



# **Bölüm 2: Programlama**

## **JAVA ile Nesne Yönelimli Programlama**



# Programlama Dilleri

- Bir makine tarafından gerçekleştirilebilecek hesaplamaları ifade etmek amacıyla tasarlanmış yapay bir dildir.
- Makinenin davranışını kontrol eden programlar oluşturmak için kullanılır.
  - Algoritmaları açık bir şekilde ifade ederek veya
  - İnsan ile iletişim şekli olarak kullanılabilir.
- Bilgisayarlar ve diğer cihazlar üzerinde işlevsellik sağlar.
- Algoritmaların ve mantıksal süreçlerin ifadesi için kullanılır.
- Programlama dili seçimi projenin ihtiyaçlarına göre yapılır.
- **Örnek diller:** C, C++, **Java**, Python, Prolog, Haskell, Scala, vb.



# Dilin Özellikleri

- Temel yapılar
  - Programlama dilleri ve doğal diller birçok ortak yön paylaşır.
  - **Programlama dili** – sayılar, diziler, basit operatörler.
  - **Türkçe** – kelimeler, noktalama işaretleri.
- Sözdizimi
  - Dilin kurallarını belirler, dilin anlaşılabilirliğini etkiler.
  - **Programlama dili** - Hangi karakter ve sembol dizilerinin düzgün olduğunu belirler.
  - **Türkçe** - "kedi köpek çocuk" kabul edilebilir bir cümle formunda olmadığı için sözdizimi açısından geçerli değildir.



## Dilin Yönleri – Statik Anlam (Semantic)

- Sözdizimi olarak doğru ifadelerin, anlamsal olarak da doğru olup olmadığını kontrol eder.
- **İngilizce** - "I are big," form olarak <isim> <fiil> <isim> şeklinde geçerli bir sözdizimine sahip olabilir, ancak "I" tekil, "are" çoğuldur, bu nedenle dil açısından geçerli değildir.
- **Programlama dili** - <değer> <operatör> <değer> geçerli bir sözdizimdir, ancak 2.3/'abc' bir statik anlamsal hata oluşturur.



# Dilin Yönleri – Anlam (Semantic)

- Semantik, bir dilde sözdizimi açısından doğru ifadelerin ne anlama geldiğini belirler.
  - İngilizce gibi doğal dillerde bazen çokanlamlılık olabilir.
    - “They saw the man with the telescope.”
- Programlama dilleri açısından, herhangi bir geçerli ifade sadece bir anlama sahiptir,
- Ancak; bu anlam programcının beklediği şey olmayabilir.
- **Örneğin**, yanlış bir işlem sonucu beklenmeyen bir sonuç verebilir.



# Neler Yanlış Gidebilir?

- Sözdizimi hataları,
  - Yazım hataları gibi açıkça tanımlanmış kurallara aykırı durumları ifade eder ve genellikle bilgisayarlar tarafından kolayca tespit edilir.
- Statik anlamsal hatalar,
  - Dil bazı kontrolleri çalıştırmadan önce yapar. Eğer bu hatalar yakalanmazsa, programın davranışı tahmin edilemez hale gelebilir.
- Programlar, geçerli sözdizimi ve statik anlamsal hatalara sahip olsa bile, beklenen sonucu üretmeyebilir.
  - Bu durumlar, çökmeler, sonsuz döngüler veya istenmeyen sonuçlar şeklinde ortaya çıkabilir.



# Amaç

- Bir programlama dilinin sözdizimi (syntax) ve anlamını (semantic) öğrenmek:
  - Bir programlama dilinin temel yapı taşlarını kavramak.
  - Sözdizimi ve semantiğini öğrenmek.
- Bu unsurları kullanarak sorunları çözmek için "tarifleri" bilgisayarın anlayacağı şekilde çevirmeyi öğrenmek:
  - Problemin çözümünü bilgisayarın kullanabileceği bir formüle dönüştürmek.
- Hesaplamalı Düşünme ile Problemleri Çözme:
  - Problemi çözmek için bir dizi yöntemi kullanabilme yeteneği.



# Programlama Dilleri

- Düşük seviyeden yüksek seviyeye farklı soyutlama seviyelerine sahiptir.
- **Düşük seviye diller** (örneğin Assembly), neredeyse makine seviyesinde çalışır, çok basit hesaplama talimatlarına sahiptir.
- **Yüksek seviye diller** (örneğin Python, C, Java), zengin ve karmaşık bir talimat kümesine sahiptir, daha yüksek soyutlama seviyesini destekler.





# Programlama Dilleri

- Uygulama alanına göre genel amaçlı veya hedeflenmiş (targeted) olabilir.
- **Genel amaçlı diller**, çeşitli uygulamaları desteklemek için geniş bir temel talimat kümesine sahiptir.
- **Hedeflenmiş diller**, belirli bir uygulama türüne odaklanır ve bu tür işlemleri kolaylaştırmak için özelleştirilir.
- Örneğin, MATLAB sayısal hesaplamalar için özel tasarlanmıştır, matris ve vektör işlemleri optimize edilmiştir.



# Programlama Dilleri

- Kodun yürütülmesine göre yorumlanan veya derlenen olarak sınıflandırılır.
- **Yorumlanan dillerde**, kaynak kod doğrudan yorumlayıcı tarafından çalıştırılır ve program akışı talimatları sırayla işler.
- **Derlenen dillerde**, kaynak kod, yürütülmeden önce derleyici tarafından nesne koduna çevrilir. Derleme işlemi, hızlı ve optimize bir yürütme sağlar.



# Programlama Dili Paradigmaları

- Programlama dilleri, problemi farklı yaklaşımlarla çözebilir.
- **Fonksiyonel** diller, hesaplamayı matematiksel işlevlerin değerlendirilmesi olarak ele alır. Örnekler: Lisp, Scheme, Haskell, vb.
- **Buyurulu** diller, programcı makineye durumunu nasıl değiştireceğini talimat verir. Örnekler: FORTRAN, BASIC, Pascal, C, vb..
- **Mantıksal** diller, hesaplamaların mantığını açıklarken kontrol akışı hakkında ayrıntı vermez. Örnek: Prolog
- **Nesne yönelimli** diller, talimatları, üzerinde işlem yaptıkları durumun bir parçasıyla gruplar. İlişkili veri ve metotları nesne adı altında bir araya getirir. Örnekler: C++, Java, C#, Python, vb.



# İfadeler

- Hesaplamaları tanımlayan matematiksel ifadelerdir.
- İfadeleri değerlendirmek ve sonuçları için belirli kurallar takip edilir.
- İfade içindeki işleçlerin sırası ve önceliği için belirli kurallar izlenir.
  - Örneğin, çarpma ve bölme, toplama ve çıkarmadan önceliklidir.
- Üs alma operatörü \*\*, diğerlerinden farklı olarak sağa doğru işlenir.
- İşlem önceliklerini geçersiz kılmak için parantezler kullanılır.



# Örnek İfadeler

- 5
- $3 + 4$
- $44 / 2$
- $2^{**} 3$
- $34 + 56$
- $(72 - 32) / 9 * 5$



# Değişkenler

- Programda değerleri saklamak ve yönetmek için kullanılır.
- Değişkenler, program içinde verilere referans oluşturur.
- Bir değişkene değer atamak için, "değişkenAdı = ifade" kullanılır.
  - Örneğin, "pi = 3.14" pi değişkenine 3.14 değerini atar.
- Değişken adları bazı kurallara uymalıdır:
  - Bir kelime olmalıdır (boşluk içermez).
  - Harf, rakam ve alt çizgi (\_) içerebilir.
  - Rakamla başlayamaz.



# Değişken Tanımlama

- **Dinamik tipli** dillerde,
  - Değişkenlerin türünü önceden belirlemeye gerek yoktur.
  - Değişken kullanıldığında türü otomatik olarak belirlenir.
  - Değişkenin türü çalışma zamanında değiştirilebilir.
  - Bu, esneklik sağlar, ancak çalışma zamanı hatalarını artırabilir.
- **Statik tipli** dillerde,
  - Değişken türü açıkça belirtilmelidir.
  - Karakter dizisi ile tamsayı gibi uyumsuz işlemler derleme zamanında hata olarak tespit edilir.



# Atama İşlemi

- Bir değişkenin değerini değiştirmek için kullanılır.
- Atama işlemi iki adımda gerçekleşir:
  - Sağ taraf ifadesi hesaplanır.
  - Değişkenin içeriği hesaplanan yeni değer ile güncellenir.
- Örneğin,  $x = 5$  ifadesinde, sağ taraftaki 5 ifadesi değerlendirilir ve bu değer  $x$  adlı değişkende saklanır. Bu,  $x$  değişkeninin değerini 5 yapar.





# Koşullu İfadeler

- Programın hangi yolu izleyeceğini belirler.
- İfadenin sonucu doğru (True) veya yanlış (False) olabilir.
- Atama işlemi (örneğin,  $x = 100$ ), bir koşul ifadesi değildir.
- **Karşılaştırma operatörleri**, değerleri karşılaştırmak ve koşullu ifadeler oluşturmak için kullanılır. Örneğin,  $x \geq 5$ ,  $x$ 'in 5 veya daha büyük olup olmadığını kontrol eder.
- **Boolean operatörler**, koşulları birleştirmek veya tersine çevirmek için kullanılır. Örneğin, not True ifadesi True'in tersidir.
- **Karma operatörler**, sayıları veya ifadeleri karşılaştırmak için kullanılır. Örneğin,  $3 < 4$  AND  $5 < 6$ , hem  $3 < 4$  doğru hem de  $5 < 6$  doğru olduğunda doğru sonuç verir.



# Dizgeler (Strings)

- Metin verilerini temsil etmek için kullanılır.
- Boş bir dizge (""), bir atama yapılmamış değişkenle aynı değildir.
- Kaçış (escape) karakteri, özel karakterleri temsil etmek için kullanılır.
  - Örneğin, '\t' sekme karakterini, '\n' yeni satırı temsil eder.
- Uzunluğunu bulma, Birleştirme ve Arama gibi işlemler yapılır.
  - Örneğin, 'a' karakterinin dizge içinde olup olmadığını kontrol et.



# Farklı Türler Karşılaştırılmamalıdır

- Farklı veri türlerini karşılaştırmak hatalara neden olabilir.
  - Örneğin, tamsayı ile bir dizgeyi doğrudan karşılaştırmak hata verebilir.
- Programlama dilleri, veri türlerinin karşılaştırılması için uygun kuralları ve tür dönüşümünü tanımlar.



# Farklı Türler Üzerinde İşlemler Farklı Davranır

- İşlemler, farklı veri türleri üzerinde farklı şekilde davranır.
  - Dizgeleri toplama işlemi birleştirme (concatenation) olarak çalışır.
- Farklı türler arasında işlem yapmak hatalara neden olabilir.
- $3.0 + 4.0$  # Doğru
- $3 + 4$  # Doğru
- $3 + 4.0$  # ???
- $"3" + "4"$  # Birleştirme (Concatenation)
- $3 + "4"$  # Hata
- $3 + \text{True}$  # Hata



# Program Bir Tarifdir (A Program is a Recipe)

- Programlama, bir problemi çözüme sürecidir.
- Program, bu sürecin sonucunda ortaya çıkar.
- Program, bilgisayara bir görevi nasıl yerine getireceğini anlatan bir tarifdir.
- Programlar, bilgisayarlar tarafından anlaşılabilir bir şekilde yazılır.
- Programlar, adım adım talimatlar içerir.
- Bilgisayarlar, talimatlara uyarak işlemleri gerçekleştirir.



# Kodlamadan Önce Algoritmayı Tasarla

- Program yazılmadan önce, problemin nasıl çözüleceği düşünülür.
- Bir algoritma tasarlanır. Algoritma, mantıklı bir çözüm yolunu ifade eder.
- Algoritmik düşünme, bir problemi çözme ve işlemleri sıralama yeteneğidir.
- İyi bir algoritma, programın verimli bir şekilde çalışmasını sağlar.
- Kodlama, algoritmayı bilgisayarın anlayabileceği sözdizimine çevirir.
- Kodlama, tasarlanan algoritmayı gerçek bir program haline getirir.



# Bir Program Nedir?

- Bir program, bir dizi talimatı içerir.
- Bu talimatlar, birer birer işlenir ve sonuç olarak bir görev gerçekleştirilir.
- Bilgisayar, programın talimatlarını sırayla takip eder.
- Bu nedenle programların işlem sırasını belirlemek önemlidir.



# İfadeler, Deyimler ve Programlar

- Bir ifade, bir değeri hesaplar ve sonucunu döndürür.
  - Örneğin,  $3 + 4$  ifadesi 7 değerini hesaplar.
- Bir deyim bir etki yaratır.
  - Örneğin,  $\pi = 3.14159$  ifadesi,  $\pi$  adlı değişkene bir değer atar ve
  - `print(pi)` ifadesi, ekrana  $\pi$  değerini yazar.
- İfadeler, başka ifadelerin ve deyimlerin içinde görünebilir.
  - Bu, daha karmaşık hesaplama ve işlemlere izin verir.
- Bir deyim bir ifadenin içinde görünemez.
  - Örneğin,  $3 + \text{print}(\pi)$  ifadesi hata verir,
  - Çünkü `print` ifadesi bir etki yaratır, ancak bir değer döndürmez.
- Bir program, birçok deyim içerir.





# Popüler Programlama Dilleri

- Programlama dilleri, farklı görevler için farklı avantajlar sunar.
- Programcılar, projelerinin ihtiyaçlarına göre en uygun dili seçerler.
- Her yıl, programlama dillerinin popülerliği değerlendirilir.
- 2023'ün en öne çıkan dilleri;
  - Python, **Java**, JavaScript, C#, C++, Rust, Swift, Kotlin ve Go (Golang)



# Öne Çıkan Diller

- **Python:** Veri analizi, yapay zeka, web geliştirme gibi alanlarda yaygın
- **Java:** Büyük ölçekli uygulamalar ve Android uygulamaları için popüler
- **JavaScript:** Web tarayıcıları için önemli, web geliştirme için uygun
- **C#:** Windows uygulamaları ve oyun geliştirmek için kullanılır
- **C++:** Oyun, sistem programlama, performans odaklı uygulamalar için
- **Rust:** Güvenli ve hızlı sistem programlama için kullanılır
- **Swift:** Apple platformları üzerine uygulama geliştirmek için kullanılır
- **Kotlin:** Android uygulama geliştirme için popüler
- **Go (Golang):** Basit ve etkili bir dil, ağ uygulamaları için ideal



SON