutarını hesaplayan getter'lar ekleyin.

Kontrol listesi

- Model sınıfı arayüz kodundan ayrı ve anlaşılır mı?
- Kurucu metotta zorunlu alanlar required olarak işaretlendi mi?
- Hesaplanan değerler gereksiz yere ayrıca saklanmak yerine getter ile mi üretiliyor?
- Modeldeki değişiklik setState içinde yapılarak ekrana yansıtılıyor mu?
- Liste boşken ve çok elemanlıyken görünüm test edildi mi?

Geri bildirim ve çözüm erişimi

Önce kendi uygulamanızı gönderin. Geri bildirimden sonra İpucunu göster bölümü yalnız yaklaşımı ve kontrol noktalarını açıklar; Örnek çözümü göster bölümü ise çalışan projenin gerçek lib/main.dart kodunu açar. Aynı bölümdeki indirme bağlantısından projeyi çevrim dışı çalıştırmak için tam kaynak paketini indirebilirsiniz.

Kaynak projeler

- Konu uygulaması: 05 - Modelleme ve Listeler/Uygulamalar/Uygulama-17-Öğrenci Modeli
- Özgün uygulama: 05 - Modelleme ve Listeler/Uygulamalar/Ozgun-17-Kitaplık Modeli