FLUTTER

Modelleme ve Listeler

17–20. haftalar

BT2025MOB1106 modülünün kazanımlarından hareketle; Android/Java ayrıntılarını kopyalamadan, öğrenci dostu öğrenme sırasıyla hazırlanmıştır.

9 birim • 36 hafta • 72 çalışan Flutter uygulaması • üç seviyeli uygulama görevleri

Öğrenme Haritası

Hafta	Ders	Uygulamalar
17	Sınıf ve Nesne Kullanımı	Öğrenci Modeli • Kitaplık Modeli
18	Dinamik Listeler	Alışveriş Listesi • Görev Takipçisi
19	Arama, Filtreleme ve Sıralama	Kişi Arama • Bütçe Hareketleri
20	Yeniden Kullanılabilir Widget	Puan Kartı • Ders Başarı Paneli

5. Birim — Modelleme ve Listeler

Bu birimde dört haftalık öğrenme döngüsü kullanılır: gör, birlikte çöz, değiştir, yeni probleme aktar.

17. Hafta — Sınıf ve Nesne Kullanımı

Dart sınıfları, kurucu metot ve model nesneleri

Bu derste konu, önce çalışan bir mobil arayüz üzerinde görülür; sonra widget ağacı küçük parçalara ayrılır. Kaynak kitaptaki kazanım korunur, ancak XML/Activity merkezli sıra yerine Flutter'ın widget, durum ve geri bildirim yaklaşımı kullanılır.

1	Dart sınıfları, kurucu metot ve model nesneleri
2	Girdi, işlem, çıktı ve durum değişimini ayırt etme
3	Okunabilir Dart kodu ve yeniden kullanılabilir widget geliştirme
4	Boş, sınır ve hatalı verilerle test etme

Çözümlü kitap uyarlaması: Öğrenci Modeli

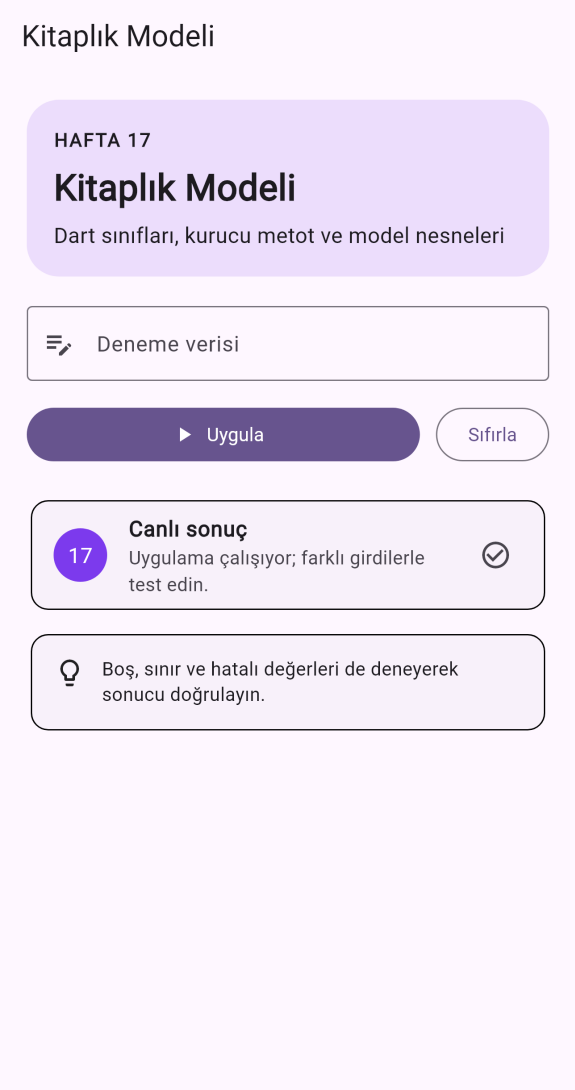
1. MaterialApp ve Scaffold ile ekran iskeletini kurun. 2. Veriyi TextField ile alın. 3. Değişen değeri State içinde tutun. 4. Olaydan sonra setState ile ekranı yenileyin. 5. Kullanıcıya açık sonuç ve hata mesajı gösterin.

```
FilledButton(onPressed: () => setState(() => value++), child: const Text('Uygula'));
```

Çalışan ekran — Öğrenci Modeli

Bu ekran Flutter'ın gerçek widget testinden alınmıştır; kaynak proje çevrimdışı pakette çalışır durumdadır.

Özgün günlük yaşam uygulaması — Kitaplık Modeli



Aynı teknik kazanım yeni bir problem bağlamında uygulanır. Öğrenci, örneği kopyalamak yerine veriyi ve ekran geri bildirimini kendi senaryosuna göre değiştirir.

Uygulama zamanı

Kolay	Başlık, renk, simge ve açıklamayı kendi senaryonuza uyarlayın.
Orta	Boş ve geçersiz girdiyi engelleyip Türkçe doğrulama mesajı gösterin.
İleri	Sonuçları bir modele dönüştürüp ListView içinde geçmiş olarak saklayın; silme ekleyin.

Geri bildirimden sonra kontrol: controller kaynakları dispose edildi mi, dar ekranda taşma var mı, boş/hata/yükleniyor durumları gösteriliyor mu ve en az üç test verisi denendi mi?

18. Hafta — Dinamik Listeler

Liste ekleme, silme, güncelleme ve dismissible

Bu derste konu, önce çalışan bir mobil arayüz üzerinde görülür; sonra widget ağacı küçük parçalara ayrılır. Kaynak kitaptaki kazanım korunur, ancak XML/Activity merkezli sıra yerine Flutter'ın widget, durum ve geri bildirim yaklaşımı kullanılır.

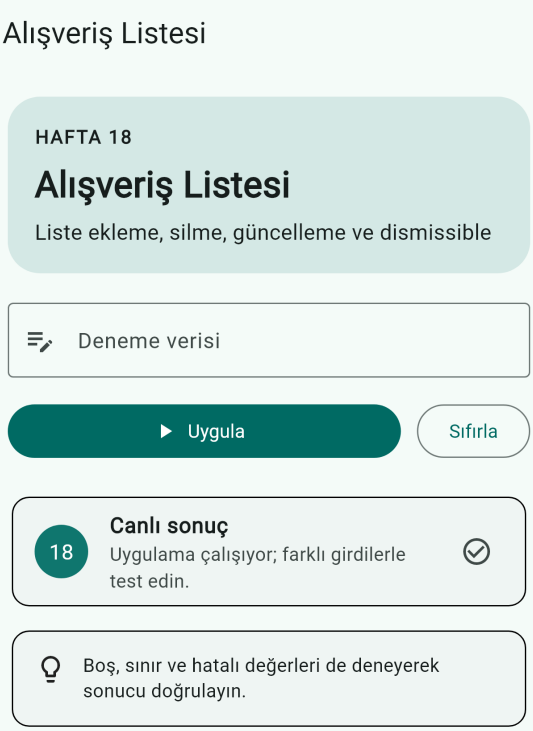
1	Liste ekleme, silme, güncelleme ve dismissible
2	Girdi, işlem, çıktı ve durum değişimini ayırt etme
3	Okunabilir Dart kodu ve yeniden kullanılabilir widget geliştirme
4	Boş, sınır ve hatalı verilerle test etme

Çözümlü kitap uyarlaması: Alışveriş Listesi

1. MaterialApp ve Scaffold ile ekran iskeletini kurun. 2. Veriyi TextField ile alın. 3. Değişen değeri State içinde tutun. 4. Olaydan sonra setState ile ekranı yenileyin. 5. Kullanıcıya açık sonuç ve hata mesajı gösterin.

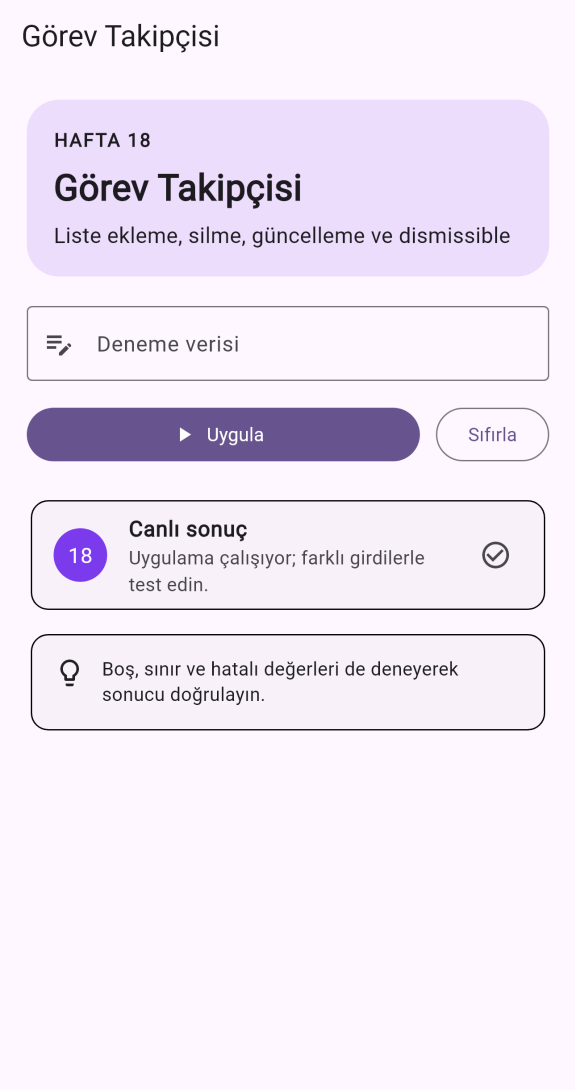
```
FilledButton(onPressed: () => setState(() => value++), child: const Text('Uygula'));
```

Çalışan ekran — Alışveriş Listesi



Bu ekran Flutter'ın gerçek widget testinden alınmıştır; kaynak proje çevrimdışı pakette çalışır durumdadır.

Özgün günlük yaşam uygulaması — Görev Takipçisi



Aynı teknik kazanım yeni bir problem bağlamında uygulanır. Öğrenci, örneği kopyalamak yerine veriyi ve ekran geri bildirimini kendi senaryosuna göre değiştirir.

Uygulama zamanı

Kolay	Başlık, renk, simge ve açıklamayı kendi senaryonuza uyarlayın.
Orta	Boş ve geçersiz girdiyi engelleyip Türkçe doğrulama mesajı gösterin.
İleri	Sonuçları bir modele dönüştürüp ListView içinde geçmiş olarak saklayın; silme ekleyin.

Geri bildirimden sonra kontrol: controller kaynakları dispose edildi mi, dar ekranda taşma var mı, boş/hata/yükleniyor durumları gösteriliyor mu ve en az üç test verisi denendi mi?

19. Hafta — Arama, Filtreleme ve Sıralama

where, sort ve kullanıcı sorgusuna göre liste

Bu derste konu, önce çalışan bir mobil arayüz üzerinde görülür; sonra widget ağacı küçük parçalara ayrılır. Kaynak kitaptaki kazanım korunur, ancak XML/Activity merkezli sıra yerine Flutter'ın widget, durum ve geri bildirim yaklaşımı kullanılır.

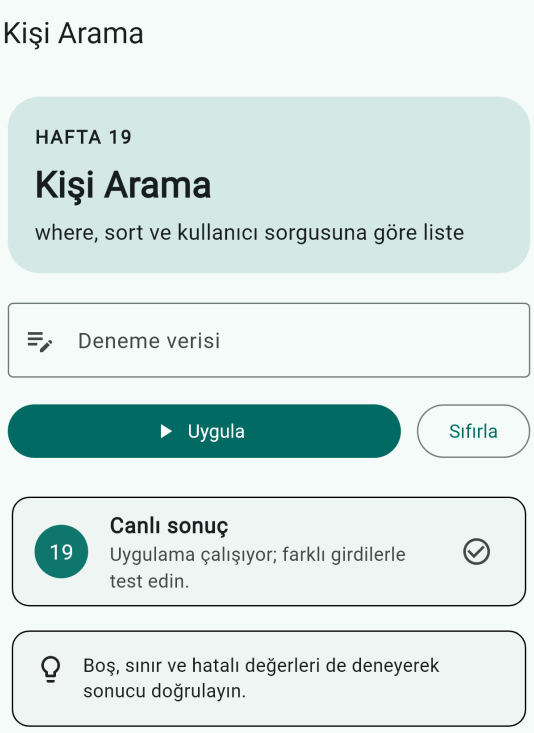
1	where, sort ve kullanıcı sorgusuna göre liste
2	Girdi, işlem, çıktı ve durum değişimini ayırt etme
3	Okunabilir Dart kodu ve yeniden kullanılabilir widget geliştirme
4	Boş, sınır ve hatalı verilerle test etme

Çözümlü kitap uyarlaması: Kişi Arama

1. MaterialApp ve Scaffold ile ekran iskeletini kurun. 2. Veriyi TextField ile alın. 3. Değişen değeri State içinde tutun. 4. Olaydan sonra setState ile ekranı yenileyin. 5. Kullanıcıya açık sonuç ve hata mesajı gösterin.

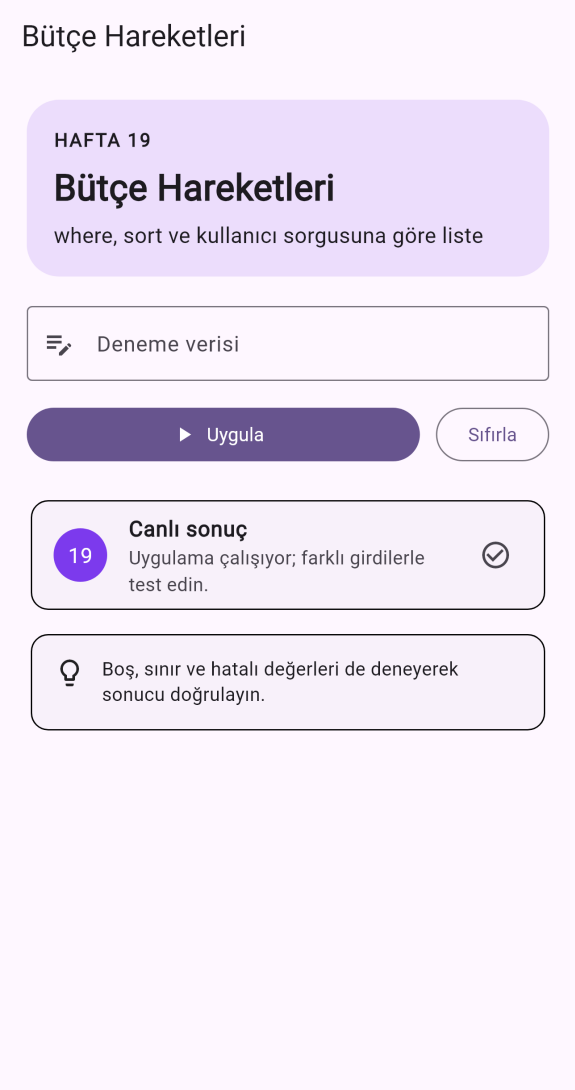
```
FilledButton(onPressed: () => setState(() => value++), child: const Text('Uygula'));
```

Çalışan ekran — Kişi Arama



Bu ekran Flutter'ın gerçek widget testinden alınmıştır; kaynak proje çevrimdışı pakette çalışır durumdadır.

Özgün günlük yaşam uygulaması — Bütçe Hareketleri



Aynı teknik kazanım yeni bir problem bağlamında uygulanır. Öğrenci, örneği kopyalamak yerine veriyi ve ekran geri bildirimini kendi senaryosuna göre değiştirir.

Uygulama zamanı

Kolay	Başlık, renk, simge ve açıklamayı kendi senaryonuza uyarlayın.
Orta	Boş ve geçersiz girdiyi engelleyip Türkçe doğrulama mesajı gösterin.
İleri	Sonuçları bir modele dönüştürüp ListView içinde geçmiş olarak saklayın; silme ekleyin.

Geri bildirimden sonra kontrol: controller kaynakları dispose edildi mi, dar ekranda taşma var mı, boş/hata/yükleniyor durumları gösteriliyor mu ve en az üç test verisi denendi mi?

20. Hafta — Yeniden Kullanılabilir Widget

Parametreli özel widget ve kodu bileşenlere ayırma

Bu derste konu, önce çalışan bir mobil arayüz üzerinde görülür; sonra widget ağacı küçük parçalara ayrılır. Kaynak kitaptaki kazanım korunur, ancak XML/Activity merkezli sıra yerine Flutter'ın widget, durum ve geri bildirim yaklaşımı kullanılır.

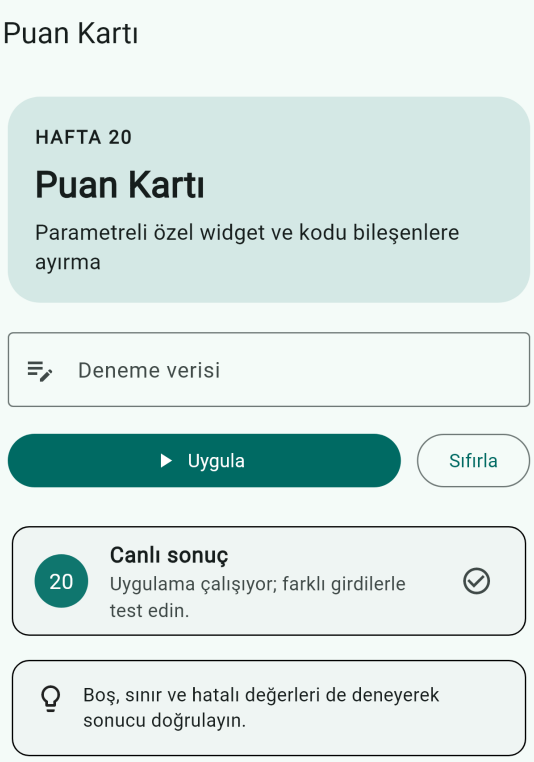
1	Parametreli özel widget ve kodu bileşenlere ayırma
2	Girdi, işlem, çıktı ve durum değişimini ayırt etme
3	Okunabilir Dart kodu ve yeniden kullanılabilir widget geliştirme
4	Boş, sınır ve hatalı verilerle test etme

Çözümlü kitap uyarlaması: Puan Kartı

1. MaterialApp ve Scaffold ile ekran iskeletini kurun. 2. Veriyi TextField ile alın. 3. Değişen değeri State içinde tutun. 4. Olaydan sonra setState ile ekranı yenileyin. 5. Kullanıcıya açık sonuç ve hata mesajı gösterin.

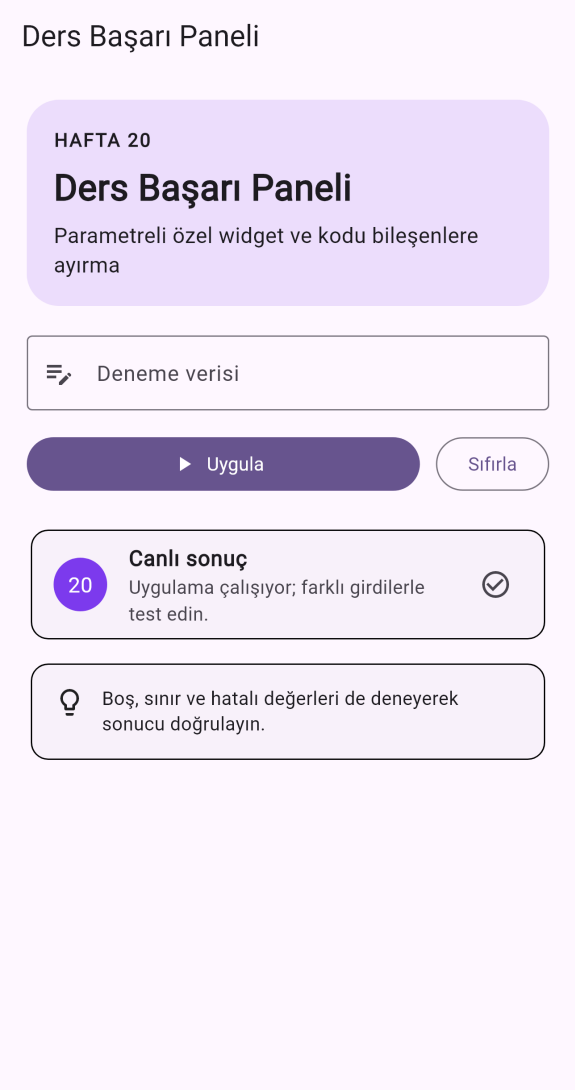
```
FilledButton(onPressed: () => setState(() => value++), child: const Text('Uygula'));
```

Çalışan ekran — Puan Kartı



Bu ekran Flutter'ın gerçek widget testinden alınmıştır; kaynak proje çevrimdışı pakette çalışır durumdadır.

Özgün günlük yaşam uygulaması — Ders Başarı Paneli



Aynı teknik kazanım yeni bir problem bağlamında uygulanır. Öğrenci, örneği kopyalamak yerine veriyi ve ekran geri bildirimini kendi senaryosuna göre değiştirir.

Uygulama zamanı

Kolay	Başlık, renk, simge ve açıklamayı kendi senaryonuza uyarlayın.
Orta	Boş ve geçersiz girdiyi engelleyip Türkçe doğrulama mesajı gösterin.
İleri	Sonuçları bir modele dönüştürüp ListView içinde geçmiş olarak saklayın; silme ekleyin.

Geri bildirimden sonra kontrol: controller kaynakları dispose edildi mi, dar ekranda taşma var mı, boş/hata/yükleniyor durumları gösteriliyor mu ve en az üç test verisi denendi mi?