MOBİL UYGULAMALAR
(Dart & Flutter)

HAFTA 3 : Dart Programlama Dili

Ayhan AKKAYA

Flutter Developer and Trainer

Bilgisayar Yüksek Mühendisi

akkayasoft

Değişkenler (Variables)

- Değişkenler uygulamalar için bellek alanlarında tutulan değerlere referans veren kayıtlardır. Örneğin, Kod 14'te sayi ismindeki değişken, int türündeki 3 değerine bir referanstır. Değişkende tutulan değer atama işleçleri kullanılarak düzenlenebilir. Kodlarda sayi ifadesi kullanıldığında derleyici bu ifadeyi referans verdiği 3 değeri ile değiştirerek işleme sokmaktadır. Bu sayede, 2 numaralı satırdaki ifadenin sonucunda sayi değişkenine 1 eklenerek ekrana 4 yazdırılmaktadır.

```
1    void main() {
2      var sayi = 3;
3      print(++sayi); // 4
4    }
```

Değişkenler (Variables)

- Dart değişkene atanan değere göre hangi tipte olması gerektiğine karar verebilmektedir. Bu nedenle değişkenlerin var anahtar sözcüğü ile tanımlanması önerilir.
- Tanımlanan bir değişkenin tekrar değiştirilmesi istenmiyorsa bu değişken final anahtar sözcüğü ile tanımlanabilir.
- Derleme süreci sabitlerinin tanımlanması için const anahtar kelimesi kullanılır.

```
1   void main() {
2     final isim = "Dart";
3     isim = "Programlama"; //The final variable 'isim' can only be set once.
4     const versiyon = 2.12;
5     versiyon = 3; // Constant variables can't be assigned a value.
6   }
```

Null-Safety Mekanizması

- Null-Safety mekanizması içeriği null olan nesnelerin kullanılmasından kaynaklanan çalışma zamanı hatalarını önlemek için geliştirilmiştir.
- 2 numaralı satırda int tipindeki sayi değişkeni tanımlanmış fakat değer ataması yapılmamıştır. Bu durumda değişkenin içeriği boş, yani null'dur. 3 numaralı satırda sayi değişkenindeki değer 2 ile çarpılmak istenmektedir. Bu kod, null değerini 2 ile çarpmaya çalıştığından çalıştırıldığı anda bir hata (ör: TypeError: null has no properties) üretecektir.

```
1   void main() {
2     int sayi;
3     print(sayi*2);
4   }
```

Null-Safety Mekanizması

- Bir değişken tanımlanırken null değer alabilir (nullable) olarak belirtilebilir. Bu amaçla değişkenin tip belirtecinin arkasına ? işleci eklenir.
- var anahtar sözcüğü arkasında ? kullanımı hata üretecektir.

```
1    void main() {
2      int sayi = null;
3      int? sayi2 = null;
4    }
```

```
1    void main() {
2      String? versiyon;
3      print(versiyon.length);
4      print(versiyon!.length);
5      // (versiyon != null) ? versiyon.length : null;
```

Öntanımlı Veri Tipleri - Sayılar

- Dart iki tür sayısal veri tipi sağlar. Tamsayılar için int veri tipi kullanılır. int veri tipindeki tam sayıların sınırları kullanılan platforma göre değişmektedir.

```
1    void main() {
2      var sayi = 1;
3      var onaltilik = 0xA00;
4      var ustel = 8e5;
5      var ondalikli = 1.1;
6      var ustel2 = 1.42e5;
7    }
```

Öntanımlı Veri Tipleri - Metinler

- String nesneleri UTF-16 standardındaki karakterlerden oluşan katarlardır. String tipinde nesneler üretmek için karakterler tek tırnak ya da çift tırnaklar arasına alınır.

```
1   void main() {
2     var s1 = 'tek tırnak içinde metin.';
3     var s2 = "çift tırnak içinde metin.";
4     var s3 = 'Tek tırnak içinde kesme (\') işaretini göstermek.';
5     var s4 = "Çift tırnak içinde kesme (') işaretini göstermek";
6     var s = 'ekleyeblir';
7     print('Dart metinler içine farklı metinleri $s');
8   }
```

Öntanımlı Veri Tipleri - Boolean

- Mantıksal verilerin tutulması için Dart bool veri tipini destekler. bool tipindeki değişkenler yalnızca true ya da false değerlerini alabilir.

```
1    void main() {
2      var tekMi = true;
3      var ciftMi = false;
4    }
```

Listeler

- Programlama dillerinde en çok kullanılan dizi yapılarından biri listelerdir. Listeler benzer özellikteki nesnelerin (Ör: sayılar, metinler) gruplarından oluşur.

```
1    void main() {
2      var sayilar=[1, 2, 4];
3      var alisverisListem=[
4        "ekmek",
5        "süt",
6        "yoğurt",
7      ];
8      print(alisverisListem[0]);
9      print(alisverisListem.toString());
10   }
```

Listeler

- Bir listenin elementlerine erişmek için listenin atandığı değişkenin arkasından köşeli parantezler içinde elementin sıra numarası verilir. Dart listeleri 0 sıra numarası ile başlatılır.
- Listelerde tanımlı elementler atama işleci kullanılarak değiştirilebilir.

```
1    void main() {
2      alisverisListem[1]="meyve suyu";
3      sayilar[2]=5;
4      sayilar[3]=10;
5    }
```

Listeler

- Listelerin sonuna element eklemek için **add** metodu kullanılır.
- Listede tanımlı olan bir elementi silmek için **removeAt** metodu kullanılabilir.
- Listede araya bir element eklemek için **insert** metodu kullanılır.

```
1    void main() {
2      sayilar.add(10);
3      alisverisListem.removeAt(2);
4      alisverisListem.insert(2, "zeytin");
5      print(alisverisListem.toString());
6    }
```

Listeler

- Listelerdeki elementler üzerinde işlem gerçekleştirmek için **forEach** metodu kullanılabilir.
- Bu metot her bir elementi kendisine parametre olarak verilen anonim fonksiyona gönderir.

```
1    void main() {
2      alisverisListem.forEach((element)=>print(element));
3    }
```

Listeler

- Listelerde arama yapmak içi **firstWhere** metodu kullanılabilir. Bu metot listedeki elementleri verilen test ifadesine göre tarar. Test ifadesini sağlayan ilk elementi geri çevirir. Test ifadesi sağlanmazsa orElse bloğundaki ifade çevrilebilir.

```
1    void main() {
2      var sonuc = alisverisListem.firstWhere(
3        (element) => element == "gazete",
4        orElse: () {
5          return "aranan ifade bulunamadı";
6        });
7      print(sonuc);
8    }
```

Listeler

```
1   void main() {
2     var sonuc=alisverisListem.indexWhere(
3       (element)=> element == "süt"
4     );
5     if(sonuc>-1){
6       sonuc++;
7       print("aranan ifade $sonuc. sırada");
8     }
9     else
10    {
11      print("aranan ifade bulunamadı");
12    }
13  }
```

Listeler

- İki listenin birleştirilmesi için yayılım işleci (...) kullanılabilir. Bu işlem listelerden birinin taranarak elementlerin teker teker diğerine eklenmesi yoluyla da gerçekleştirilebilir. Fakat yayılım işlecinin kullanımı çok daha kolaydır.

```
1    void main() {
2      var dogalSayilar=[0, 1, 2, 4];
3      var tamSayilar = [-1, -4, -11, ...dogalSayilar];
4      print(tamSayilar.toString());
5    }
```

Listeler

```
1    void main() {
2      var dogalSayilar = [1,2,3];
3      var tamSayilar = [-1, -2, -3];
4      dogalSayilar.forEach((sayi)=>tamSayilar.add(sayi));
5      print(tamSayilar.toString());
6    }
```

Kümeler

- Kümeler, adından da anlaşılabileceği gibi birbirinden farklı elementlerin sırasız bir listesidir. Listelerden farklı olarak kümelerde aynı ifade iki farklı element olarak yer alamaz.

```
1    void main() {
2      var haftaIci = {"salı", "çarşamba", "perşembe", "cuma"};
3      var gunler = <String>{};
4    }
```

Kümeler

```
1    void main() {
2      var haftaSonu = {"cumartesi", "pazar"};
3      gunler.add("pazartesi");
4      print(gunler.toString());
5      gunler.addAll(haftaIci);
6      gunler.addAll(haftaSonu);
7      gunler.add("pazartesi");
8      print(gunler.toString());
9    }
```

Kümeler

```
1    void main() {
2    var ilkGun=gunler.elementAt(0);
3    var ilkGun=gunler.first;
4    var gunSayisi = gunler.length;
5    print("günler kümesinde $gunSayisi adet gün var.");
6    }
```

```
1     void main() {
2       gunler.forEach((element)=> print(element));
3     }
```

Kümeler

```
1    void main() {
2      if(gunler.lookup("salı")!=null)
3      {
4        gunler.remove("salı");
5      }
6    }
```

```
1     void main() {
2       sayilar.removeWhere((element)=> element>9);
3       sayilar.clear();
4     }
```

Kümeler

```
1   void main() {
2     var aranan = "cumartesi";
3     if(gunler.contains(aranan))
4     {
5       print("$aranan günler kümesinde yer alıyor.");
6     }
7     else
8     {
9       print("$aranan günler kümesinde yer almıyor");
10    }
11  }
```

Kümeler

```
1    void main() {
2      var sayilar = {1, 2, 3, 6, 10, 12};
3      var ondanKucukSayilar = sayilar.where((element)=> element<10);
4      print("sayilar kümesindeki ondan küçük sayılar: $ondanKucukSayilar");
5    }
```

Haritalar

- Haritalar bir anahtar ve buna bağlı bir değerden oluşan nesnelerin bir kümesidir. Haritalarda anahtarlar eşsiz olmak zorundadır. Fakat bi değer birden fazla anahtara atanabilir. Anahtarlar ve değerler farklı veri türlerinde olabilir. Örneğin int tipinde bir anahtara String tipinde veriler atanabilir.

```
1   void main() {
2     var elementler = {
3       1: "hidrojen",
4       3: "lityum",
5     };
6     var baskentler = Map<String, String>();
7     var soyGazlar = Map<int, String>();
8   }
```

Haritalar

```dart
void main() {
  elementler[4] = "berilyum";
  baskentler["tr"] = "Ankara";
  var soygazlar = {
    2: "helyum",
    10: "neon",
    18: "argon",
    36: "kripton",
    54: "ksenon",
    86: "radyum",
  };
  elementler.addAll(soygazlar);
  print(elementler);
}
```

Haritalar

```
1    void main() {
2      var elementSayisi = elementler.length;
3      print("elementler haritasında şuan $elementSayisi adet element ekli");
4    }
```

```
1     void main() {
2       elementler.forEach(
3         (an, deg)=> print("Atom numarası: $an, Element: $deg")
4       );
5     }
```

Haritalar

```
1    void main() {
2      elementler.remove(3);
3      elementler[86] = "radon";
4      elementler.update(86, "radon");
5    }
```

```
1    void main() {
2      var elementler = {"H" : "helyum", "Li": "lityum"};
3      var simgeler = elementler.keys;
4      var isimler = elementler.values;
5      print(simgeler);
6    }
```