

1- Aşağıdaki soruları cevaplayınız

- **1- Cross platform nedir?**

Cross-platform, bir yazılımın tek bir kod tabanıyla birden fazla işletim sisteminde çalıştırılabilmesini sağlar. Bu yöntem, geliştiricilerin uygulamayı sadece bir kez yazıp, farklı platformlarda kullanılabilir hale getirmesine olanak tanır. Uygulama, işletim sistemi ile bir köprü veya SDK (Software Development Kit) aracılığıyla derlenir ve çalıştırılır. Bu teknolojiye örnek olarak React Native, Flutter, ve Xamarin gibi framework'ler gösterilebilir.

- **2- Native platform nedir?**

Native platform, uygulamaların işletim sistemi tarafından doğrudan derlenebildiği geliştirme yöntemidir. Her platformun kendine özgü programlama dilleri kullanılarak geliştirilir. Örneğin, Android için Android Studio, Eclipse, veya IntelliJ IDEA IDE'leri kullanılarak Java veya Kotlin dilleriyle uygulama geliştirilir. iOS için ise Xcode kullanılarak Objective-C veya Swift dilleriyle mobil uygulamalar geliştirilebilir. Native uygulamalar, platforma özgü en iyi performansı ve kullanıcı deneyimini sunar.

- **3- Üsttekine alternatifler Ionic, Cordoba... nedir ve neden ihtiyaç duyulur? alternatif midir?**

Ionic ve Apache Cordova, hibrit mobil uygulama geliştirme çerçeveleridir. Bu çerçeveler, HTML, CSS, ve JavaScript gibi web teknolojilerini kullanarak mobil uygulama geliştirilmesine olanak tanır. Geliştirilen bu uygulamalar, bir native kabuğa sarılarak Android ve iOS gibi platformlarda çalıştırılır. Yani web teknolojileriyle yazılan uygulamalar, mobil uygulamalara dönüştürülmüş olur. Bu sayede, geliştiriciler tek bir kod tabanıyla birden fazla platforma uygulama yayımlayabilirler.

Apache Cordova, web teknolojileriyle yazılan uygulamaların native bileşenlerle entegre edilmesi için kullanılır. Cordova, cihazın kamera, GPS gibi donanımlarına erişimi sağlayarak, geliştiricilerin web teknolojileriyle oluşturduğu uygulamaların mobil cihazlarla uyumlu şekilde çalışmasını sağlar. Cordova, web uygulamalarını mobil platformlara uyarlamak için bir köprü görevi görür.

Ionic, Cordova'nın üzerine inşa edilmiş olup, geliştiricilere daha modern bir arayüz ve kullanıcı deneyimi sağlar. Ionic, çeşitli kullanıcı arayüzü bileşenleri ve temalar sunarak, mobil uygulama geliştirme sürecini hızlandırır. Ionic ile yapılan uygulamalar, hem estetik açıdan daha güçlü bir arayüze sahip olur hem de kullanıcı dostu bir deneyim sunar.

Bu çerçevelere ihtiyaç duyulmasının temel nedeni, bir kez yazılan kodun birden fazla platformda çalıştırılabilmesidir. Geliştiriciler, native uygulama geliştirme sürecinde her platform için ayrı ayrı kod yazmak yerine, hibrit çözümlerle iş yükünü azaltabilirler. Bu açıdan Ionic ve Cordova, React Native, Flutter gibi diğer cross-platform çözümlerine birer alternatif olarak öne çıkar.

- **4- Arayüz geliştirme hakkında bilgi veriniz**

Arayüz geliştirme, bir mobil veya web uygulamasının kullanıcı ile etkileşim kurduğu kısmı oluşturma sürecidir. Kullanıcı arayüzü (UI), görsel tasarım unsurlarını içerir ve kullanıcının uygulamayla nasıl etkileşimde bulunduğunu belirler. İyi bir arayüz, kullanıcı deneyimini (UX) olumlu yönde etkileyen bir yapıya sahip olmalıdır.

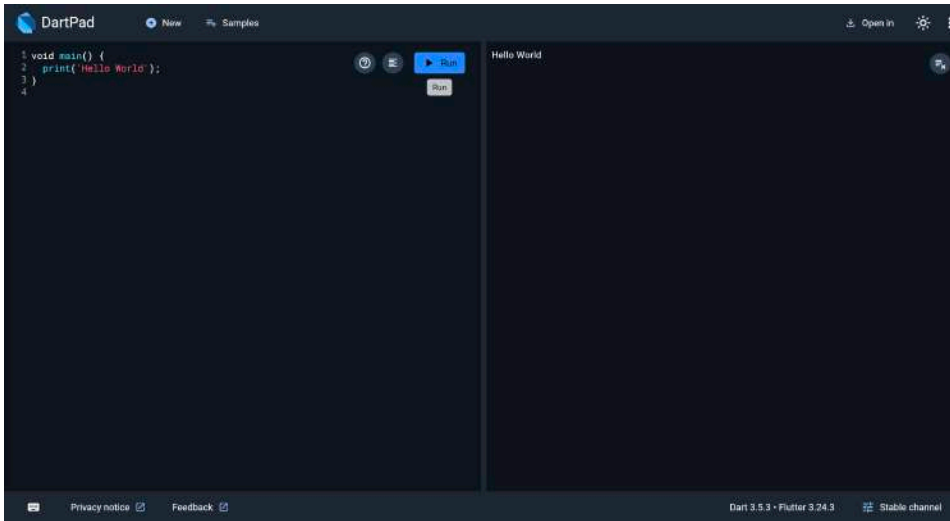
Mobil uygulama arayüzü geliştirme sürecinde:

- **Wireframe:** Uygulamanın temel yapısını ve tasarımını oluşturur.
- **Mockup:** Wireframe üzerinde daha detaylı, renkler ve görseller içeren arayüz tasarımıdır.
- **Prototip:** Uygulamanın interaktif versiyonu olup, arayüz bileşenlerinin nasıl çalışacağını gösterir.

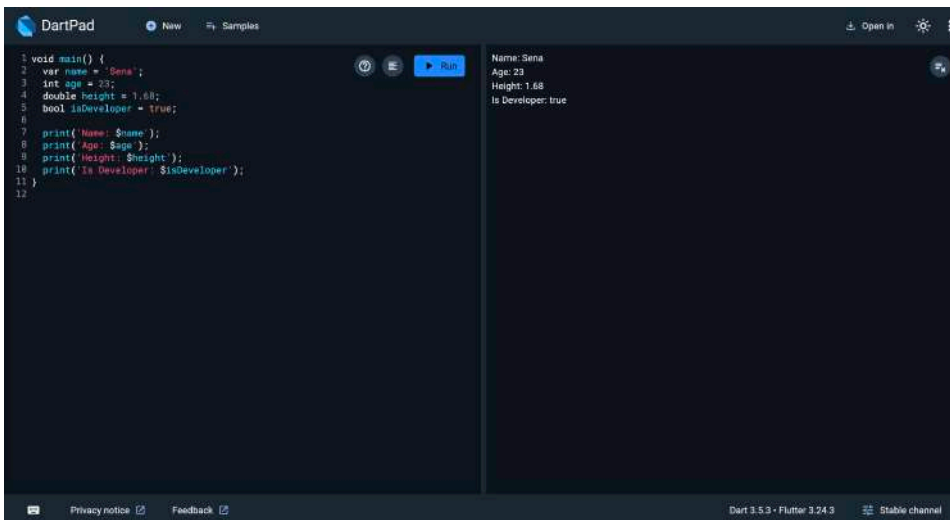
Geliştirme sırasında arayüz elementleri tasarlanır ve kullanıcının deneyimsel bir şekilde uygulamayı kullanmasını sağlar. React Native ve Flutter gibi framework'ler ile arayüz geliştirme yapılabilir.

2- Aşağıdaki soruları cevaplayınız

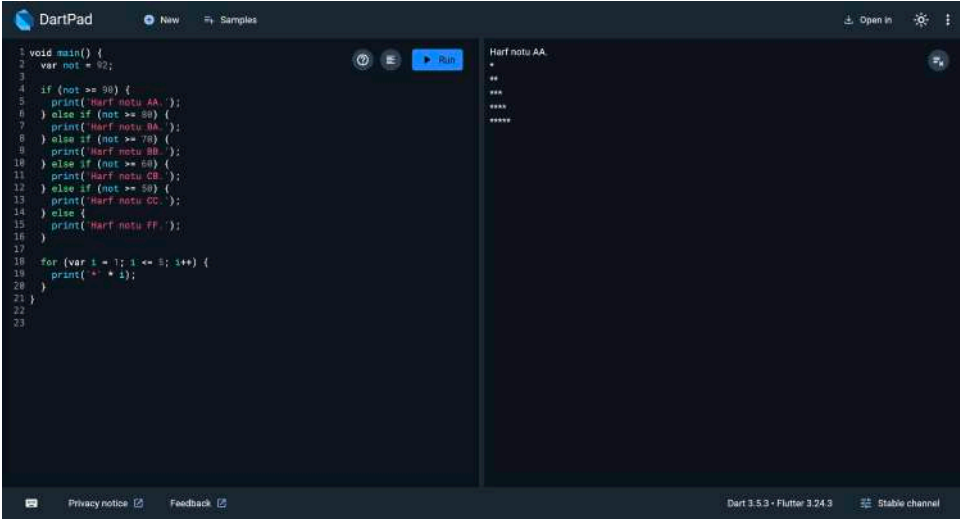
- Hello World



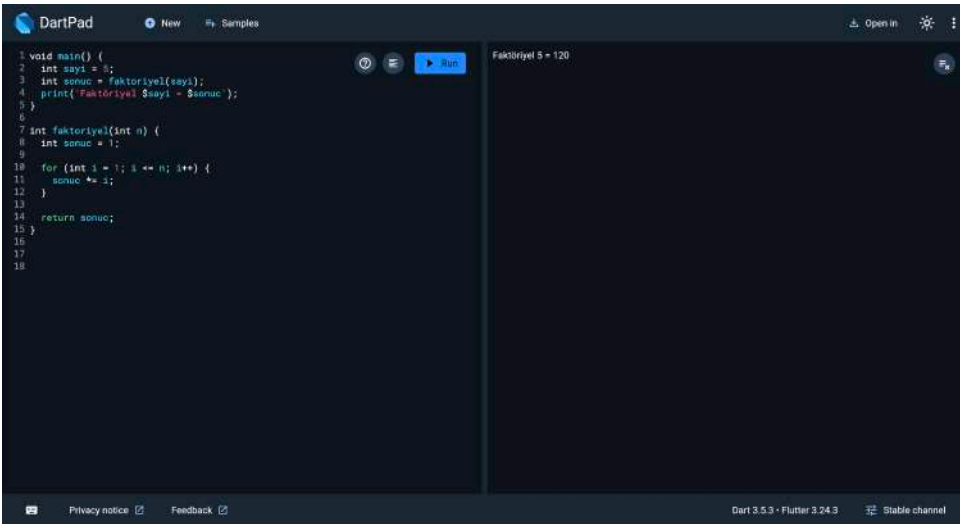
- Variables



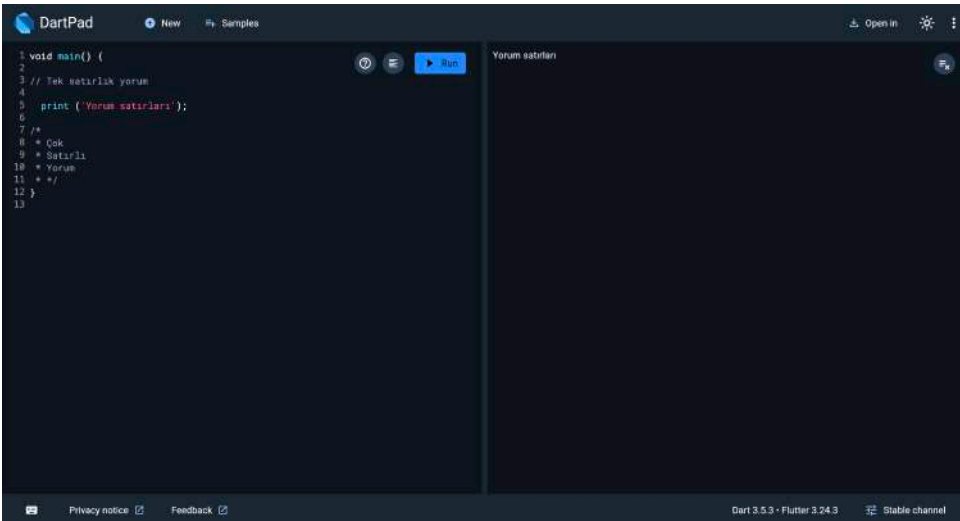
- Control flow statements



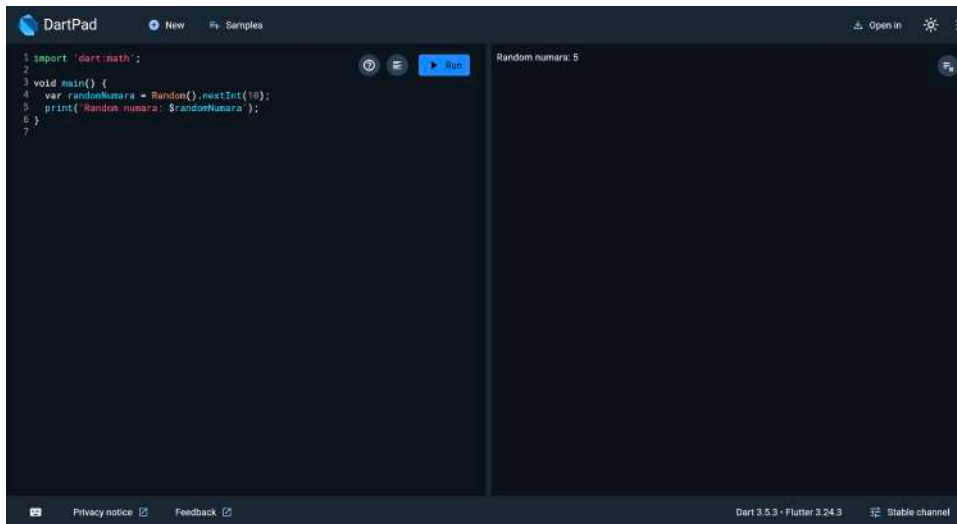
- Functions



- Comments



- Imports



The screenshot shows the DartPad interface. The code editor on the left contains the following Dart code:

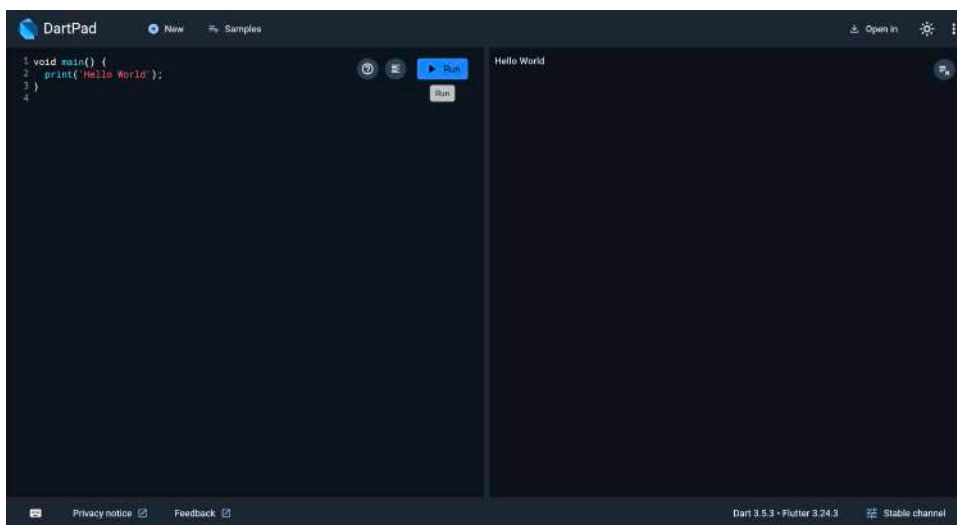
```
1 import 'dart:math';
2
3 void main() {
4   var randomNumara = Random().nextInt(10);
5   print('Random numara: $randomNumara');
6 }
7
```

The output console on the right displays the result of the program execution:

```
Random numara: 5
```

The bottom status bar indicates the environment: Dart 3.5.3 • Flutter 3.24.3 • Stable channel.

- Classes



The screenshot shows the DartPad interface. The code editor on the left contains the following Dart code:

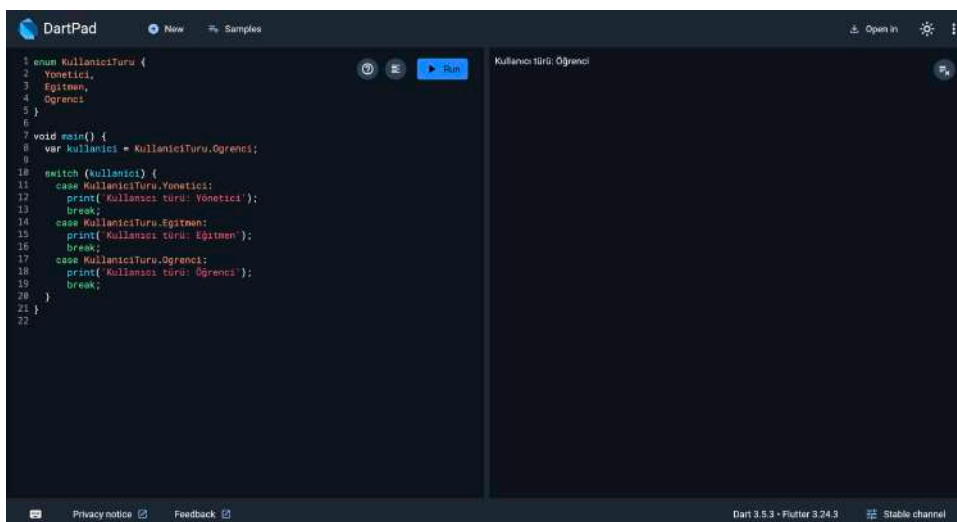
```
1 void main() {
2   print('Hello World');
3 }
4
```

The output console on the right displays the result of the program execution:

```
Hello World
```

The bottom status bar indicates the environment: Dart 3.5.3 • Flutter 3.24.3 • Stable channel.

- Enums



The screenshot shows the DartPad interface. The code editor on the left contains the following Dart code:

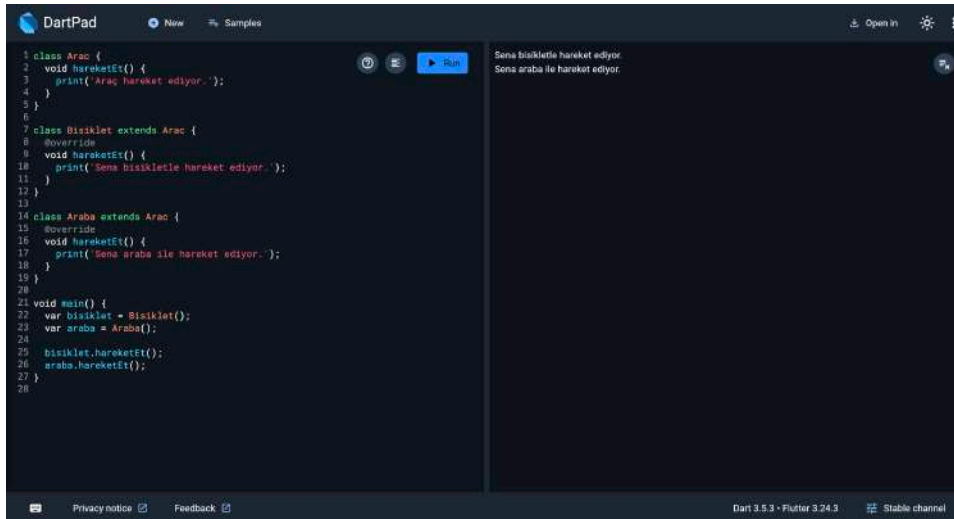
```
1 enum KullanciTuru {
2   Yoneticici,
3   Egitmen,
4   Ogrenci
5 }
6
7 void main() {
8   var kullanci = KullanciTuru.Ogrenci;
9
10  switch (kullanci) {
11    case KullanciTuru.Yoneticici:
12      print('Kullanci turu: Yoneticici');
13      break;
14    case KullanciTuru.Egitmen:
15      print('Kullanci turu: Egitmen');
16      break;
17    case KullanciTuru.Ogrenci:
18      print('Kullanci turu: Ogrenci');
19      break;
20  }
21 }
22
```

The output console on the right displays the result of the program execution:

```
Kullanci türü: Öğrenci
```

The bottom status bar indicates the environment: Dart 3.5.3 • Flutter 3.24.3 • Stable channel.

- Inheritance



The screenshot shows the DartPad interface with a code editor on the left and a console on the right. The code defines three classes: `Arac`, `Bisiklet`, and `Araba`. `Arac` has a `hareketEt()` method that prints "Araç hareket ediyor.". `Bisiklet` extends `Arac` and overrides `hareketEt()` to print "Sena bisikletle hareket ediyor.". `Araba` extends `Arac` and overrides `hareketEt()` to print "Sena araba ile hareket ediyor.". The `main` function creates instances of `Bisiklet` and `Araba` and calls their `hareketEt()` methods. The console output shows the results of these calls.

```
1 class Arac {
2   void hareketEt() {
3     print('Araç hareket ediyor.');
```

```
7 class Bisiklet extends Arac {
8   @override
9   void hareketEt() {
10    print('Sena bisikletle hareket ediyor.');
```

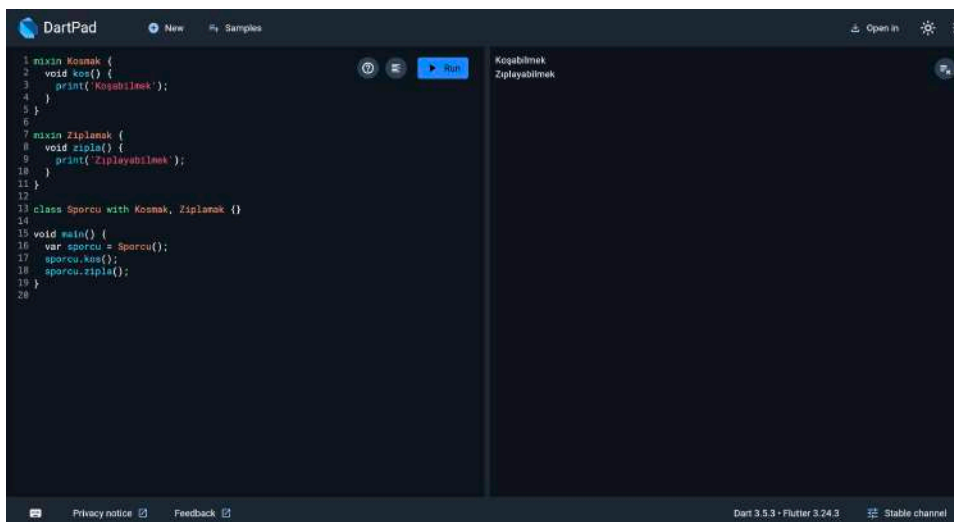
```
14 class Araba extends Arac {
15   @override
16   void hareketEt() {
17    print('Sena araba ile hareket ediyor.');
```

```
21 void main() {
22   var bisiklet = Bisiklet();
23   var araba = Araba();
24
25   bisiklet.hareketEt();
26   araba.hareketEt();
27 }
28
```

Sena bisikletle hareket ediyor.
Sena araba ile hareket ediyor.

Dart 3.5.3 • Flutter 3.24.3 Stable channel

- Mixins



The screenshot shows the DartPad interface with a code editor on the left and a console on the right. The code defines two mixins: `Kosmak` and `Ziplamak`. `Kosmak` has a `kos()` method that prints "Koşabiliyorum.". `Ziplamak` has a `zipla()` method that prints "Zıplayabiliyorum.". `Sporcu` is a class that uses both mixins with `with` and implements both `kos()` and `zipla()` methods. The `main` function creates a `Sporcu` instance and calls both `kos()` and `zipla()` methods. The console output shows the results of these calls.

```
1 mixin Kosmak {
2   void kos() {
3     print('Koşabiliyorum.');
```

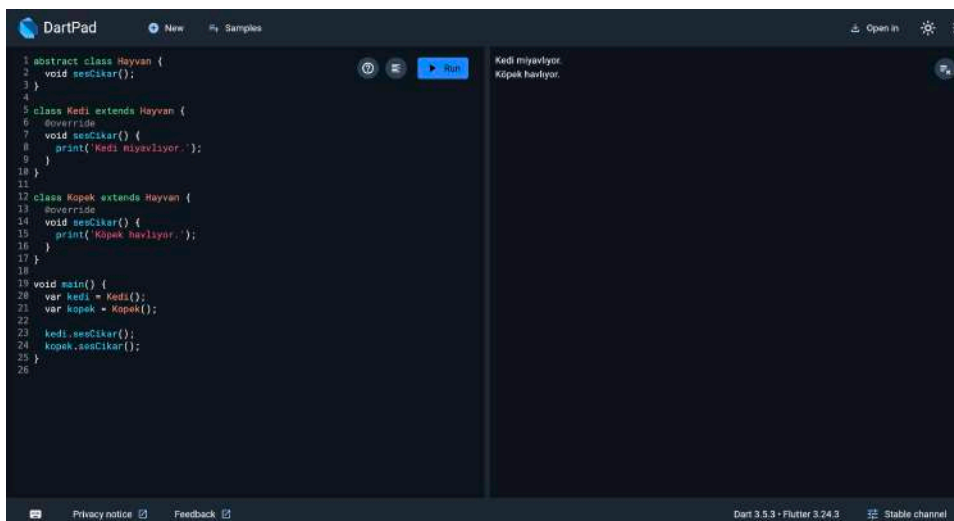
```
7 mixin Ziplamak {
8   void zipla() {
9     print('Zıplayabiliyorum.');
```

```
13 class Sporcu with Kosmak, Ziplamak {}
14
15 void main() {
16   var sporcu = Sporcu();
17   sporcu.kos();
18   sporcu.zipla();
19 }
20
```

Koşabiliyorum.
Zıplayabiliyorum.

Dart 3.5.3 • Flutter 3.24.3 Stable channel

- Interfaces and abstract classes



The screenshot shows the DartPad interface with a code editor on the left and a console on the right. The code defines an abstract class `Hayvan` with an abstract method `sesCikar()`. `Kedi` and `Kopek` are classes that extend `Hayvan` and override `sesCikar()` to print "Kedi miyavlıyor." and "Köpek havlıyor." respectively. The `main` function creates instances of `Kedi` and `Kopek` and calls their `sesCikar()` methods. The console output shows the results of these calls.

```
1 abstract class Hayvan {
2   void sesCikar();
3 }
4
5 class Kedi extends Hayvan {
6   @override
7   void sesCikar() {
8     print('Kedi miyavlıyor.');
```

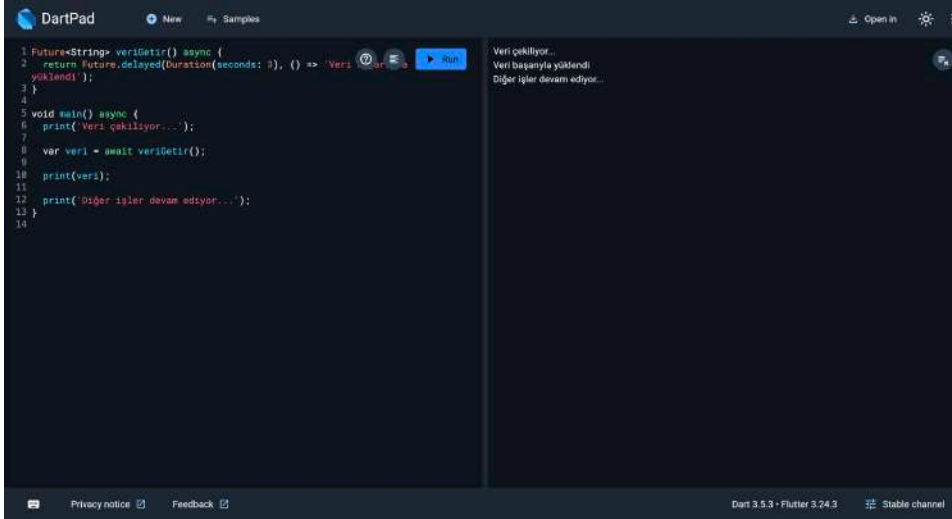
```
12 class Kopek extends Hayvan {
13   @override
14   void sesCikar() {
15     print('Köpek havlıyor.');
```

```
18 void main() {
19   var kedi = Kedi();
20   var kopek = Kopek();
21
22   kedi.sesCikar();
23   kopek.sesCikar();
24 }
25
```

Kedi miyavlıyor.
Köpek havlıyor.

Dart 3.5.3 • Flutter 3.24.3 Stable channel

- Async



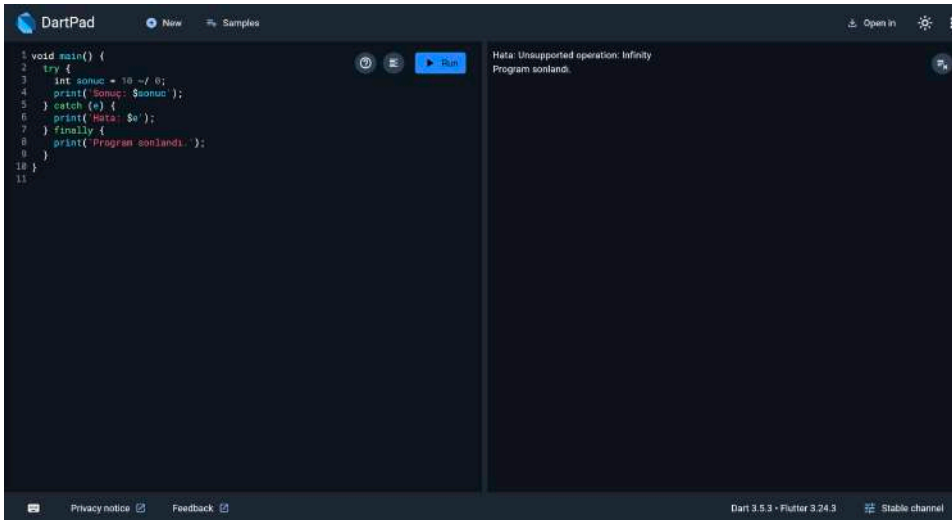
The screenshot shows the DartPad interface with a code editor on the left and a console on the right. The code defines an asynchronous function `veriGetir()` that returns a `Future<String>` after a 2-second delay. The `main()` function calls `veriGetir()` and prints the result. The console output shows the sequence of events: 'Veri çekiliyor...' (Data is being fetched...), 'Veri başarıyla yüklendi' (Data loaded successfully), and 'Diğer işler devam ediyor...' (Other tasks continue...).

```
1 Future<String> veriGetir() async {
2   return Future.delayed(Duration(seconds: 2), () => 'Veri çekiliyor...');
3 }
4
5 void main() async {
6   print('Veri çekiliyor...');
7   var veri = await veriGetir();
8   print(veri);
9   print('Diğer işler devam ediyor...');
10 }
11
12
13
14
```

Veri çekiliyor...
Veri başarıyla yüklendi
Diğer işler devam ediyor...

Dart 3.5.3 • Flutter 3.24.3 Stable channel

- Exceptions



The screenshot shows the DartPad interface with a code editor on the left and a console on the right. The code defines a `main()` function that uses a `try-catch-finally` block to handle a division by zero error. The console output shows the error message: 'Hata: Unsupported operation: Infinity' and 'Program sonlandı.' (Program ended).

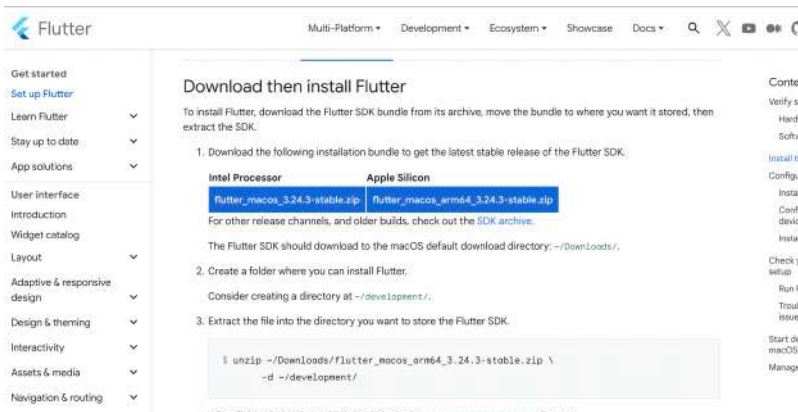
```
1 void main() {
2   try {
3     int sonuc = 10 ~/ 0;
4     print('Sonuç: $sonuc');
5   } catch (e) {
6     print('Hata: $e');
7   } finally {
8     print('Program sonlandı. ');
9   }
10 }
11
```

Hata: Unsupported operation: Infinity
Program sonlandı.

Dart 3.5.3 • Flutter 3.24.3 Stable channel

3- Kurulumlar

Geçen hafta 1. haftadaki ödevi React Native kurulumu olarak mailden iletmiştim, ancak Flutter ile devam etmeye karar verdiğim için kurulum kısmını güncelleyerek tekrar iletiyorum. Xcode (simülator için) ve VSCode bilgisayarımda zaten kurulu. Öncelikle Flutter'ı bilgisayarıma kurarak başladım.



```
senagurkan@senas-mac-3 ~ % mkdir development
```

```
senagurkan@senas-mac-3 ~ % unzip ~/Downloads/flutter_macos_arm64_3.24.3-stable.zip -d ~/development/
```

```
senagurkan@senas-mac-3 ~ % cd development
```

```
senagurkan@senas-mac-3 ~ % open .zshrc
```

```
.zshrc - Edited
export PATH="/opt/homebrew/bin:$PATH"

# Herd injected NVM configuration
export NVM_DIR="/Users/senagurkan/Library/Application Support/Herd/config/nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm

[[ -f "/Applications/Herd.app/Contents/Resources/config/shell/zshrc.zsh" ]] && builtin source "/Applications/Herd.app/Contents/Resources/config/shell/zshrc.zsh"

# Herd injected PHP 8.3 configuration
export HERD_PHP_83_INI_SCAN_DIR="/Users/senagurkan/Library/Application Support/Herd/config/php/83/"

# Herd injected PHP binary
export PATH="/Users/senagurkan/Library/Application Support/Herd/bin/"$PATH
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm
bash_completion
ulimit -n 10240

export PATH=$HOME/development/flutter/bin:$PATH
```

```
senagurkan@senas-mac-3 ~ % flutter doctor
Last login: Thu Oct 20 02:02:15 on ttys005
senagurkan@senas-mac-3 ~ % flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.24.3, on macOS 15.0.1 24A348 darwin-arm64, locale en-TR)
[✓] Android toolchain - develop for Android devices (Android SDK version 35.0.0)
[✓] Xcode - develop for iOS and macOS (Xcode 16.0)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2024.1)
[✓] VS Code (version 1.94.0)
[✓] Connected device (3 available)
[✓] Network resources

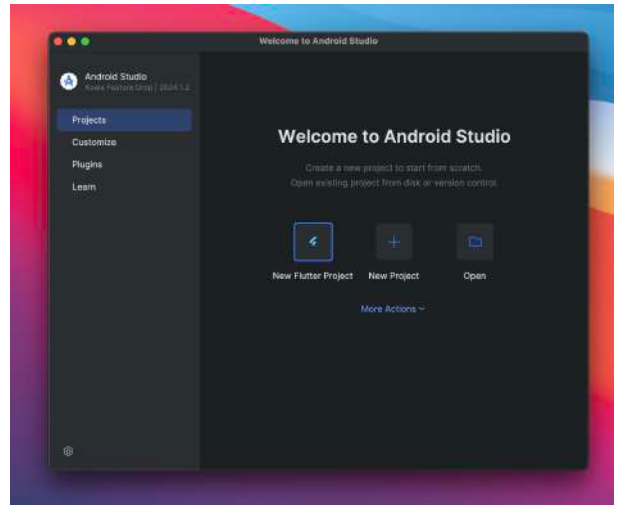
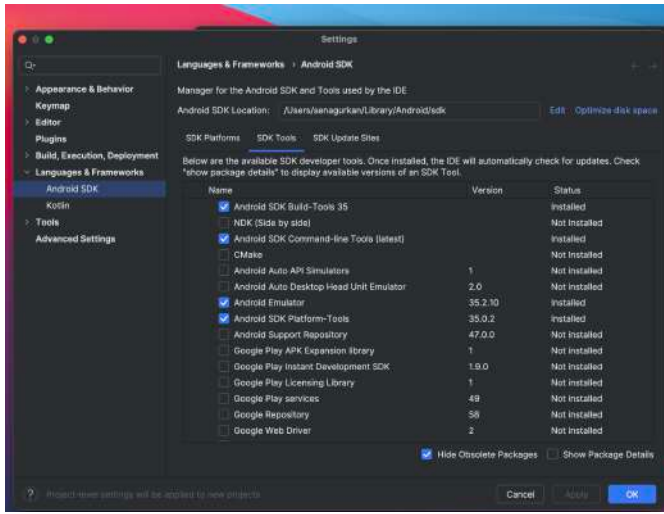
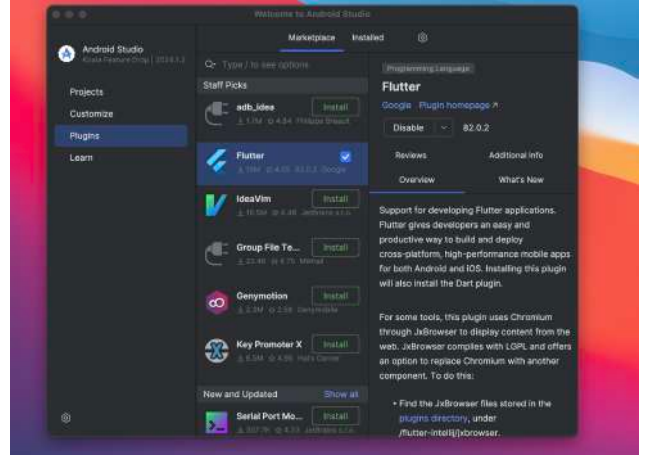
* No issues found!
senagurkan@senas-mac-3 ~ % open .zshrc
senagurkan@senas-mac-3 ~ % open .zshrc
senagurkan@senas-mac-3 ~ % flutter --version
Flutter 3.24.3 • channel stable • https://github.com/flutter/flutter.git
Framework • revision 2663184aa7 (4 weeks ago) • 2024-09-11 16:27:48 -0500
Engine • revision 36335019a8
Tools • Dart 3.5.3 • DevTools 2.37.3
senagurkan@senas-mac-3 ~ %
```

Flutter kurulumu bu şekildeydi, daha sonra Android Studio kurulumuna geçtim.

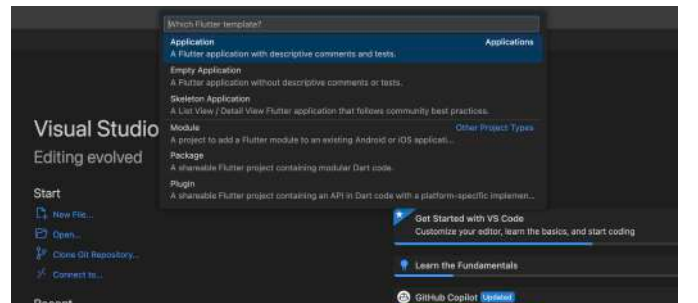
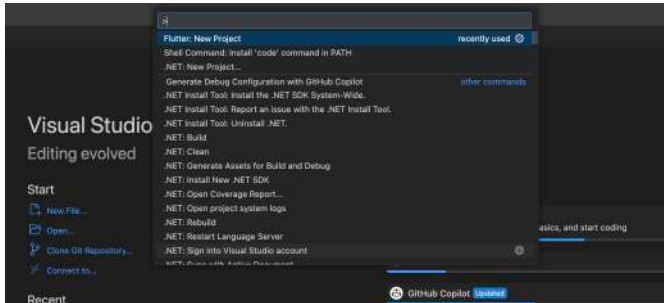
Android Studio Kurulum :

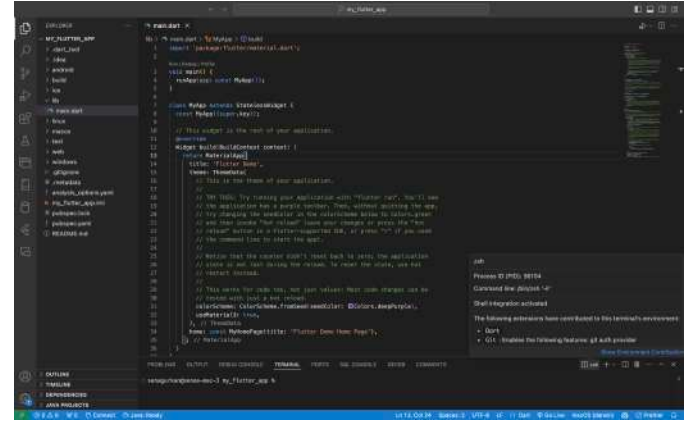
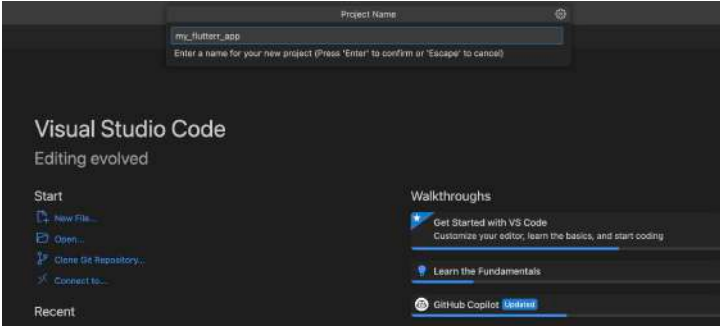
Platform	Android Studio paketi	Boyut	SHA-256 sağlanması
Windows (64 bit)	android-studio-2024.1.2.12-windows.exe Önerilir.	1,2 GB	81aaf21a7e6dd70c0eff77ab74d6ba18eeb1c4249103a42b9fab9094
Windows (64 bit)	android-studio-2024.1.2.12-windows.zip .exe yükleyici yok	1,2 GB	5018dd57d045f4c946bataw92018a0e780f2722502676a5d32E71727779
Mac (64 bit)	android-studio-2024.1.2.12-mac.dmg	1,3 GB	1ec2779c25a544bfc3e3ec7457d53348dd66c8f98bf2c6fd9e61c549
Mac (64 bit, ARM)	android-studio-2024.1.2.12-mac_arm.dmg	1,3 GB	828f9f6564a4d4614cfb3c47700c88e4f92059f92583a4b7042783a6dc2bf
Linux (64 bit)	android-studio-2024.1.2.12-linux.tar.gz	1,2 GB	76536820cf99a908f842447ce54101d8c9d7f5bb20170f6e8f616e92
Chrome OS	android-studio-2024.1.2.12-cros.deb	992,2 MB	0e01c7dfb3ced5b8498f77932981f9228a4c401e87acc04d67d0c776f3c6

Daha fazla indirme burada mevcuttur: [arşivler indirebilirsiniz](#). Örneğin, Android Emulator'ın indirilmesi için Emulator indirme [arşivleri](#).



Daha sonra projemi oluşturdum ve hem iOS hem android simütatörde çalıştım:





Bu da simulatorslarda çalıştırdığım hali :

