MOBİL UYGULAMALAR
(Dart & Flutter)

HAFTA 2 : Dart Programlama Dili

Ayhan AKKAYA

Flutter Developer and Trainer

Bilgisayar Yüksek Mühendisi

akkayasoft

Temel Kavramlar

- Dart çeşitli platformlarda çalışabilecek uygulamalar üretmek için geliştirilmiş bir programlama dilidir (Google, t.y.). Dart dilinin amacı platformlara (Ör: iOS, Android) özgü istemci uygulamalar üretmektir. Flutter uygulamaları Dart programlama dili ile kodlanır.
- Dart nesne yönelimli bir dildir. Değişkenlere atanan her ifade (Ör: sayılar, fonksiyonlar) bir sınıftan türetilen nesnelerdir.
- Dart, Java gibi dillerde kullanılan public, protected erişim belirleyicilerini kullanmaz. Bir ifadenin kütüphaneye özgü kalması ve dışarıdan erişilmemesi isteniyorsa önüne alt çizgi eklenir

(Ör: _ozelDegiskenim).

Temel Kavramlar

- Dart, sıkı veri tipi kontrolü (strongly typed) kullanır. Bu sayede değişkenler tek türde veri (Ör: int, String) tutabilir. Bir değişkene atanamayacak bir değerin atanmasını isteyen kodlar derleme sırasında yakalanır ve hataların ayıklanması istenir. Örneğin sayısal değerler alabilen bir değişkene Boolean tipinde bir değer atamaya çalışıldığında, Dart derleyecisi bir hata mesajı vererek uygulamayı derlemez.
- Dart dilinde değişkenlerin tanımlandığı satırlarda belirli bir veri tipinin belirtilmesi zorunluluğu yoktur. Dart'ın tip çıkarımı mekanizması gerçekleştirilen atamayı temel alarak değişkenin hangi tipte olması gerektiğine karar verebilir. Örneğin sayi değişkeni int tipinde tanımlamak zorunda değildir. Bunun yerine var anahtar sözcüğü ile tanımlanır.

Temel Kavramlar

- Dart null güvenliği mekanizması değer atanmamış değişkenlerin kullanılmasından kaynaklanan hataları (null exception) engeller. Programcı özellikle belirtmediği sürece Dart değişkenleri null değerini alamaz.
- Sıcak yükleme (hot reload) mekanizması sayesinde, yazılan Dart kodları koşturulmakta olan bir uygulamaya enjekte edilebilir. Bu sayede bir kez çalıştırılan bir uygulama üzerinde, uygulama halen çalışıyorken düzenleme yapmaya devam edilebilir

Dart Uygulamaları

- Dart farklı görevleri olan çekirdek kütüphaneler kullanmaktadır. Temel veri tipleri, listeler ve bu verilerle ilgili işlemler için gerekli araçlar **dart:core** kütüphanesi tarafından sağlanır.
- Kütüphanelerin uygulamaya eklenmesi için **import** anahtar sözcüğü kullanılır.

```
1  import "dart:math";
2  void main() {
3    var sayi = 40;
4    print(karesiniAl(sayi));
5  }
6  num karesiniAl(sayim) {
7    return (pow(sayim, 2));
8  }
```

Dart Uygulamaları

- Dart uygulamaları main adındaki üst düzey fonksiyondan başlatılır. Uygulama bu fonksiyondan başlayarak gerekli dallara ayrılır. 2 numaralı satırda main fonksiyonu tanımlanmaktadır. Sıkı veri tipi kontrolü mekanizması nedeniyle değer döndürmesi beklenen fonksiyonların hangi türde değer döndüreceğinin belirtilmesi gerekmektedir. Değer döndürmesi beklenmeyen fonksiyonların ise void tipi ile tanımlanması gerekir.

```
1  import "dart:math";
2  void main() {
3    var sayi = 40;
4    print(karesiniAl(sayi));
5  }
6  num karesiniAl(sayim) {
7    return (pow(sayim, 2));
8  }
```

Dart Uygulamaları

- 2 numaralı satırdaki main fonksiyonu bir değer döndürmediğinden void tipi ile tanımlanmıştır.
- 6 numaralı satırdaki karesiniAl fonksiyonu ise num tipinde sayısal değer döndürecek şekilde tanımlanmıştır.

```
1  import "dart:math";
2  void main() {
3    var sayi = 40;
4    print(karesiniAl(sayi));
5  }
6  num karesiniAl(sayim) {
7    return (pow(sayim, 2));
8  }
```

Dart Uygulamaları

- Dart dilinde değişkenleri tanımlamak için var anahtar sözcüğü ya da tip belirteçleri (ör: String, num) kullanılır. Değişkenler var ile tanımlanırsa, Dart ilk atamada tip çıkarımı mekanizması ile değişkenin tipine karar verir.
- 3 numaralı satırda sayi isimli değişken tanımlanarak 40 değeri bu değişkene atanmaktadır.

```
1  import "dart:math";
2  void main() {
3    var sayi = 40;
4    print(karesiniAl(sayi));
5  }
6  num karesiniAl(sayim) {
7    return (pow(sayim, 2));
8  }
```

Dart Uygulamaları

- Dart dilinde tanımlı fonksiyonlar isimleri ile çağırılır. Bu fonksiyonlara değer göndermek için parametreler kullanılabilir.
- 6 numaralı satırda tanımlanan karesiniAl fonksiyonu sayim adında bir parametre beklemektedir.
- 4 numaralı satırda, sayi değişkeni karesiniAl fonksiyonuna parametre olarak gönderilmektektedir. sayi değişkenindeki 40 değerine karesiniAl fonksiyonu içinde sayim parametresi yoluyla erişilecektir.

```
1  import "dart:math";
2  void main() {
3    var sayi = 40;
4    print(karesiniAl(sayi));
5  }
6  num karesiniAl(sayim) {
7    return (pow(sayim, 2));
8  }
```

Dart Uygulamaları

- 7 numaralı satırda dart:math kütüphanesi tarafından sağlanan pow fonksiyonu kullanılarak sayim değişkenindeki değerin karesi hesaplanmaktadır. Bu değer return anahtar sözcüğü kullanılarak geriye çevrilmektedir. return anahtar kelimesi ile uygulama
- 4 numaralı satırdaki karesiniAl çağrı noktasına geri dönecektir. Burada karesiniAl çağrı ifadesi yerine döndürülen değer print fonksiyonuna gönderilerek konsoldan yazdırılmaktadır.

```
1  import "dart:math";
2  void main() {
3    var sayi = 40;
4    print(karesiniAl(sayi));
5  }
6  num karesiniAl(sayim) {
7    return (pow(sayim, 2));
8  }
```

Yorum Satırları

- Yorum satırları programcıların kendilerine ya da çalıştıkları takımlardaki diğer programcılara mesaj bırakmak için kullandıkları yapılardır. Dart ile iki türde yorum satırı eklenebilir.
- Tek satıra sığabilecek kadar kısa yorumlar eklemek için // işareti satırın basına eklenir.

1.3.1. Yorum satırları

```
1    void main() {
2      // Dart uygulamaları main() fonksiyonu ile başlar.
3      // Bu uygulama konsola “uygulama başladı” yazmaktadır.
4      print("uygulama başladı");
5    }
```

Kod 6. Yorum satırı kullanımı

Yorum alanları

- Tek bir satırı aşacak yorumları eklemek için yorum alanları /* ve */ işaretleri ile belirlenir. Yorum alanını başlatmak için /* kullanılır. Bu kullanımdan sonra */ işaretine kadar yazılan her karakter yorum olarak kabul edilir.

```
1   void main() {
2     /*
3       Dart uygulamaları main() fonksiyonu ile başlar.
4       Bu fonksiyon konsola “uygulama başladı” yazmaktadır
5     */
6     print("uygulama başladı");
7   }
```

İşleçler (Operatörler)

- Programlama dilinin derleyicisine bir işlem gerçekleştirme komutunu vermek için işleçler kullanılır.
- Örneğin iki adet değerin toplanması işlemini gerçekleştirmek için bu değerler toplama işleci (+) ile birleştirilir.
- İki değer bir işleç ile birleştirildiğinde bir ifade oluşturulmuş olur. Derleyiciler otomatik olarak bu ifadelerin sonuçlarını üretir.
- Son ek işleçler bir ifadenin arkasında eklenen işleçleri ifade eder.
- Ön ek işleçler ise ifadenin önüne getirilerek kullanılan işleçleri ifade eder.

İşleçler (Operatörler)

Son ek işleçler	`ifade++  ifade--  ()  []  .  ?.`
Ön ek işleçler	`-ifade  !ifade  ~ifade  ++ifade  —-ifade`
Çarpma / Bölme	`*  /  %  ~/`
Toplama / Çıkarma	`+  -`
İlişki kontrolü	`>=  >  <=  <`
Eşitlik kontrolü	`==  !=`
Mantıksal VE	`&&`
Mantıksal VEYA	`||`

Aritmetik İşleçler

```dart
void main() {
  print(4 + 6); // 10
  print(3 - 6); // -3
  var sayi = 10;
  print(-sayi); // -10
  print(3 * sayi); // 30
  print(sayi / 3); // 3.33
  print(sayi ~/ 3); // 3
  print(sayi % 3); // 1
}
```

Ön Ek ve Son Ek Arttırma-Azaltma İşleçleri

```dart
void main() {
  var sayi = 10;
  print(++sayi); // 11
  print(sayi++); // 11
  print(sayi); // 12
  print(sayi--); // 12
  print(--sayi); // 10
  print(sayi); // 10
}
```

Not : Ön ek işleçlerin sonuçları ifade işletilmeden önce hesaplanarak değişkene aktarılır. Son ek işleçlerde ise, işlecin etkisi ifade işletildikten sonra değişkene aktarılır.

İlişkileri Kontrol Etme İşleçleri

```
1    void main() {
2      print(2==2); // true
3      print(2!=2); // false
4      print(2>2); // false
5      sayi = 2;
6      print(sayi>=2); // true
7      print(3<=sayi); // false
8    }
```

Atama ve Birleşik Atama İşleçleri

```
1    void main() {
2      sayi = 2; // 2
3      sayi += 3; // sayi = sayi + 3;
4      print(sayi); // 5
5      sayi *= 3; // sayi = sayi * 3;
6      print(sayi); // 15
7      sayi ~/= 4;
8      print(sayi); // 4
9    }
```

Durumsal Atama

```dart
void main() {
  var sayi = 3;
  var sonuc = (sayi % 2 == 0) ? "çift" : "tek";
  print(sonuc); // tek
  /*
  var sonuc;
  if(sayi % 2 ==0)
  {
    sonuc = "çift";
  }
  else
  {
    sonuc = "tek";
  }
  */
}
```

Mantıksal İşleçler

```
1    void main() {
2      print(!true); // false
3      print(true && true); // true
4      print(true && false); // false
5      print(true || false); // true
6    }
```