Statefull Widget

A) Hazırlık:

1. İlk olarak, Flutter projesini oluşturarak işe başlayalım: Bunun için hem **VS Code'u** hem de **Android Studio Emulator'u** açın. Yeni bir Flutter projesi oluşturun ve projeye bir isim verin (örneğin: **clock_app**). lib/main.dart dosyasını açın.
2. Öncelikle uygulamanın ana yapısını başlatan ilk adımı oluşturun. **void main()** fonksiyonu programınızın başlama noktasıdır. Bu fonksiyon çalıştığında Flutter uygulamanız başlar. **runApp(const MyApp())** komutu, Flutter uygulamasını başlatır ve tüm widget ağacını ekrana çizer.

```
void main() {
  runApp(const MyApp());
}
```

3. Bu adımda, uygulamamızın ana yapısını oluşturacak olan **MyApp** sınıfını yazacağız. Bu sınıf, uygulamanın giriş noktası olan **widget**'ı temsil eder.

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: ClockWidget(),
    );
  }
}
```

Burada MyApp adında bir sınıf tanımlar ve bu sınıf, **StatelessWidget** sınıfını miras alır. **StatelessWidget d**urumu olmayan, sabit bir kullanıcı arayüzü oluşturan bir widget'tır. **super.key** ifadesi, widget'ın anahtarını üst sınıfa

(StatelessWidget) geçirir. **build** metodu, widget'ı ekranda nasıl göstereceğimizi tanımlar. **context** parametresi, widget'ın içinde bulunduğu ortamı temsil eder ve bu sayede uygulamanın widget ağacına erişim sağlar. **MaterialApp**: Uygulamamızın ana yapısını oluşturan widget'tır. Material tasarım prensiplerini kullanarak uygulamayı oluşturur.

4. Bu adımda, saati gösterecek olan ana widget'ımız olan ClockWidget sınıfını oluşturacağız. Bu widget, saati güncellemek ve kullanıcıya göstermek için StatefulWidget (durumlu widget) yapısını kullanacak. Bu adımda sadece sınıf yapısını kuracağız.

```
class ClockWidget extends StatefulWidget {
  const ClockWidget({super.key});

  @override
  State<ClockWidget> createState() => _ClockWidgetState();
}

class _ClockWidgetState extends State<ClockWidget> {

  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

Uygulamada zaman gibi değişken bilgileri göstermek istediğimizde **StatefulWidget** kullanırız. **StatefulWidget**: Bu widget, durum bilgisine sahip bir widget'tır. Yani, bir butona tıkladığınızda ya da bir değişiklik olduğunda bu widget kendini güncelleyebilir. Burada **_ClockWidgetState** sınıfı, widget'ın durumu ile ilgilenecektir. Bu sınıfı oluşturun ve build() metodunda şimdilik boş bir **Container** döndürüyor.

5. **Scaffold** widget'ını kullanarak uygulamanın temel iskeletini oluşturun. **AppBar** ekleyerek, uygulamanıza bir başlık alanı (title) ekleyin. Başlığı "Saat Uygulaması" olarak ayarlayın. **AppBar**'ın başlık metnini ortalayın ve arka plan rengini **deepOrange** yapın.

```
return Container(
  child: Scaffold(
    appBar: AppBar(
      title: const Text("Saat Uygulaması"),
      centerTitle: true,
      backgroundColor: Colors.deepOrange,
    ),
```

6. Bu adımda, saat uygulamamızın arayüzüne bir buton ve metin ekleyeceğiz. Bunun için Scaffold içindeki **body** alanına bir **Center** widget ekleyin. **Center** widget'ının içine bir **Column** widget'ı yerleştirin. **Column** widget'ı içinde bir **Text** widget ekleyerek saatin gösterileceği alanı oluşturun. Şimdilik "Saat buraya yazılacak" şeklinde bir metin ekleyin. **ElevatedButton** widget'ı ile butonu ekleyin. Butonun üzerindeki yazı "Saat Kaç?" olacak. **onPressed** özelliği şimdilik **null** olarak bırakılacak.

```
    backgroundColor: Colors.deepOrange,
  ),
  body: Center(
    child: Column(
      children: [
        Text("Saat buraya yazılacak"),
        ElevatedButton(
            onPressed: null, child: const Text("Saat Kaç?")),
      ],
    ),
```

7. Biçimsel olarak düzenlemeye eklediğimiz metin ve butonu hizalama ile devam edelim. **`mainAxisAlignment`**, Flutter'da bir **Column** veya **Row** widget'ı içindeki çocukların (alt widget'ların) ana eksen (main axis) boyunca hizalanmasını ayarlamak için kullanılan bir özelliktir. **Column** widget'ında ana eksen dikey eksendir.

```
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Text("Saat buraya yazılacak"),
    ElevatedButton(
```

8. Bu adımda, uygulamanızda saati göstermek için kullanacağımız değişkenleri tanımlayacağız. Bunun için **_ClockWidgetState** sınıfının içine üç değişken ekleyin: **saat**, **dakika**, ve **saniye**. Her bir değişken, **DateTime.now()** kullanılarak o anki saat, dakika ve saniye bilgilerini tutacak.

```
class _ClockWidgetState extends State<ClockWidget> {
  var saat = DateTime.now().hour;
  var dakika = DateTime.now().minute;
  var saniye = DateTime.now().second;

  @override
```

9. Bu adımda, önceki adımda tanımladığımız **saat**, **dakika** ve **saniye** değişkenlerini ekrana yazdırmak için **Text** widget'ını kullanacağız. Böylece, uygulamanın o anki saat bilgisini ekranda görmüş olacağız.

```
    Text("$saat:$dakika:$saniye"),
    ElevatedButton(
```

10. Elevatedbutton'null olarak kullanmıştık, şimdi onu işlevsel hale getirelim. Öncelikle bu butonu işlevsel hale getirmek için **null** ifadesini normal fonksiyon haline getirelim.

```
ElevatedButton(onPressed: () {},
```

11. Bu adımda, butona tıklandığında saati güncelleyip ekranda gösteren bir işlevi nasıl ekleyeceğimizi öğreneceğiz. **onPressed** içerisine bir **setState()** fonksiyonu ekleyin ve saati, dakikayı ve saniyeyi güncelleyin. Bu işlem, butona her tıkladığınızda o anki saatin ekranda gösterilmesini sağlar.

```
ElevatedButton(
  onPressed: () {
    setState(() {
      saat = DateTime.now().hour;
      dakika = DateTime.now().minute;
      saniye = DateTime.now().second;
    });
```

12. Bu adımda, saati güncelleyen bir asenkron fonksiyon olan **fetchTime()**'ı öğreneceğiz. Bu fonksiyon, saat bilgilerini güncelleyip uygulamamızda kullanmamızı sağlayacak.

`Future<void> fetchTime() async` ifadesi ile bir fonksiyon tanımlayın. Bu fonksiyon, saat bilgilerini güncelleyecek. `setState()` fonksiyonunun içine, saati, dakikayı ve saniyeyi güncelleyen kodları yazın. Bu işlem, güncel saat bilgilerini alarak değişkenlere atayacaktır. Burada **fetchTime()** ile Asenkron bir fonksiyon tanımlıyoruz. Bu fonksiyon, güncel saat bilgisini almak için kullanılır. **Future**: Asenkron (eşzamansız) programlama kavramıdır. Bir `Future`, gelecekte bir zamanda tamamlanacak bir işlemi temsil eder. Sonuç olarak, `Future<void>` kullanarak, asenkron bir işlemi başlatıyoruz ama bu işlemin sonucunu kullanmak istemiyoruz. Bu, özellikle durumu güncellemek veya ekrandaki içerikleri değiştirmek için faydalıdır.

```
var saniye = DateTime.now().second;

Future<void> fetchTime() async {
  setState(() {
    saat = DateTime.now().hour;
    dakika = DateTime.now().minute;
    saniye = DateTime.now().second;
  });
```

13. Bu adımda düğmeye tıklandığında `fetchTime` fonksiyonu çalışacak ve güncel saat bilgisi ekranda gösterilecektir. Bu, uygulamanızın etkileşimli olmasını sağlamak için önemli bir adımdır.

```
Text("$saat:$dakika:$saniye"),
                ElevatedButton(
                    onPressed: fetchTime, child: const
```

14. Bu adımda, `showTime` adında bir değişken tanımlayacağız. Bu değişken, saat bilgisinin ekranda görünür olup olmadığını kontrol etmek için kullanılacak. Aşağıdaki kodu `_ClockWidgetState` sınıfı içine ekleyin:

```
bool showTime = false;
```

15. **Ekranda Saat Bilgisi Gösterme:** Kod, `showTime` değişkeninin değerine göre saat bilgisinin gösterilmesini sağlar. Eğer `showTime true` ise, saat bilgisi ekranda gösterilecektir. Aksi halde, hiçbir şey gösterilmeyecektir.

```
            children: [
              if (showTime == true)
Text("$saat:$dakika:$saniye"),
```

16. **Değişkenin Güncellenmesi**: `fetchTime` fonksiyonunun içinde, saat bilgisi güncellenirken `showTime` değişkeni de `true` olarak ayarlanmalıdır.

```
setState(() {
  showTime = true; // Saat gösterilsin
  saat = DateTime.now().hour;
  dakika = DateTime.now().minute;
  saniye = DateTime.now().second;
});
```

17. **Gizleme İşlemi**: Saat bilgisinin 5 saniye sonra gizlenmesi için `Timer` kullanılarak `showTime` değeri `false` olarak ayarlanmalıdır. Bu işlem, saat bilgisi ekranda belirli bir süre kaldıktan sonra otomatik olarak gizlenmesini sağlar.

```
    saniye = DateTime.now().second;
  });
  Timer(const Duration(seconds: 5), () {
    setState(() {
      showTime = false;
    });
  });
```

18. Son adımda, Flutter uygulamanızda kullanıcıya kısa bir mesaj göstermek için kullanılan `SnackBar` widget'ını inceleyeceğiz. Aşağıdaki kod, bir kullanıcı düğmeye bastığında ekranda bir mesaj göstermektedir. **SnackBar**, kullanıcıya geçici bir bildirim gösteren bir widget'tır. Genellikle, belirli bir eylem gerçekleştirildiğinde kullanıcıya bilgi vermek amacıyla kullanılır.

```
  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
      content: Text('Saat bulunuyor..'),
    ),
  );
  Timer(const Duration(seconds: 5), () {
```

`ScaffoldMessenger.of(context)`: Bu ifade, mevcut widget ağacındaki doğru `ScaffoldMessenger` nesnesini bulmak için kullanılır.

`showSnackBar(...)`: `showSnackBar` metodu, belirtilen `SnackBar`'ı ekranda göstermek için kullanılır. Bu metodu çağırarak belirli bir mesajı kullanıcıya iletmiş olursunuz.