SINIFLAR

Class Yapısı : Sınıf isminin ilk harfini büyük yazmak genel kuraldır.

```
class Ogrenci{

  int numara=111213001;    // örnek değişkeni   instance variable

  String ad='Ali';         // örnek değişkeni   instance variable


void goster(){             // örnekte tanımlanan method
  print('Ogrenci gösterildi');
}
}
```

Bir sınıftan bir nesne üretmek için 4 farklı ifade kullanabiliriz;

```
void main() {
  var ogr1 = new Ogrenci();
  var ogr2 = Ogrenci();
  Ogrenci ogr3 = new Ogrenci();
  Ogrenci ogr4 = Ogrenci();
}
```

Nesneyi ürettikten sonra;

ogr1. dediğimiz zaman bu nesnemize ait ad ve numara değişkenleri ile göster metodu açılan liste içerisinde gösterilir. Bunları seçerek istediğimiz işlemlerde kullanabiliriz.

```
print(ogr1.numara);
print(ogr1.ad);
ogr1.goster();
ogr1.numara=1;
ogr1.ad='Ahmet';
print(ogr1.numara);
print(ogr1.ad);
```

Eğer sınıf tanımlanırken değişkenlere ilk değer ataması yapılmaz ise Null Safety özelliği nedeniyle hata verir. Bu durumda değişken tanımlamasında tip isminin sağına ? (soru işareti) simgesini yazarak bu değişkenlerin null değerine sahip olabilecekleri söylenmiş olur ve hata kalkar.

```
class Ogrenci{
  int? numara;
  String? ad;

void goster(){
  print('Ogrenci gösterildi');
}
}
```

Constructor (Yapıcı): Bir sınıftan nesne üretirken, nesnenin özelliklerinin ilk başta belirlenmesinin, değiştirilmesinin imkanını verir.

```
class Ogrenci{
    int? numara;
    String? ad;


  Ogrenci(int gelennum){
    numara=gelennum;
  }

void goster(){
  print('Ogrenci gösterildi');
}
}
```

Constructor (yapıcı) tanımlaması yapılan bir sınıftan nesne oluşturulması için constructorda belirtilen parametrenin verilmesi gerekir;

```
var ogr1 = new Ogrenci(30);
```

Constructorda tüm özellik değişkenleri tanımlanabilir. Nesne oluşturulurken bu parametreler verileceği için hangi parametrenin hangi özelliğe karşılık geldiği karıştırılabilir.

```
Ogrenci(int gelennum, gelenad){
  numara=gelennum;
  ad=gelenad;
}

var ogr1 = new Ogrenci(30,'Ahmet');
```

Bu karışıklığı önlemek içim **isimli parametre** olarak adlandırılan yapı kullanılabilir. Bunun için küme parantezi kullanırız.

```
Ogrenci({int gelennum, gelenad}){    // Burada int ve String veritipi ismi
                                        Kullanıldığı için Null hatası verir
                                        Bu nedenle ya bu tanımlar
                                        kaldırılır. Yada var olarak
                                        tanımlanır
  numara=gelennum;
  ad=gelenad;
}


Ogrenci({gelennum, gelenad}){
  numara=gelennum;
  ad=gelenad;
}

Veya
Ogrenci({var gelennum, var gelenad}){
  numara=gelennum;
```

```
  ad=gelenad;
}

var ogr1 = new Ogrenci(gelennum: 30, gelenad: 'Ahmet');
```

Constructorda tanımlaman parametre nesnenin özellik değişkeni ile aynı isimde verilebilir ama bu sefer gelen parametre değerinin özellik değişkenine aktarılması için this. Kelimesi kullanılmalıdır.

```
Ogrenci({var numara, var ad}){
  this.numara=numara;
  this.ad=ad;
}

var ogr1 = new Ogrenci(numara: 30, ad: 'Ahmet');
```

Constructorda parametre tanımlamak ve içeride bu parametreye gelen değeri sınıftaki nesneye ait özellik değişkenine atamasını yapmak yerine daha kısa bir yol mevcuttur.

```
Ogrenci({this.numara, this.ad});

var ogr1 = new Ogrenci(numara: 30, ad: 'Ahmet');
```

İstersek Constructorda default değerde verebiliriz.

```
  Ogrenci({this.numara=3, this.ad='Ali'});
```

NAVIGATOR DEVAMI…..

Bir sayfadan gidilecek sayfaya veri aktarmak için;

Gidilecek sayfanın sınıfı için bir constructor yapıcı yazmak gerekir.

```
var gelen;    // yapıcının alacağı değeri kullanmak için oluşturulan değişken
A(this.gelen);  // sınıfın yapıcısı. Bununla A sınıfının kullanıldığı yerde
                    parametre olarak bir değer gönderebiliriz.
```

```
A(this.gelen);  bu şekilde tanımlanırsa  A(5) şeklinde parametre verilir.

A({this.gelen}); bu şekilde tanımlanırsa  A(gelen: 5) şeklinde parametre verilir.
```

İlk sayfadan parametreyi göndermek için;

```
A(this.gelen); şeklinde tanımlanırsa;

Navigator.push(context, MaterialPageRoute(builder: (context)=>A('AAA')));


A({this.gelen}); şeklinde tanımlanırsa;

Navigator.push(context, MaterialPageRoute(builder: (context)=>A(gelen: 'AAA')));
```

Gidilen sayfada parametre olarak gönderilen değeri almak için;

```
print(gelen);
print(this.gelen);
```

Bir sayfadan gidilecek sayfaya NESNE verisi aktarmak için;

```
Önce bir sınıf tanımlarız;
class Ogrenci{
  int? numara;
  String? ad;

  Ogrenci({ this.numara, this.ad='Ahmet'});
  void Goster(){
    print('Öğrenci gösterildi');
  }
}

Sonra gidilecek sayfada nesne oluşturmayı ve constructor'ı belirler ve gelen nesne
verilerini görebilmek için gerekli komutları yazarız;

Ogrenci? ogr=Ogrenci();
Kisi({this.ogr});

print(ogr?.numara);
print(ogr?.ad);
```

Daha Sonra ilk sayfada nesneyi oluşturur ve gönderilecek nesne verisini diğer sayfaya yönlendiririz.

```
Ogrenci ogr1=Ogrenci(numara: 15,);

Navigator.push(context, MaterialPageRoute(builder: (context)=>Kisi(ogr: ogr1,
 )));
```

Gidilen sayfadan ilk sayfaya veri aktarmak için;

Önce gidilen sayfada bir değişken oluştururuz. Pop ile bu veriyi göndeririz.

```
var giden=5;

Navigator.pop(context, giden);
```

Popdan gönderilecek değeri ilk sayfada push ile alabilmek için **iki farklı yöntem** kullanabiliriz.

1. Yöntem; İlk sayfadaki push metodunun .then özelliğini kullanırız

```
Navigator.push(context, MaterialPageRoute(builder:
    (context)=>Kisi(gelen: 'Mobil', ogr: ogr1,))).then(
        (value) => print('popdan gelen $value'));
```

2. Yöntem; Push metodunun popdan gelen veriyi alabilmesi için asenkron tanımlamak ve veriyi beklemesini sağlamak gerekir. Bu nedenle ilk sayfada push metodunu kullandığımız Elevated butonu asenkron tanımlarız.

```
ElevatedButton(onPressed: () async {

  int popgelen= await Navigator.push(context, MaterialPageRoute(builder:
        (context)=>Kisi(gelen: 'Mobil', ogr: ogr1,)));
   print('Popdan gelen $popgelen');

}
```

Gidilen sayfadan ilk sayfaya NESNE verisini aktarmak için;

Gidilen sayfa örneğimizdeki Öğrenci nesnesinin bir özelliğini değiştirelim ve ilk sayfaya öğrenci nesnesini gönderelim;

```
ElevatedButton(onPressed: (){

  print(gelen);
  print(ogr?.numara);
  print(ogr?.ad);
  ogr?.numara=20;
  Navigator.pop(context, ogr);
}
```

Gönderilen öğrenci nesnesini ilk sayfada almak için push metodunun iki farklı kullanımını yazalım.

1. Yöntem;

```
Navigator.push(context, MaterialPageRoute(builder:
    (context)=>Kisi(gelen: 'Mobil', ogr: ogr1,))).then(
        (value) => print('popdan gelen ${value.numara}'));
```

2.Yöntem;

```
ElevatedButton(onPressed: () async {

  Ogrenci popgelen= await Navigator.push(context, MaterialPageRoute(builder:
        (context)=>Kisi(gelen: 'Mobil', ogr: ogr1,)));

  print('Popdan gelen ${popgelen.numara}');

}
```

Gidilecek Sayfaları route olarak tanımlamak;

Bunun için MaterialApp'in routes: parametresini kullanmamız gerekir.
routes: parametresine Map tipinde rota tanımlarının yapılması gerekir.

routes: parametresi kullanıldığı zaman home: parametresi yerine initialRoute: parametresi ile anasayfa adresi verilmelidir.

```
//home: Anasayfa(),
initialRoute: '/',
routes: {
  '/': (context)=>Anasayfa(),
  '/kisi': (context)=>Kisi(),
},
```

Anasayfa içerisinde ise gidilecek sayfa kodu için;

```
Navigator.pushNamed(context, '/kisi');
```

kullanılmalıdır.

İsimli route (Named Route) kullanımında ModalRoute.of ile veri aktarımı;

İlk sayfada pushNamed ile veri gidilecek sayfaya gönderilir.

```
Navigator.pushNamed(context, '/kisi',arguments: 'Mobil' );
```

Gidilen sayfada pushNamed ile gelen veriyi alabilmek için ModalRoute.of(context) kullanılmalıdır. Context istediği için build(context) widgetından sonra bu ifade yazılmalıdır.

```
Widget build(BuildContext context) {

        String deger=ModalRoute.of(context)?.settings.arguments as String;
```

onpressed fonksiyonunda;

```
        print(deger);
```

Bir nesne göndermek istiyorsak;

Anasayfada;

```
        Navigator.pushNamed(context, '/kisi',arguments: ogr1 );
```

Nesne verisini gidilen sayfada;

```
Widget build(BuildContext context) {

          Ogrenci ogr1=ModalRoute.of(context)?.settings.arguments as Ogrenci;
        return Scaffold(…….
```

onpressed fonksiyonunda;

```
      print(ogr1.numara);
      print(ogr1.ad);
```

ile verileri görebiliriz.

İsimli route (Named Route) kullanımında onGenerate ile veri aktarımı;

MateriaApp'da routes yerine onGenerate kullanırız.

```
home: Anasayfa(),
// initialRoute: '/',
   // routes: {
   //   '/':(context)=>Anasayfa(),
   //   '/kisi':(context)=>Kisi()
   // },
   onGenerateRoute: (settings){
     if (settings.name=='/kisi')
       {return MaterialPageRoute(builder: (context)=>Kisi(gelen:5 ,ogr:
settings.arguments as Ogrenci));}
     if (settings.name=='/')
     {return MaterialPageRoute(builder: (context)=>Anasayfa());}
   },
```

İlk sayfada veri gönderimi için;

```
Navigator.pushNamed(context, '/kisi', arguments: ogr1);
```

Gidilen sayfada (Kisi sayfası) veriyi alabilmek için aşağıdaki tanımları yapar;

```
var gelen;
Ogrenci? ogr=Ogrenci();
Kisi({this.gelen, this.ogr});
```

NOT: Bir önceki konuda geçen ModalRoute.of'u kullanmıyoruz…

Ve

```
print(gelen);
print(ogr?.numara);
print(ogr?.ad);
```

ile verileri görebiliriz.

Gidilen sayfadan Anasayfaya veri göndermek için;

1. Yöntem;

```
Navigator.pushNamed(context, '/kisi',arguments: ogr1).then((dynamic value) =>
print(value.numara));
```

2.Yöntem

```
Ogrenci popgelen=await Navigator.pushNamed(context, '/kisi', arguments: ogr1) as
Ogrenci;
print(popgelen.numara);
```