



www.youtube.com/BMdersleri

MAKÜ

BURDUR MEHMET AKİF ERSOY ÜNİVERSİTESİ

Flutter ile Mobil Programlamaya Giriş



7.HAFTA

Null Safety ve Asenkron Programlama

1



Hazırlayan	: Zeynep İrem KESLER 1911404048
Tarih	: 25/04/2022
Sürüm	: v1
Ders Yürütücüsü	: Doç. Dr. İsmail KIRBAŞ

İÇİNDEKİLER

- Null Safety
- List tipinin null safety durumları
- Map tipinin null safety durumları
- Late Kullanımı
- Asenkron Programlama
- Future Kavramı
- Senkron ve Asenkron Programlama Farkı
- Yardımcı Kaynaklar



Null Safety

2021'in ilk ayları 2.12 sürümü ile Dart programlama diline **Null Safety** özelliği ekleniyor.

Null Safety, bir değişkenin oluşturulduğu andan itibaren null olamayacağını belirtir. Bu durumda bir değişkene bir değer atamanız veya nullable type olarak belirtmeniz gerekiyor.

```
int nullOlamaz ;  
int? nullOlabilir ;
```

List<String> **birListe** ; (Bu durumda listemiz ve içerisinde null değer olamaz.)

List<String>? **birNullableListe**; (Bu durumda ise listemiz null olabilir ancak içerisinde null değer(ler) olamaz.)

List<String?> **birNullAlabilenListe**; (Listemizin elemanları null değerler alır.)

List<String?>? **nullListe**; (Hem listemizin kendisi null olabilir hem de elemanları null değerler alır.)

NOT: Null olabilen değişkenlerin sonuna bir '?' işareti eklendiği durumda bu değişkenin null değer alabileceğini belirtiriz.

Null Safety

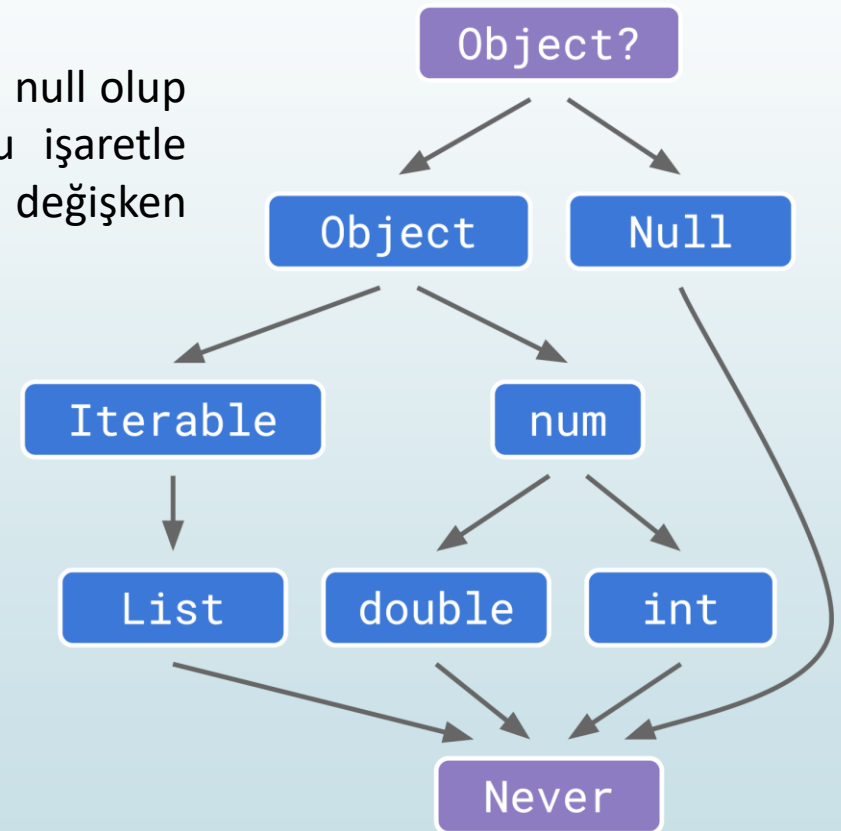
0 != Null

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A
(1 byte) 0_	NUL 0000	SOH 0001	STX 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A
(1) 1_	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A
(1) 2_	SP 0020	! 0021	" 0022	# 0023	\$ 0024	% 0025	& 0026	' 0027	(0028) 0029	* 002A
(1) 3_	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A
(1) 4_	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A
(1) 5_	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A

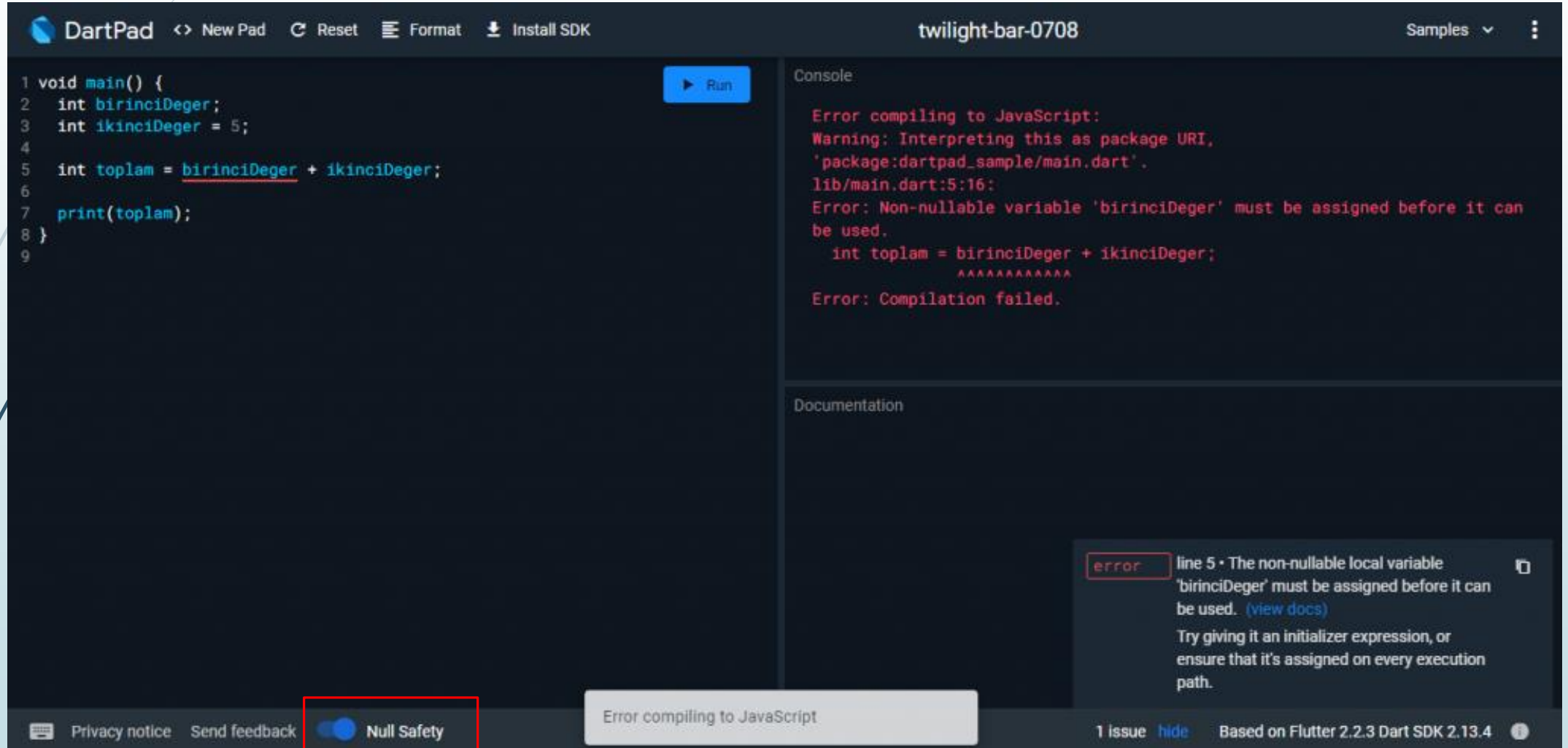
Null Safety

Eğer bir değişkenin null olmadığını biliyorsanız '!' işaretiyle null olup olmama durumunu kontrol etmenize gerek kalmaz. Bu işaretle birlikte Dart derleyicisi, değişkenin null olamayan bir değişken olduğunu anlar.

```
void main() {  
    int? nullOlabilenNum = 2;  
    int value = nullOlabilenNum!;  
}
```



Null Safety



The screenshot displays the DartPad web IDE interface. The top bar includes the DartPad logo, navigation links (New Pad, Reset, Format, Install SDK), the project name 'twilight-bar-0708', and a 'Samples' dropdown menu. The main editor area contains the following Dart code:

```
1 void main() {  
2   int birinciDeger;  
3   int ikinciDeger = 5;  
4  
5   int toplam = birinciDeger + ikinciDeger;  
6  
7   print(toplam);  
8 }  
9
```

A blue 'Run' button is located to the right of the code. The right sidebar is divided into two sections: 'Console' and 'Documentation'. The 'Console' section displays the following error message:

```
Error compiling to JavaScript:  
Warning: Interpreting this as package URI,  
'package:dartpad_sample/main.dart'.  
lib/main.dart:5:16:  
Error: Non-nullable variable 'birinciDeger' must be assigned before it can  
be used.  
    int toplam = birinciDeger + ikinciDeger;  
                  ~~~~~  
Error: Compilation failed.
```

The 'Documentation' section is currently empty. At the bottom of the interface, there is a status bar with links for 'Privacy notice' and 'Send feedback', a 'Null Safety' toggle switch (which is currently turned on), a toast notification that reads 'Error compiling to JavaScript', and a summary of '1 issue' with a 'hide' link. The bottom right corner indicates the environment is 'Based on Flutter 2.2.3 Dart SDK 2.13.4'.

Null Safety

List tipinin null safety durumları:

Tip	List Null mu?	Item Null mu?	Tanım
List<String>	Hayır	Hayır	Null olmayan değerlere sahip bir null olamayan liste
List<String>?	Evet	Hayır	Null olamayan değerlere sahip bir null olabilen liste
List<String?>	Hayır	Evet	Null olabilen değerlere sahip bir null olamayan liste
List<String?>?	Evet	Evet	Null olabilen değerlere sahip bir null olabilen liste

Null Safety

Map tipinin null safety durumları:

Tip	Map Null mu?	Item Null mu?
Map<String, int>	Hayır	Hayır
Map<String, int>?	Evet	Hayır
Map<String, int?>	Hayır	Evet
Map<String, int?>?	Evet	Evet

Null Safety

Late Kullanımı:

Değişkenimizin önüne **“Late”** koymamız ile Dart’a, bu değişkene değer atamadığımızı fakat ilk fırsatta değer atayacağımızı ve null yapmamasını belirtiyoruz. Bu sayede kodumuz, null gözüken değer belirlendiği satıra kadar sorunsuzca çalışmaya devam edecektir.

- Burada “late” yerine “?” kullanmak da işe yarayabilir. Ancak “late” kullanmaktaki amacımız, kodun değer alana kadar sorunsuzca çalışmasını sağlarken null olmayacağından emin olmak.
- Ayrıca **“late”** başka güçlere de sahip, örneğin **initializer** bulunduran bir alanda **“late”** kullanarak bu alanı tembel hale getirebilir tam olarak bir üst düzey değişken veya statik alanda bir initializer gibi çalıştırabilirsiniz. Bu, bir alanın çalıştırılmasının maliyetli ya da gereksiz olduğu anlarda oldukça kullanışlı olabilir.

```
Late String birinciDeger;
```

Asenkron Programlama

Senkron Programlama:

Senkron kelimesi eşzaman anlamına gelmektedir. Yani aynı anda ve eşit zaman aralıklarıyla yapılan iş veya eylem anlamına gelmektedir. Senkron programlama ise programlama yaparken her bir işin sıra ile yapılması anlamına gelmektedir.

Asenkron Programlama:

Kelime anlamı başlama ve bitiş zamanları ayrı olan, aynı zamanda olmayan demek olan kelimedir. Diğer adı da eşzamansızdır.

Asenkron programlama, işin parçalara ayrılıp tüm işlemlerin aynı anda sürdürülmesini sağlar. Yani asenkron programlama ile programımız içerisinde yazdığımız bir kodu işletebilirken, aynı program içerisinde diğer kodları da işletebiliriz. Bu sayede kullanıcı programımızın bir bölümünü kullanırken, başka bir bölümü ile de işlem yapabilir.



Senkron bir restoran

Asenkron Programlama

Asenkron Programlama: Asenkron fonksiyonlarda işleme sonuçları bazen bekletilmek istenebilir. Bunun için **await** ifadesi kullanılır.

```
void function1() {  
    print("Function 1");  
}  
  
Future function2() async {  
    await null;  
    for(int i = 0; i < 1000000000; i++);  
    print("Function 2");  
}  
  
void function3() {  
    print("Function 3");  
}  
  
// Kullanımı  
function1();  
function2();  
function3();  
  
// Çıktı:  
Function 1  
Function 3  
Function 2
```

Asenkron Programlama

Future Kavramı: Asenkron fonksiyonlar; **Future** sınıfını geriye döndürür. Bu sınıf jenerik bir değer alan sınıftır. Yani bir değer döndürmek istiyorsanız, o değer türünü **Future** sınıfına bildirmelisiniz. Future sınıfı size bir callback işlevi sunar.

```
void main(List<String> args) {  
    birinci().then((gelecekDeger) => print(gelecekDeger));  
    print("ikinci");  
}
```

```
Future<int> birinci() async {  
    int toplam=0;  
    for (var i = 0; i<10; i++){  
        toplam = toplam+i;  
    }  
    return toplam;  
}
```

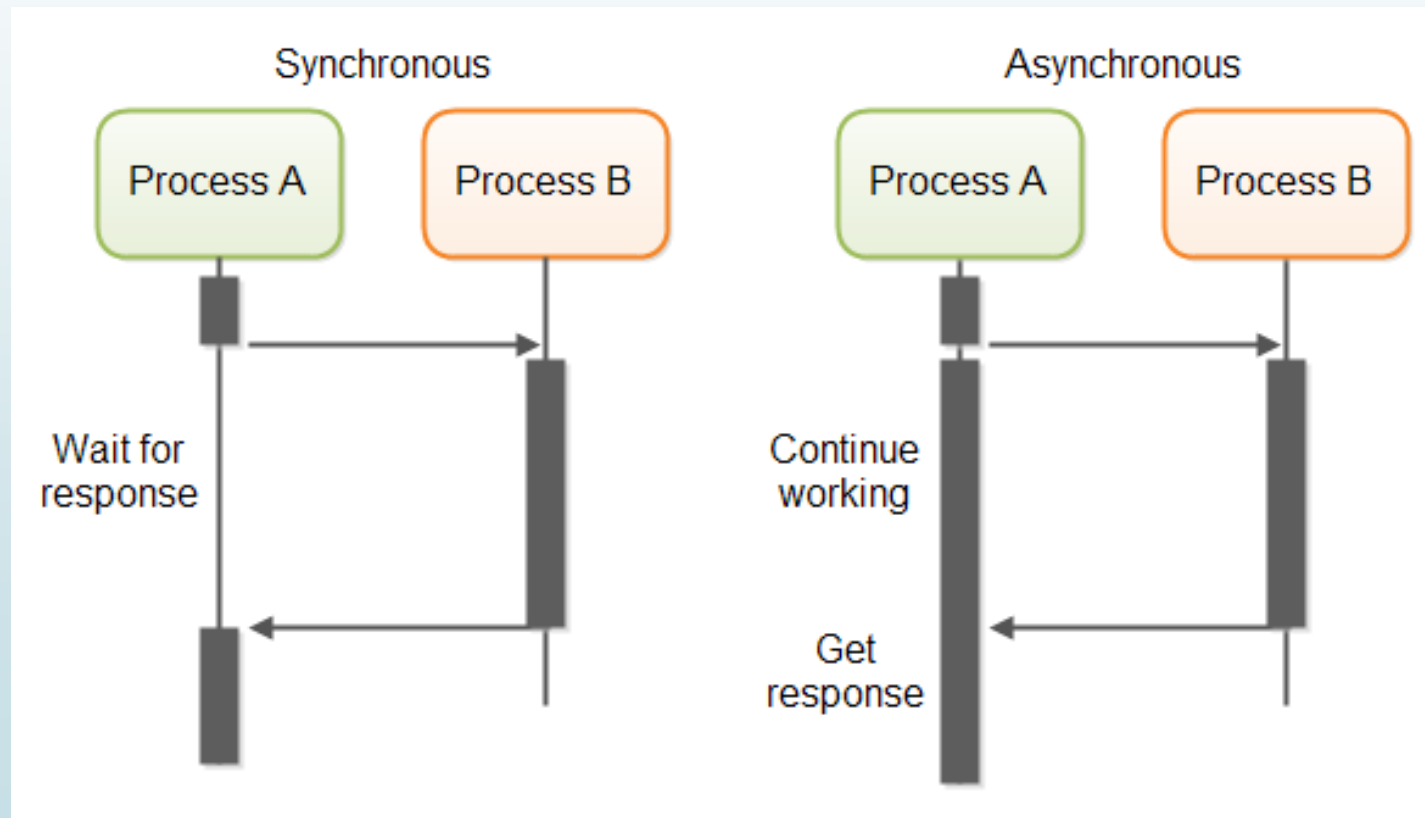
Çıktı:

ikinci

45

Asenkron Programlama

Senkron ve Asenkron Programlama Farkı:



Yardımcı Kaynaklar

- Adım Adım Flutter İle Mobil Uygulamalar (Rakıcı Oğuz , 2021)





www.youtube.com/BMdersleri

MAKÜ
BURDUR MEHMET AKİF ERSOY ÜNİVERSİTESİ

Flutter ile Mobil Programlamaya Giriş



İlginiz için teşekkürler...

15



Hazırlayan
E-posta

: **Zeynep İrem KESLER 1911404048**
: zeynepiremkesler@gmail.com

Tarih

: 25/04/2022

Ders Yürütücüsü

: Doç. Dr. İsmail KIRBAŞ

E-posta

: ismkir@gmail.com