Dart Notlarım **04.09.2024**

- Mixin, Cascade Notation

```dart
void main() {
 Bank moneyBank1 = Bank(50);
 Bank moneyBank2 = Bank(50);
 print(moneyBank1 + moneyBank2);

 //Cascade Notation
 //Birçok nesne işlemini tek satırda yapmamızı sağlayan notasyon.
 moneyBank1
   ..money += 5
   ..regularCustomer(true)
   ..company();
}

class Bank with CustomerTypes {
 int money;

 Bank(this.money);

 int operator +(Bank newBank) {
   return this.money + newBank.money;
 }
}
//mixinler sınıflarınıza extra özellikler ekleyebileceğiniz, içerisinde fonksiyonlar ve değerler tutabilen, birden fazla mixin ile bir araya getirilebilecek yapılardır.
//Kullanım senryosu olarak bir sınıfın 1 kez extend edildikten sonra tekrar extend edilemediğinden o sınıfa yeni özellikler katmak istediğimiz zaman kullanabileceğimiz bir özellik diyebiliriz.
//Extend edilmemiş sınıflar ile de kullanılabilir esnek yapısı vardır
//constructerları olmaz
mixin CustomerTypes {
 late bool onay;
 void regularCustomer(bool onay) {
   print("Basit müşteri $onay");
 }
 void company() {
   print("Şirket");
```

```
 }
}
```

- Singelton(Static)

```
void main(){
List<dynamic> newProduct = [Product.money,
Product.companyName];

newProduct[0] += 5;
print(newProduct[0].runtimeType);
}

//Singelton(static)
//Bu class içerisinde olan değerler program çalışma
hayatı boyunca bilinçli şekilde öldürülmedikçe yaşarlar.
//Kısıt olmaksızın heryerden erişilebilirler.
//Bu yüzden kullanımında çok dikkatli olunmalıdır.
Bilinçsiz müdahale sonucunda tespiti zor crashlere
sebebiyet verebilirler.
class Product{
 static int money = 10;

 //const anahtar kelimesi static kavramının heryerden
müdahaleye izin verme özelliğine karşı değişmemesi
gereken bir değerde önlem olarak kullanılabilir.
 static const companyName = 'company';
}
```

- as,is,is!

```dart
void main(){
 /*
 'as' anahtar kelimesi
 Bir nesneyi belirli bir türdeki bir nesne olarak tür dönüştürmesi yapar.
 Bu tür dönüşümü başarılıysa, nesne dönüşümün sonucunda belirli bir türde olur.
 Eğer dönüşüm mümkün değilse, TypeError hatası fırlatılır.
 */
 dynamic deger = 'aaaaaa';
 String strDeger= deger as String;
 print(strDeger.runtimeType); //Çıktı: String

 /*
 'is' anahtar kelimesi
 Bir nesnenin belirli bir türde olup olmadığını kontrol eder.
 Bu, boolean (true/false) bir değer döndürür: eğer nesne belirli bir türde ise true, değilse false döner.
 */
 if (deger is int) {
   print('deger == int');
 } else {
   print('deger != int');
 }

 // 'is!' is'in tersidir
 if (deger is! int) {
   print('deger != int');
 } else {
   print('deger == int');
 }
}
```

- İleri Seviye Listeler, try-catch-finally,

```
void main() {
 final phoneItems = [
   PhoneModel(model: PhoneModels.Iphone_13, price: 15000,
inch: 6.1, isSecondHand: false),
   PhoneModel(model: PhoneModels.Iphone_14, price:
25000),
   PhoneModel(model: PhoneModels.Samsung_s23, price:
45000, isSecondHand: false),
   PhoneModel(model: PhoneModels.Samsung_s23, price:
35000),
   PhoneModel(model: PhoneModels.Samsung_s23, price:
38000),
 ];

 // phoneItems listesinin içerisinde modeli Iphone_14
olan nesne var mı?
 // any bool döndürür
 print(phoneItems.any((element) => element.model ==
PhoneModels.Iphone_14));

 // phoneItems listesinin içerisinde fiyatı 36000'den
küçük ve model ismi "Iphone" barındıran nesnelerin
adetini döndürür.
 print(phoneItems.where((element) => element.price <
36000 &&
element.model.toString().contains("Iphone")).length);

 final wantedModel = PhoneModel(model:
PhoneModels.Iphone_13, price: 15000, inch: 6.1,
isSecondHand: false);

 // contains default olarak referans değerlerini
karşılaştırır
 // fakat biz containsin kullandığı == operatorünü
PhoneModel sınıfının içerisinde override yaptık ve artık
nesnenin instancelarını karşılaştırıyor
 var isWantedModel = phoneItems.contains(wantedModel);

 if(isWantedModel){
   print("evet istenen model");
 }else print("hayır değil");
```

```dart
 // Where sorgusuyla phoneItems içerisinde aradığımız nesneleri alır, her birini stringe çevirir ve verilen ayırıcı karakter ile birleştirir
 // toString() metodunun class içinde override edilmesi sayesinde nesnelerin istenlen özelliklerine string olarak erişip join metodu sayesinde birleştirebildik
 final resultsS23 = phoneItems.where((element) {
   return element.model == PhoneModels.Samsung_s23 && element.price <40000;
 }).join(" | ");
 print(resultsS23);

 // map() yöntemi, yinelenebilir nesnenin testi geçen tüm öğelerini içeren yeni bir yinelenebilir(iterable) nesne oluşturur.
 final phoneNames = phoneItems.map((e) => e.model).join(' | ');
 print(phoneNames);

 // Dart dilinde .singleWhere() metodu, bir listede veya iterable üzerinde belirli bir koşulu sağlayan tek bir elemanı bulmak için kullanılır.
 // Eğer sadece bir eleman bu koşulu sağlıyorsa, bu eleman döndürülür. Eğer hiçbir eleman koşulu sağlamazsa, StateError hatası fırlatılır.
 // Eğer birden fazla eleman koşulu sağlarsa, yine StateError hatası fırlatılır.
 bool bumper = false;
 try {
   final  sixPointOneInchScreen = phoneItems.singleWhere((element) => element.inch == 6.1);
   print("Model: "+sixPointOneInchScreen.model.toString()+" | Fiyat:"+ sixPointOneInchScreen.price.toString());
   bumper = true;
 } catch (e) {
   print('6.1 inch telefon stoklarda yok.');
 } finally {
   print('Stok durumu: $bumper');
 }

 // Dart dilinde .indexOf() metodu, bir listenin (veya bir string'in) içerisinde belirli bir elemanın ilk olarak geçtiği indeks numarasını bulmak için kullanılır.
```

```
 // Eğer eleman listede bulunuyorsa, o elemanın bulunduğu
ilk indeks değeri döndürülür; eğer eleman listede yoksa,
-1 döndürülür.
 final index = phoneItems.indexOf(wantedModel);
 print(index);

 phoneItems.add(wantedModel);
 print(phoneItems);

 phoneItems.remove(wantedModel);
 print(phoneItems);

 phoneItems.sort((first, second) =>
first.price.compareTo(second.price));
 print(phoneItems);
}

class PhoneModel {
 PhoneModels model;
 int price;
 double? inch;
 bool isSecondHand;

 PhoneModel(
     {required this.model,
     required this.price,
     this.inch,
     this.isSecondHand = true});

 @override
 bool operator ==(Object other) {
   if (identical(this, other)) return true;

   return other is PhoneModel &&
       other.model == model &&
       other.price == price &&
       other.inch == inch &&
       other.isSecondHand == isSecondHand;
 }

 @override
 String toString() {
   return '$model - $price';
 }
```

```
}

enum PhoneModels {
 Samsung_s23,
 Iphone_13,
 Iphone_14,
 Iphone_15,
}
```