Bölüm 6: Değişkenler, Veri Tipleri ve İşleçler

Bilgisayar ve Programlamaya Giriş

Değişken | Tip | Dönüşüm | İşleç | Girdi/Çıktı

Sercan KÜLCÜ · Giresun Üniversitesi

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü:
https://sercankulcu.github.io/files/computer/slides/Bolum_06_Degiskenler_Veri_Tipleri_ve_Islecler.html

GİRİŞ

Öğrenme Çıktıları

Değişken

Değişken bildirmek, ilk değer vermek ve güncellemek.

Tip

İlkel tipleri ve değer aralıklarını seçmek.

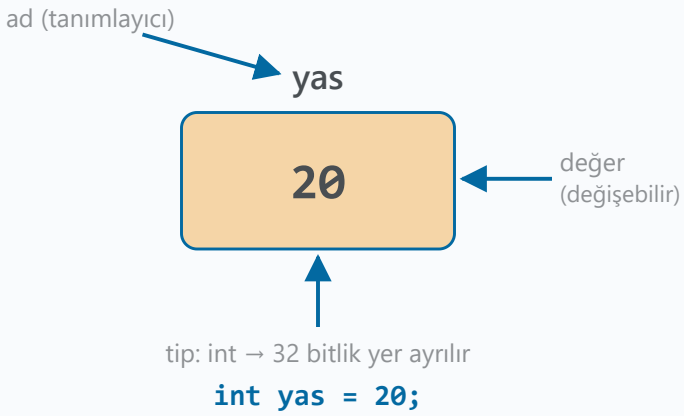
İfadeler

Atama, dönüşüm ve işleçleri doğru kullanmak.

Girdi/Çıktı

Scanner ile girdi almak, printf ile biçimli çıktı üretmek.

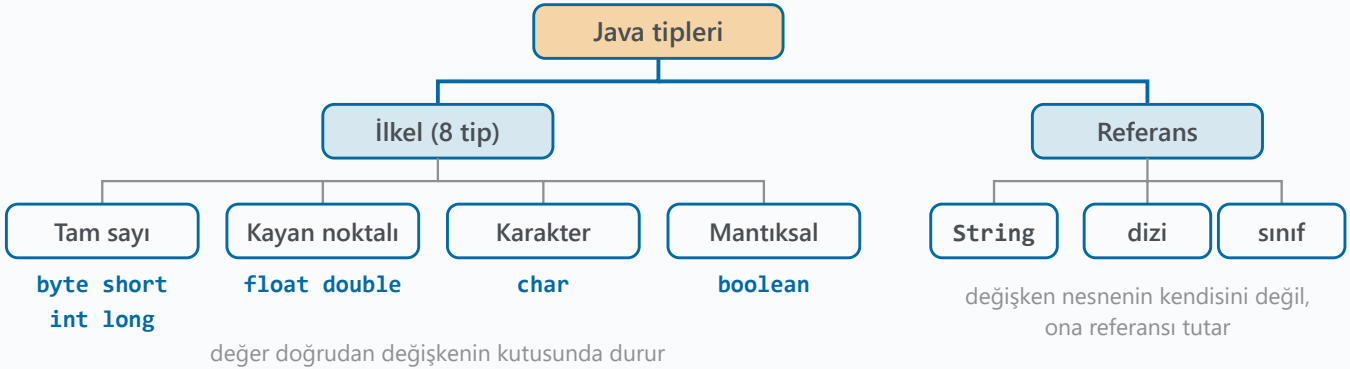
Değişkenler



- Değişken, belirli tipteki bir değere bellekte ayrılmış **adlı bir kutudur**.
- Bildirim tip ve adı; ilk değer verme başlangıç değerini belirtir.
- Tip, kutunun boyutunu ve içine konabilecek değerleri belirler.
- Yerel değişken kullanılmadan önce kesin olarak değer almış olmalıdır.

Temel Tip Grupları

Grup	Örnek	Kullanım
Tam sayı	byte, short, int, long	Sayma ve ayrık değer
Kayan noktalı	float, double	Ölçüm ve yaklaşık hesap
Karakter	char	Tek UTF-16 kod birimi
Mantıksal	boolean	true / false
Referans	String, dizi, sınıf	Nesneye referans

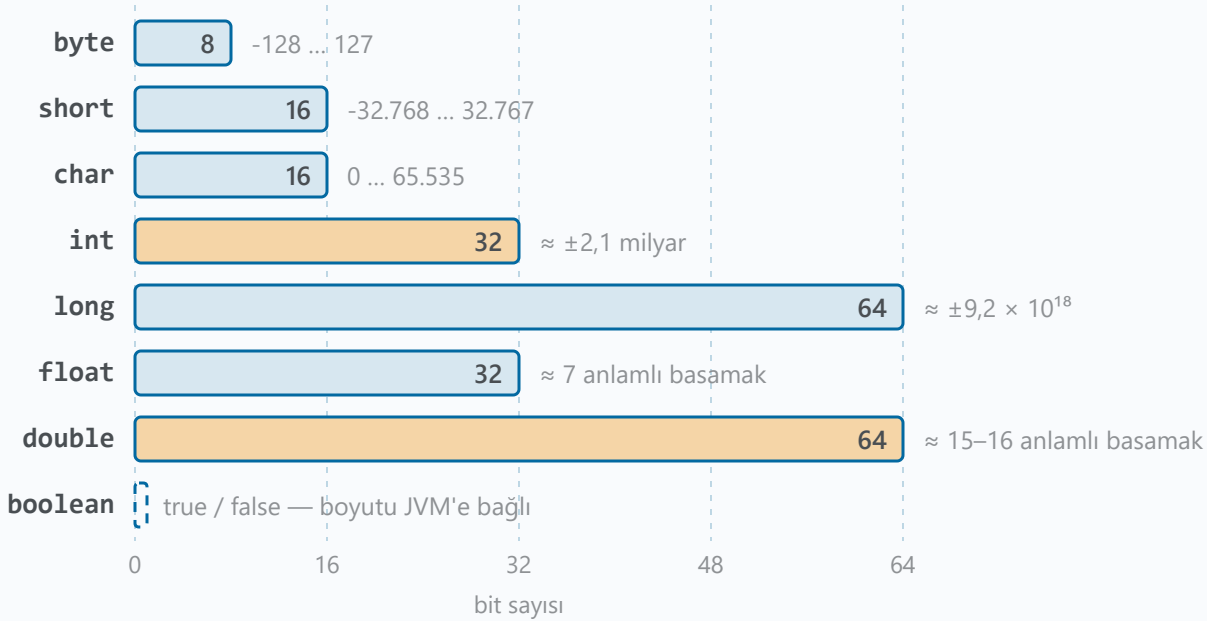


İlkel Veri Tipleri

Tip	Yaklaşık kullanım	Örnek
byte / short	Küçük tam sayılar	byte seviye = 7;
int	Genel tam sayı	int adet = 42;
long	Büyük tam sayı	long nufus = 8_000_000L;
float	Kayan noktalı, düşük hassasiyet	float oran = 1.5F;
double	Genel kayan noktalı sayı	double fiyat = 19.90;
char / boolean	Karakter ve mantık	char not='A';

VERİ TİPLERİ

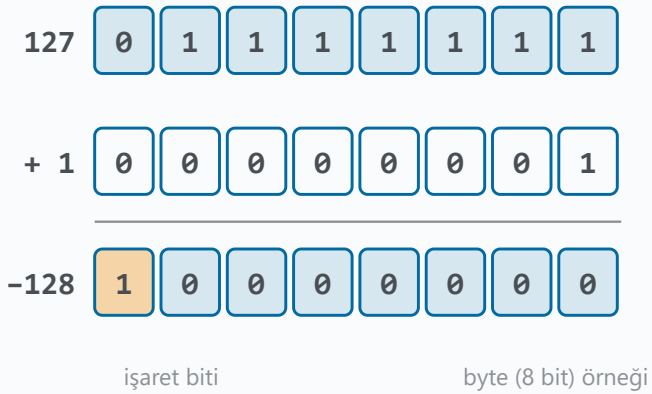
Tip Boyutları (bit)



Vurgulu: varsayılan seçimler (int, double). 1 bayt = 8 bit; boyutlar her platformda aynıdır.

Tam Sayı Tipi Seçimi

- int çoğu tam sayı hesabı için varsayılan seçimdir; long sabitleri **L** sonekiyle yazılır.
- Aralığı aşan tam sayı işlemi **taşar**: Java hata vermez, değer sessizce öbür uca sarar.
- `Integer.MAX_VALUE + 1` → `-2147483648`.
- Büyük değerler bekleniyorsa baştan long seçilir.



Taşma: `Integer.MAX_VALUE + 1`



int: 32 bit. En büyük değer 2.147.483.647 ($2^{31} - 1$)

Integer.MAX_VALUE: işaret biti 0, diğer 31 bitin hepsi 1. int için ulaşılabilecek **en büyük** sayı.

Adım 1 / 6

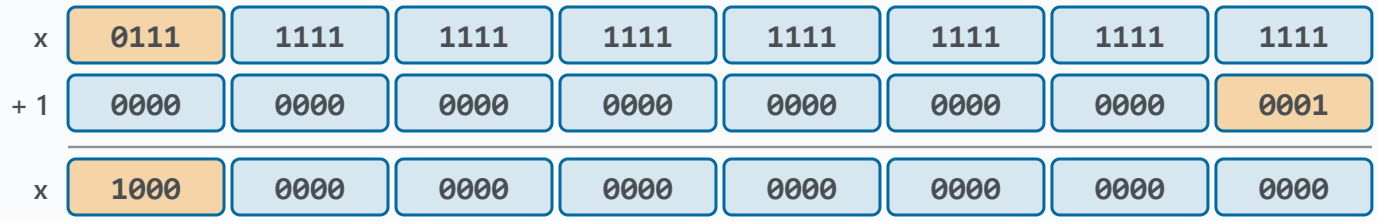
```
▶ int x = Integer.MAX_VALUE;
2 x++;
3 byte b = 127;
4 b++;
5 System.out.println(x + " " + b);
```

DEĞİŞKENLER

x = 2147483647

ÇIKTI

Taşma: `Integer.MAX_VALUE + 1`



x = −2.147.483.648 (en küçük int)

x++ ile 1 eklenir: bitler taşar ve **işaret biti 1** olur. Sonuç hata değil, **−2147483648**: sayı çember gibi başa sarar.

Adım 2 / 6

```
1 int x = Integer.MAX_VALUE;
▶ x++;
3 byte b = 127;
4 b++;
5 System.out.println(x + " " + b);
```

DEĞİŞKENLER

x = −2147483648

ÇIKTI

Taşma: `Integer.MAX_VALUE + 1`



byte 8 bitliktir: en büyük değeri 127 (0111 1111).

Adım 3 / 6

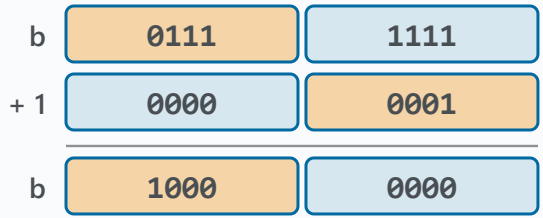
```
1 int x = Integer.MAX_VALUE;
2 x++;
▶ byte b = 127;
4 b++;
5 System.out.println(x + " " + b);
```

DEĞİŞKENLER

`x = -2147483648` `b = 127`

ÇIKTI

Taşma: `Integer.MAX_VALUE + 1`



b = −128 (en küçük byte)

b++ ile 127 + 1: bitler 1000 0000 olur → **−128**. Aynı sarma, sadece daha küçük kutuda.

Adım 4 / 6

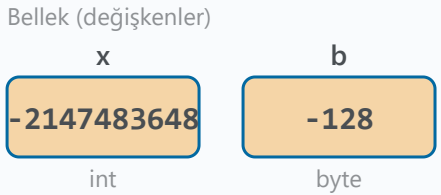
```
1 int x = Integer.MAX_VALUE;
2 x++;
3 byte b = 127;
▶ b++;
5 System.out.println(x + " " + b);
```

DEĞİŞKENLER

x = −2147483648 b = −128

ÇIKTI

Taşma: `Integer.MAX_VALUE + 1`



Çıktı: **-2147483648 -128**. İki değişken de en büyük değerden en küçüğe sardı.

Adım 5 / 6

```
1 int x = Integer.MAX_VALUE;
2 x++;
3 byte b = 127;
4 b++;
▶ System.out.println(x + " " + b);
```

DEĞİŞKENLER

`x = -2147483648` `b = -128`

ÇIKTI

```
-2147483648 -128
```

Taşma: `Integer.MAX_VALUE + 1`

Java taşmada hata vermez, sessizce sarar!

`Math.addExact(x, 1)`

→ `ArithmeticException: integer overflow`

Taşma ne derleme hatası ne de çalışma zamanı hatasıdır. Taşma istenmiyorsa `Math.addExact` gibi metotlar istisna fırlatır.

Adım 6 / 6

```
1 int x = Integer.MAX_VALUE;
2 x++;
3 byte b = 127;
4 b++;
▶ System.out.println(x + " " + b);
```

DEĞİŞKENLER

`x = -2147483648` `b = -128`

ÇIKTI

```
-2147483648 -128
```

Sınır Sabitleri ve Math.addExact

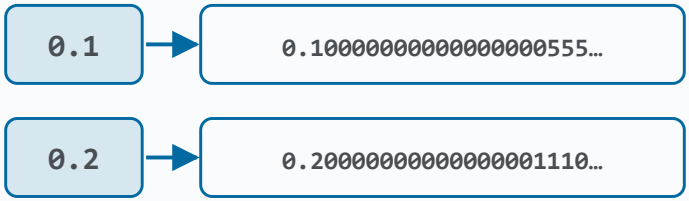
Sabit	Değer
`Integer.MIN_VALUE`	`-2147483648`
`Integer.MAX_VALUE`	`2147483647`
`Long.MIN_VALUE`	`-9223372036854775808`
`Long.MAX_VALUE`	`9223372036854775807`

```
int x = Integer.MAX_VALUE;
System.out.println(x + 1);
// -2147483648 (sessiz taşma)
System.out.println(
    Math.addExact(x, 1));
// ArithmeticException:
// integer overflow
```

- Sabitler elle yazılmaz: `Integer.MAX_VALUE` hem okunaklıdır hem hatasızdır.
- `Math.addExact` taşmayı yakalar: int için **integer overflow**, long için **long overflow** iletisiyle istisna fırlatır.
- `Math.abs(Integer.MIN_VALUE)` de taşar ve yine **-2147483648** verir.

Kayan Noktalı Sayılar

- double çoğu yaklaşık hesap için varsayılan tiptir; float sabiti **F** soneki ister.
- İkili kayan nokta bazı ondalık değerleri (0.1 gibi) tam temsil edemez.
- Para gibi kesin ondalık hesaplarda uygun sınıf (BigDecimal) ve yuvarlama kuralı seçilmelidir.



bellekte saklanan en yakın ikili değer

`0.1 + 0.2 → 0.30000000000000004`

bu yüzden double değerler == ile karşılaştırılmaz

Kayan Noktalı Sayı Karşılaştırma

0.1 ve 0.2 ikili tabanda tam gösterilemez; toplam 0.3 değil, ona çok yakın bir değerdir.

Adım 1 / 4

```
▶ double a = 0.1 + 0.2;
2 System.out.println(a);
3 System.out.println(a == 0.3);
4 System.out.println(Math.abs(a - 0.3) < 1e-9);
```

DEĞİŞKENLER

a = 0.30000000000000004

ÇIKTI

Kayan Noktalı Sayı Karşılaştırma

Yazdırınca küçük hata görünür: **0.30000000000000004**.

Adım 2 / 4

```
1 double a = 0.1 + 0.2;
▶ System.out.println(a);
3 System.out.println(a == 0.3);
4 System.out.println(Math.abs(a - 0.3) < 1e-9);
```

DEĞİŞKENLER

a = 0.30000000000000004

ÇIKTI

```
0.30000000000000004
```

Kayan Noktalı Sayı Karşılaştırma

`a == 0.3` → **false**. Aynı sonuç `0.1 + 0.2 == 0.3` için de geçerlidir. Kayan noktalı sayılar `==` ile karşılaştırılmaz.

Adım 3 / 4

```
1 double a = 0.1 + 0.2;
2 System.out.println(a);
▶ System.out.println(a == 0.3);
4 System.out.println(Math.abs(a - 0.3) < 1e-9);
```

DEĞİŞKENLER

`a = 0.30000000000000004`

ÇIKTI

```
0.30000000000000004
false
```

Kayan Noktalı Sayı Karşılaştırma

Fark yaklaşık 5.55E-17 olduğu için `Math.abs(a - 0.3) < 1e-9` → **true**. Fark küçük bir eşikle (epsilon) sınanır.

Adım 4 / 4

```
1 double a = 0.1 + 0.2;
2 System.out.println(a);
3 System.out.println(a == 0.3);
▶ System.out.println(Math.abs(a - 0.3) < 1e-9);
```

DEĞİŞKENLER

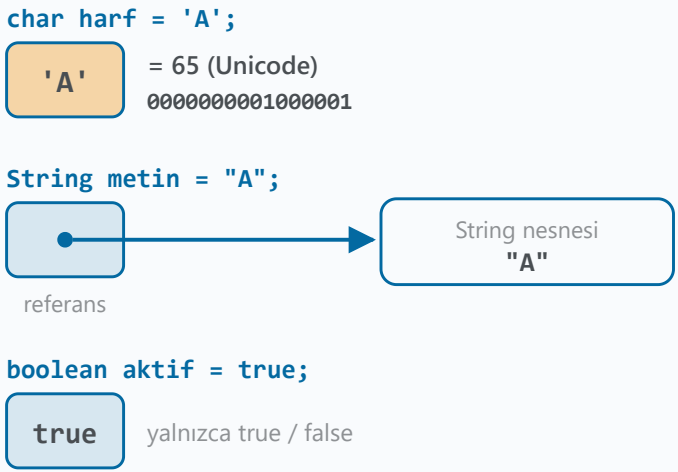
`a = 0.30000000000000004`

ÇIKTI

```
0.30000000000000004
false
true
```

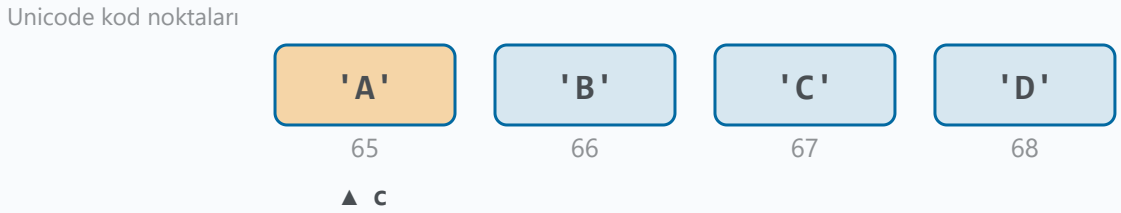
char ve boolean

- char tek tırnakla yazılır; bir UTF-16 kod birimini (sayı olarak) saklar: 'A' = 65.
- String çift tırnakla yazılan bir **nesnedir**; değişken nesneye referans tutar. char ile aynı tip değildir.
- boolean yalnızca true veya false değerini alır; 0/1 ile yer değiştirmez.



VERİ · KODU ADIM ADIM ÇALIŞTIR

char Aritmetiği: Harfler Sayıdır



c = 'A'. char aslında bir **sayıdır**: 'A' için Unicode kod noktası 65.

Adım 1 / 5

```
▶ char c = 'A';
2 c++;
3 int k = c + 1;
4 char d = (char)(c + 1);
5 System.out.println(c + " " + k + " " + d);
```

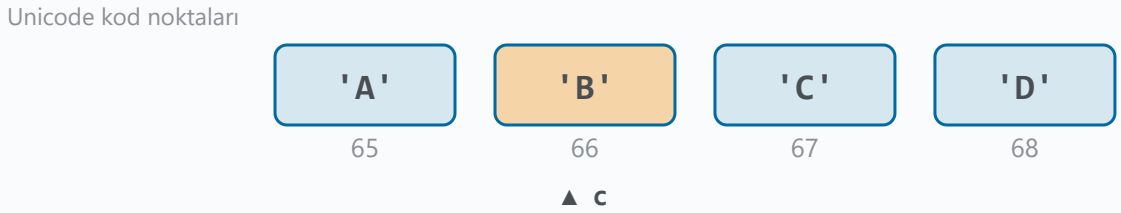
DEĞİŞKENLER

c = 'A' (65)

ÇIKTI

VERİ · KODU ADIM ADIM ÇALIŞTIR

char Aritmetiği: Harfler Sayıdır



c++ kod noktasını 1 artırır: 65 → 66, yani c = **'B'**. Tip hâlâ char.

Adım 2 / 5

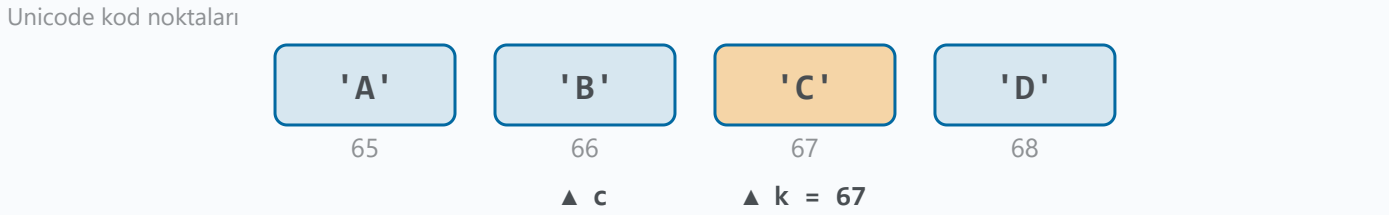
```
1 char c = 'A';
▶ c++;
3 int k = c + 1;
4 char d = (char)(c + 1);
5 System.out.println(c + " " + k + " " + d);
```

DEĞİŞKENLER

c = 'B' (66)

ÇIKTI

char Aritmetiği: Harfler Sayıdır



c + 1: c int'e yükselir, 66 + 1 = 67 (int)

`c + 1` ifadesi **int** olur (66 + 1 = 67). k bir int olduğundan 67 sayısını tutar, harf değil.

Adım 3 / 5

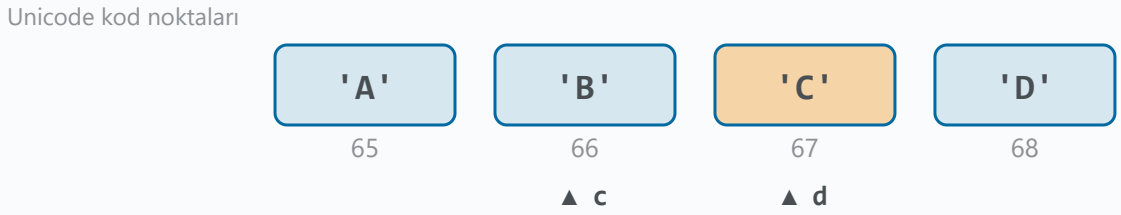
```
1 char c = 'A';
2 c++;
▶ int k = c + 1;
4 char d = (char)(c + 1);
5 System.out.println(c + " " + k + " " + d);
```

DEĞİŞKENLER

`c = 'B' (66)` `k = 67`

ÇIKTI

char Aritmetiği: Harfler Sayıdır



(char)(c + 1): 67 tekrar char'a çevrilir → 'C'

Sonucu harf olarak saklamak için **(char)** dönüşümü gerekir: d = 'C'. Dönüşüm olmazsa `char d = c + 1;` derlenmez.

Adım 4 / 5

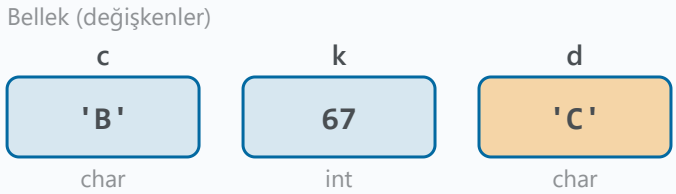
```
1 char c = 'A';
2 c++;
3 int k = c + 1;
▶ char d = (char)(c + 1);
5 System.out.println(c + " " + k + " " + d);
```

DEĞİŞKENLER

`c = 'B' (66)` `k = 67` `d = 'C' (67)`

ÇIKTI

char Aritmetiği: Harfler Sayıdır



Çıktı: **B 67 C**. char yazdırılırken harf, int yazdırılırken sayı görünür.

Adım 5 / 5

```
1 char c = 'A';
2 c++;
3 int k = c + 1;
4 char d = (char)(c + 1);
▶ System.out.println(c + " " + k + " " + d);
```

DEĞİŞKENLER

`c = 'B' (66)` `k = 67` `d = 'C' (67)`

ÇIKTI

```
B 67 C
```

Tanımlayıcı Kuralları

Kural	Geçerli	Geçersiz
Harf, _ veya $ ile başlar	ogrenci1	1ogrenci
Boşluk içermez	toplamPuan	toplam puan
Anahtar sözcük değildir	sinifNo	class
Tek alt çizgi ad olamaz	_deger	_
Büyük-küçük harfe duyarlı	puan ve Puan	Aynı değişken sanmak

Adlandırma

- Değişken ve metot adlarında lowerCamelCase kullanılır.
- Sınıf adları UpperCamelCase biçimindedir.
- Sabitler genellikle BÜYÜK_HARF_VE_ALT_ÇİZGİ ile adlandırılır.
- Ad, değerin rolünü açıklamalıdır: x yerine toplamPuan.

lowerCamelCase
değişken ve metot

toplam | Puan

```
int toplamPuan = 0;
```

UpperCamelCase
sınıf

Ogrenci | Kaydi

```
class OgrenciKaydi { }
```

BÜYÜK_HARF
sabit (final)

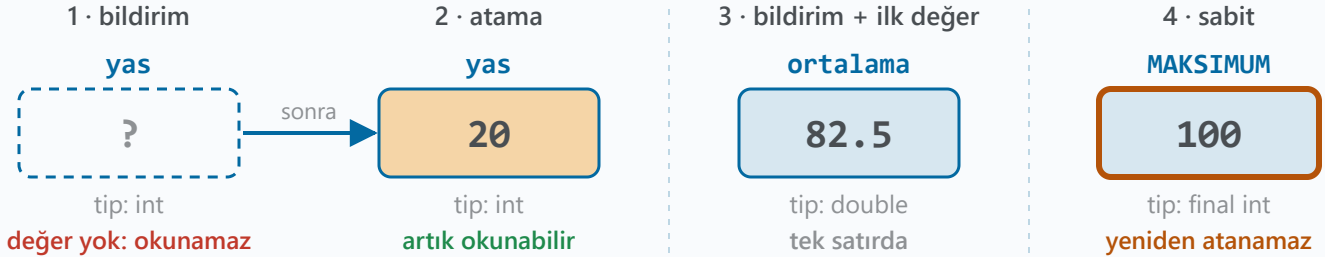
MAKSIMUM | _ | PUAN

```
final int MAKSIMUM_PUAN = 100;
```

Rolü anlat: `x` ✗ → `toplamPuan` ✓ Kelime sınırını büyük harf (ya da sabitte _) gösterir.

Bildirim ve İlk Değer

```java
int yas;           // bildirim
yas = 20;          // atama
double ortalama = 82.5; // birlikte
final int MAKSIMUM = 100; // sabit
```



- Yerel değişken okunmadan önce değer almalıdır.
- final değişkene yalnızca bir kez atama yapılır.

Değişkenler Bellekte Adım Adım



Bildirim: int tipinde **yas** kutusu ayrılır ama değeri yok. Bu hâliyle okunursa derleyici hata verir.

Adım 1 / 10

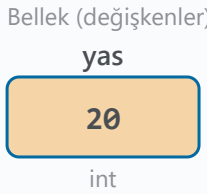
```
▶ int yas;                    // bildirim
2 yas = 20;                   // ilk atama
3 yas = yas + 1;              // güncelleme
4 double ortalama = 82.5;     // bildirim + ilk
  değer
5 ortalama += 5;              // bileşik atama
6 final int MAKSIMUM = 100; // sabit
7 System.out.println(yas);
8 System.out.println(ortalama);
```

DEĞİŞKENLER

`yas = ? (atanmadı)`

ÇIKTI

Değişkenler Bellekte Adım Adım



İlk atama: sağdaki 20 değeri yas kutusuna yazılır.

Adım 2 / 10

```
1 int yas;                    // bildirim
▶ yas = 20;                   // ilk atama
3 yas = yas + 1;              // güncelleme
4 double ortalama = 82.5;     // bildirim + ilk
  değer
5 ortalama += 5;              // bileşik atama
6 final int MAKSIMUM = 100; // sabit
7 System.out.println(yas);
8 System.out.println(ortalama);
```

DEĞİŞKENLER

`yas = 20`

ÇIKTI

Değişkenler Bellekte Adım Adım



Önce **sağ taraf** hesaplanır: yas + 1 → 20 + 1 → 21. Kutu henüz değişmedi.

Adım 3 / 10

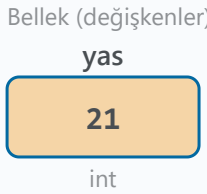
```
1 int yas;                    // bildirim
2 yas = 20;                   // ilk atama
▶ yas = yas + 1;              // güncelleme
4 double ortalama = 82.5;     // bildirim + ilk
  değer
5 ortalama += 5;              // bileşik atama
6 final int MAKSIMUM = 100; // sabit
7 System.out.println(yas);
8 System.out.println(ortalama);
```

DEĞİŞKENLER

yas = **20**

ÇIKTI

Değişkenler Bellekte Adım Adım



Sonuç aynı kutuya yazılır: eski 20 silinir, **yas = 21**.

Adım 4 / 10

```
1 int yas;                    // bildirim
2 yas = 20;                   // ilk atama
▶ yas = yas + 1;              // güncelleme
4 double ortalama = 82.5;     // bildirim + ilk
  değer
5 ortalama += 5;              // bileşik atama
6 final int MAKSIMUM = 100; // sabit
7 System.out.println(yas);
8 System.out.println(ortalama);
```

DEĞİŞKENLER

`yas = 21`

ÇIKTI

Değişkenler Bellekte Adım Adım



double tipinde **ortalama** bildirilir ve aynı deyimde 82.5 atanır.

Adım 5 / 10

```
1 int yas;                    // bildirim
2 yas = 20;                   // ilk atama
3 yas = yas + 1;              // güncelleme
▶ double ortalama = 82.5;     // bildirim + ilk değer
5 ortalama += 5;              // bileşik atama
6 final int MAKSIMUM = 100;   // sabit
7 System.out.println(yas);
8 System.out.println(ortalama);
```

DEĞİŞKENLER

`yas = 21` `ortalama = 82.5`

ÇIKTI

Değişkenler Bellekte Adım Adım

Bellek (değişkenler)

yas	ortalama
21	82.5
int	double

İfade

```
ortalama + 5
82.5 + 5
→ 87.5
```

`ortalama += 5` , `ortalama = ortalama + 5` demektir: 82.5 + 5 → 87.5.

Adım 6 / 10

```
1 int yas;                    // bildirim
2 yas = 20;                   // ilk atama
3 yas = yas + 1;              // güncelleme
4 double ortalama = 82.5;     // bildirim + ilk
  değer
▶ ortalama += 5;              // bileşik atama
6 final int MAKSIMUM = 100; // sabit
7 System.out.println(yas);
8 System.out.println(ortalama);
```

DEĞİŞKENLER

`yas = 21` `ortalama = 82.5`

ÇIKTI

Değişkenler Bellekte Adım Adım



Kutu güncellenir: **ortalama = 87.5**.

Adım 7 / 10

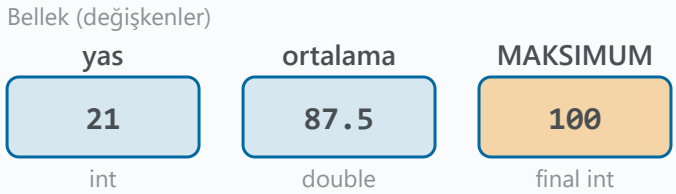
```
1 int yas;                    // bildirim
2 yas = 20;                   // ilk atama
3 yas = yas + 1;              // güncelleme
4 double ortalama = 82.5;     // bildirim + ilk
  değer
▶ ortalama += 5;              // bileşik atama
6 final int MAKSIMUM = 100; // sabit
7 System.out.println(yas);
8 System.out.println(ortalama);
```

DEĞİŞKENLER

`yas = 21` `ortalama = 87.5`

ÇIKTI

Değişkenler Bellekte Adım Adım



final sabit: 100 bir kez atanır. `MAKSIMUM = 200;` yazılsaydı derleme hatası olurdu.

Adım 8 / 10

```
1 int yas;                    // bildirim
2 yas = 20;                   // ilk atama
3 yas = yas + 1;              // güncelleme
4 double ortalama = 82.5;     // bildirim + ilk
  değer
5 ortalama += 5;              // bileşik atama
▶ final int MAKSIMUM = 100; // sabit
7 System.out.println(yas);
8 System.out.println(ortalama);
```

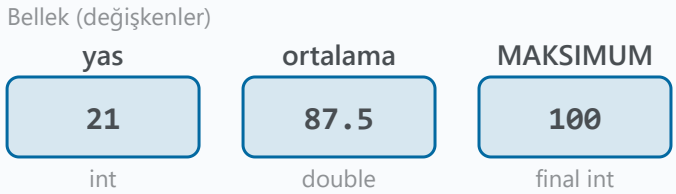
DEĞİŞKENLER

`yas = 21` `ortalama = 87.5`

`MAKSIMUM = 100`

ÇIKTI

Değişkenler Bellekte Adım Adım



yas kutusu **okunur** ve 21 yazdırılır. Okuma değeri değiştirmez.

Adım 9 / 10

```
1 int yas;                    // bildirim
2 yas = 20;                   // ilk atama
3 yas = yas + 1;              // güncelleme
4 double ortalama = 82.5;     // bildirim + ilk
  değer
5 ortalama += 5;              // bileşik atama
6 final int MAKSIMUM = 100; // sabit
▶ System.out.println(yas);
8 System.out.println(ortalama);
```

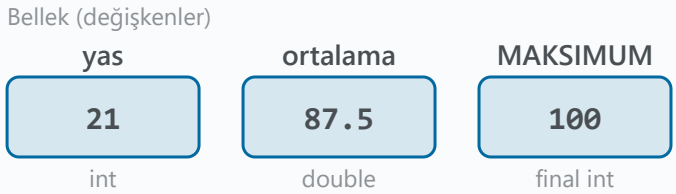
DEĞİŞKENLER

`yas = 21` `ortalama = 87.5` `MAKSIMUM = 100`

ÇIKTI

```
21
```

Değişkenler Bellekte Adım Adım



ortalama okunur: 87.5. Son durum: 3 kutu, 2 satır çıktı.

Adım 10 / 10

```
1 int yas;                    // bildirim
2 yas = 20;                   // ilk atama
3 yas = yas + 1;              // güncelleme
4 double ortalama = 82.5;     // bildirim + ilk
  değer
5 ortalama += 5;              // bileşik atama
6 final int MAKSIMUM = 100; // sabit
7 System.out.println(yas);
▶ System.out.println(ortalama);
```

DEĞİŞKENLER

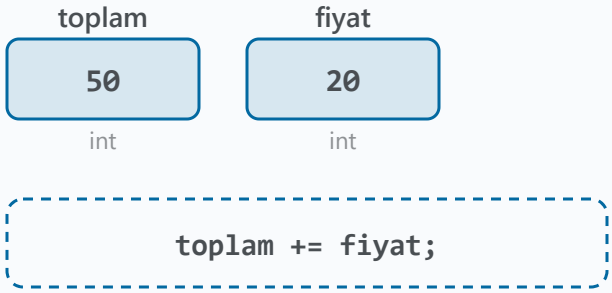
`yas = 21` `ortalama = 87.5` `MAKSIMUM = 100`

ÇIKTI

```
21
87.5
```

Atama ve Güncelleme

- Atama işleci `=` önce sağ tarafı hesaplar, sonra değeri soldaki değişkene yazar.
- Yeni değer değişkenin tipiyle uyumlu olmalıdır.
- Bileşik atama mevcut değeri kullanarak günceller: `toplam += fiyat` ≡ `toplam = toplam + fiyat`.

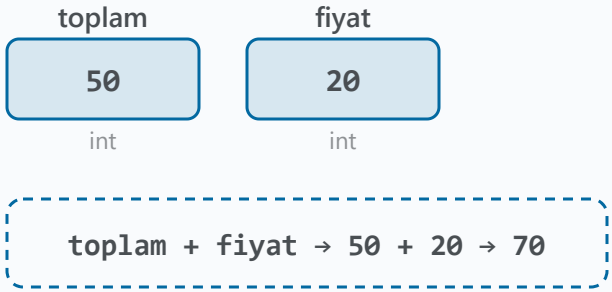


Başlangıç: toplam = 50, fiyat = 20.

Adım 1 / 3

Atama ve Güncelleme

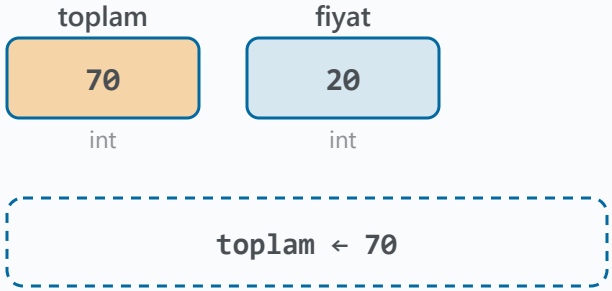
- Atama işleci `=` önce sağ tarafı hesaplar, sonra değeri soldaki değişkene yazar.
- Yeni değer değişkenin tipiyle uyumlu olmalıdır.
- Bileşik atama mevcut değeri kullanarak günceller: `toplam += fiyat` ≡ `toplam = toplam + fiyat`.



Önce iki kutu **okunur** ve sağ taraf hesaplanır.

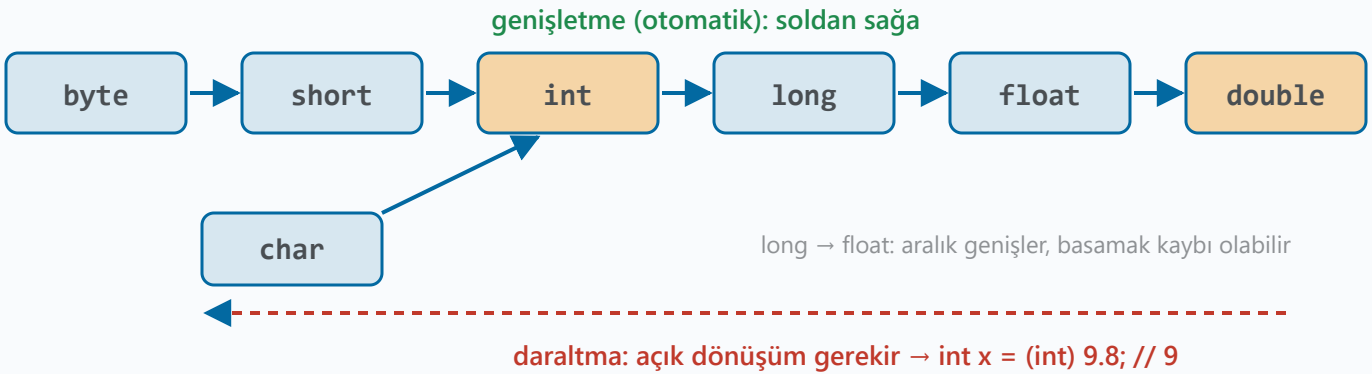
Atama ve Güncelleme

- Atama işleci `=` önce sağ tarafı hesaplar, sonra değeri soldaki değişkene yazar.
- Yeni değer değişkenin tipiyle uyumlu olmalıdır.
- Bileşik atama mevcut değeri kullanarak günceller: `toplam += fiyat` ≡ `toplam = toplam + fiyat`.



Sonuç soldaki kutuya yazılır: toplam = 70. fiyat değişmez.

Atama Uyumluluğu

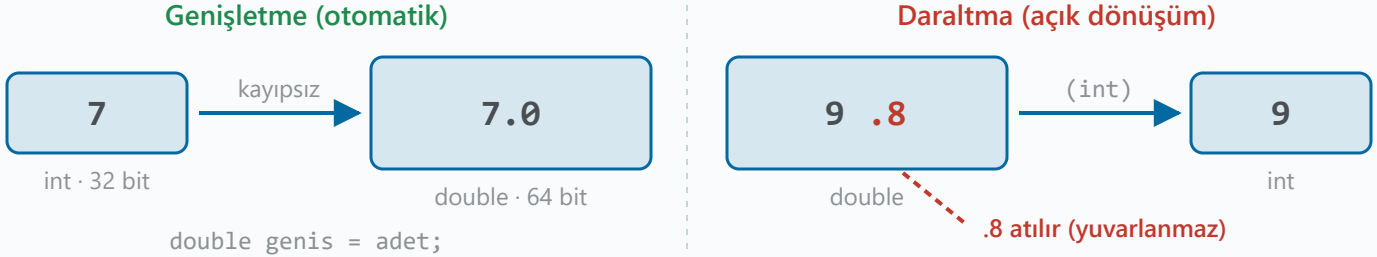


- Dar tipten geniş tipe dönüşüm (genişletme) otomatik yapılır: `double d = 7;`
- Daraltma bilgi kaybı riski taşıdığı için açık dönüşüm (cast) ister.
- Dönüşüm kaynak değişkeni değiştirmez; yalnızca üretilen sonucu etkiler.

DÖNÜŞÜM

Tip Dönüşümü

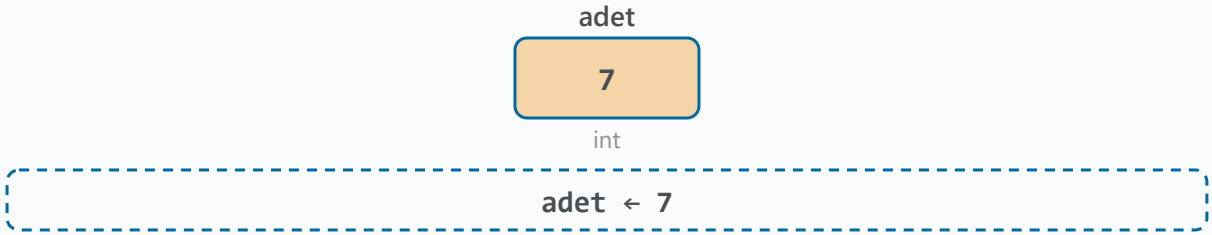
```
int adet = 7;
double genis = adet;        // 7.0
double oran = 9.8;
int dar = (int) oran;       // 9
int bolum = 5 / 2;          // 2
double dogru = 5.0 / 2;     // 2.5
```



- double değerini int'e çevirmek kesir bölümünü atar.
- İşlemden sonra yapılan dönüşüm kaybolan kesri geri getirmez.

Tip Dönüşümü Adım Adım

Bellek (değişkenler)



adet: int kutusu (32 bit), değer 7.

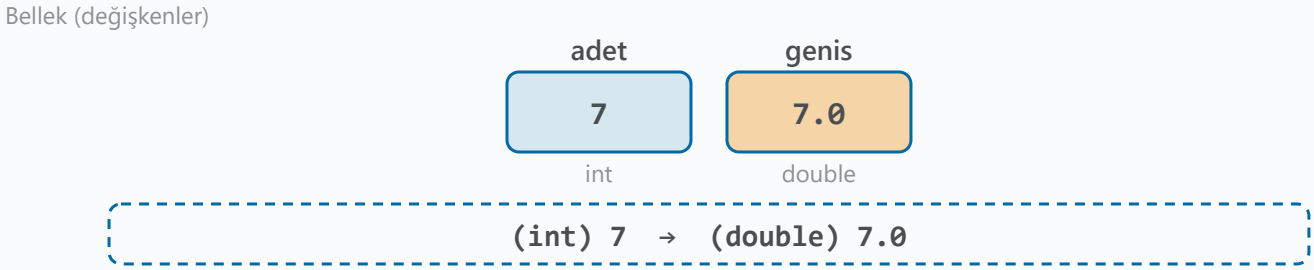
Adım 1 / 6

```
▶ int adet = 7;
2 double genis = adet;        // 7.0
3 double oran = 9.8;
4 int dar = (int) oran;       // 9
5 int bolum = 5 / 2;          // 2
6 double dogru = 5.0 / 2;     // 2.5
```

DEĞİŞKENLER

adet = **7 (int)**

Tip Dönüşümü Adım Adım



Genişletme: int değer double kutuya otomatik dönüşür: 7 → 7.0. Bilgi kaybı yok.

Adım 2 / 6

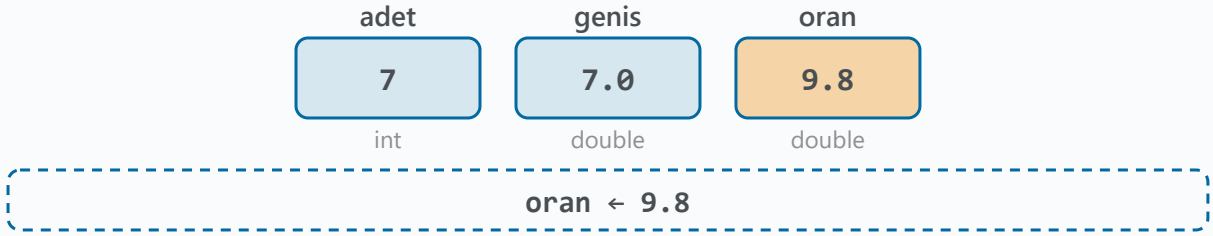
```
1 int adet = 7;
▶ double genis = adet;        // 7.0
3 double oran = 9.8;
4 int dar = (int) oran;       // 9
5 int bolum = 5 / 2;          // 2
6 double dogru = 5.0 / 2;     // 2.5
```

DEĞİŞKENLER

adet = 7 (int) genis = 7.0 (double)

Tip Dönüşümü Adım Adım

Bellek (değişkenler)



oran: double kutusu (64 bit), değer 9.8.

Adım 3 / 6

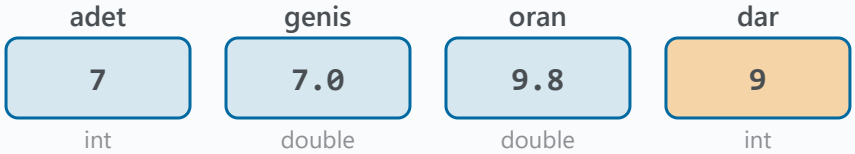
```
1 int adet = 7;
2 double genis = adet;        // 7.0
▶ double oran = 9.8;
4 int dar = (int) oran;       // 9
5 int bolum = 5 / 2;          // 2
6 double dogru = 5.0 / 2;     // 2.5
```

DEĞİŞKENLER

adet = 7 (int) genis = 7.0 (double) oran = 9.8 (double)

Tip Dönüşümü Adım Adım

Bellek (değişkenler)



Daraltma: `(int)` açıkça yazılmalı. Kesir **atılır** (yuvarlanmaz): 9.8 → 9. oran hâlâ 9.8.

Adım 4 / 6

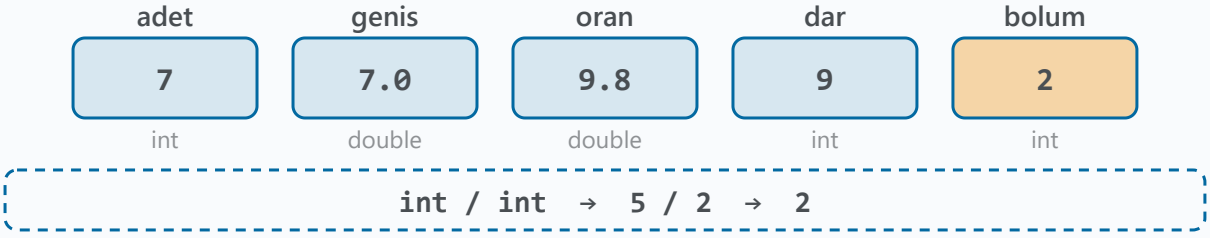
```
1 int adet = 7;
2 double genis = adet;        // 7.0
3 double oran = 9.8;
▶ int dar = (int) oran;       // 9
5 int bolum = 5 / 2;          // 2
6 double dogru = 5.0 / 2;     // 2.5
```

DEĞİŞKENLER

`adet = 7 (int)` `genis = 7.0 (double)` `oran = 9.8 (double)` `dar = 9 (int)`

Tip Dönüşümü Adım Adım

Bellek (değişkenler)



5 ve 2 ikisi de int → **tam sayı bölmesi**: 5 / 2 = 2, kalan atılır.

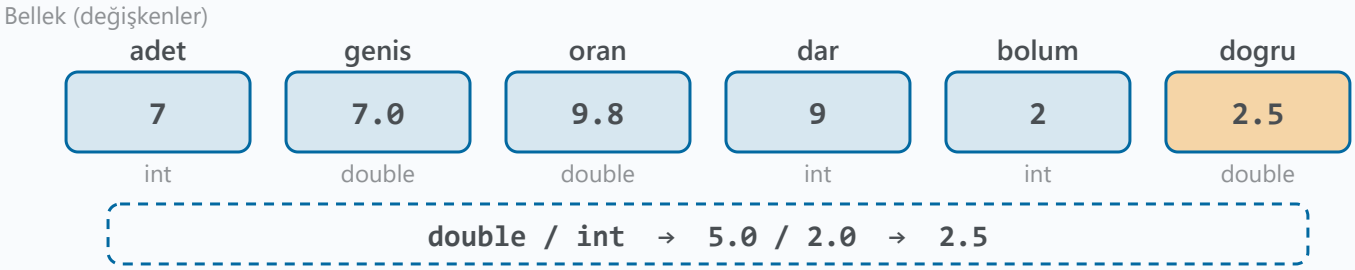
Adım 5 / 6

```
1 int adet = 7;
2 double genis = adet;        // 7.0
3 double oran = 9.8;
4 int dar = (int) oran;       // 9
▶ int bolum = 5 / 2;          // 2
6 double dogru = 5.0 / 2;     // 2.5
```

DEĞİŞKENLER

adet = **7 (int)** genis = **7.0 (double)** oran = **9.8 (double)** dar = **9 (int)** bolum = **2 (int)**

Tip Dönüşümü Adım Adım



5.0 double olduğu için 2 de 2.0'a genişler: 5.0 / 2.0 = **2.5**.

Adım 6 / 6

```
1 int adet = 7;
2 double genis = adet;        // 7.0
3 double oran = 9.8;
4 int dar = (int) oran;       // 9
5 int bolum = 5 / 2;          // 2
▶ double dogru = 5.0 / 2;     // 2.5
```

DEĞİŞKENLER

`adet = 7 (int)` `genis = 7.0 (double)` `oran = 9.8 (double)` `dar = 9 (int)` `bolum = 2 (int)` `dogru = 2.5 (double)`

DÖNÜŞÜM

Dönüşümlerde Taşma ve Kesme

İfade	Sonuç	Neden
`(byte) 200`	`-56`	sadece alt 8 bit kalır: 200 - 256
`(int) 3000000000L`	`-1294967296`	long → int: üst bitler atılır
`(int) 3.9`	`3`	ondalık kısım kesilir, yuvarlanmaz
`(int) -2.7`	`-2`	sıfıra doğru kesilir
`(int) 3.9e10`	`2147483647`	double → int: int sınırında sıkışır
`Math.round(3.9)`	`4`	en yakın tam sayı (long döner)
`Math.round(-2.7)`	`-3`	en yakın tam sayı
`Math.round(3.9e10)`	`39000000000`	long olduğu için sığar

Yuvarlamak için `Math.round`, int istiyorsan `(int) Math.round(x)`. Daraltma dönüşümü hata vermez; veri sessizce kaybolur.

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



int / int → int

İki işlenen de **int** → tam sayı bölmesi: bölüm alınır, kesir atılır.

Adım 1 / 14

```
▶ System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

—

ÇIKTI

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



Sonuç **2** (int) yazdırılır.

Adım 2 / 14

```
▶ System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

`sonuc = 2`

ÇIKTI

```
2
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



int işlenen double'a genişler

Bir işlenen double → diğeri de double'a **yükseltilir** (2 → 2.0). Kayan noktalı bölme yapılır.

Adım 3 / 14

```
1 System.out.println(5 / 2);
▶ System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

—

ÇIKTI

```
2
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



Sonuç **2.5** (double) yazdırılır.

Adım 4 / 14

```
1 System.out.println(5 / 2);
▶ System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

`sonuc = 2.5`

ÇIKTI

```
2
2.5
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



Bir işlenen double → diğeri de double'a **yükseltilir** (5 → 5.0). Kayan noktalı bölme yapılır.

Adım 5 / 14

```
1 System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
▶ System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

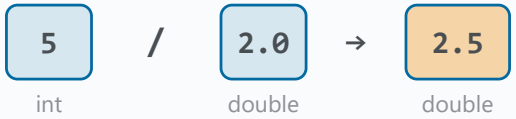
—

ÇIKTI

```
2
2.5
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



Sonuç **2.5** (double) yazdırılır.

Adım 6 / 14

```
1 System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
▶ System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

`sonuc = 2.5`

ÇIKTI

```
2
2.5
2.5
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



Tip dönüşümü `/`'dan önce gelir: (double) 5 → **5.0**.

Adım 7 / 14

```
1 System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
▶ System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

—

ÇIKTI

```
2
2.5
2.5
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



int işlenen double'a genişler

Bir işlenen double → diğeri de double'a **yükseltilir** (2 → 2.0). Kayan noktalı bölme yapılır.

Adım 8 / 14

```
1 System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
▶ System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

—

ÇIKTI

```
2
2.5
2.5
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



Sonuç **2.5** (double) yazdırılır.

Adım 9 / 14

```
1 System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
▶ System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

`sonuc = 2.5`

ÇIKTI

```
2
2.5
2.5
2.5
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



Parantez içi önce: 5 / 2, ikisi de int → tam sayı bölmesi = **2**. Kesir burada kaybolur.

Adım 10 / 14

```
1 System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
▶ System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

—

ÇIKTI

```
2
2.5
2.5
2.5
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



Sonra `(double) 2` → **2.0**. Kaybolan .5 geri gelmez.

Adım 11 / 14

```
1 System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
▶ System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

`sonuc = 2.0`

ÇIKTI

```
2
2.5
2.5
2.5
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



2.0 yazdırılır: işlemden **sonra** yapılan dönüşüm kesri kurtarmaz.

Adım 12 / 14

```
1 System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
▶ System.out.println((double) (5 / 2));
6 System.out.println(5 % 2);
```

DEĞİŞKENLER

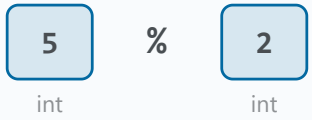
`sonuc = 2.0`

ÇIKTI

```
2
2.5
2.5
2.5
2.0
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



int % int → int

İki işlenen de **int** → kalan: 5 = 2·2 + 1.

Adım 13 / 14

```
1 System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
▶ System.out.println(5 % 2);
```

DEĞİŞKENLER

—

ÇIKTI

```
2
2.5
2.5
2.5
2.0
```

DÖNÜŞÜM · KODU ADIM ADIM ÇALIŞTIR

`5 / 2` ile `5.0 / 2`



Sonuç **1** (int) yazdırılır.

Adım 14 / 14

```
1 System.out.println(5 / 2);
2 System.out.println(5.0 / 2);
3 System.out.println(5 / 2.0);
4 System.out.println((double) 5 / 2);
5 System.out.println((double) (5 / 2));
▶ System.out.println(5 % 2);
```

DEĞİŞKENLER

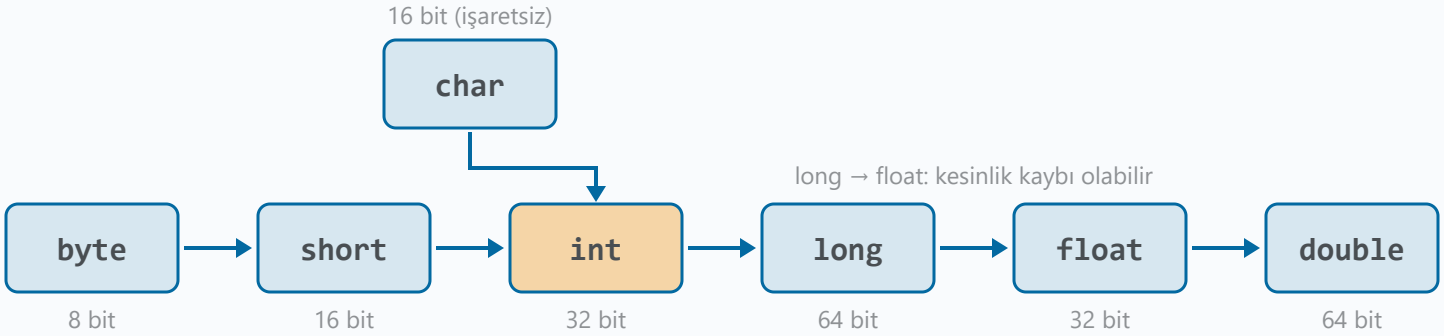
`sonuc = 1`

ÇIKTI

```
2
2.5
2.5
2.5
2.0
1
```

DÖNÜŞÜM

Tip Yükseltme Sırası

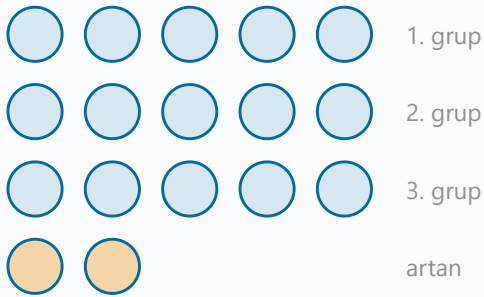


Oklar yönünde dönüşüm otomatiktir · ters yönde açık (tip) dönüşümü gerekir

- Karma ifadede küçük tip, büyük tipe **otomatik** yükselir: `int + double` → double.
- `byte`, `short`, `char` işlemde önce **int** olur; bu yüzden `b + 1` bir int'tir.
- Okun tersi (daraltma) için açık `(tür)` dönüşümü gerekir.

Aritmetik İşleçler

İşleç	Anlam	Örnek
+ -	Toplama / çıkarma	a + b
* /	Çarpma / bölme	a * b
%	Kalan	sayi % 2
++ --	Bir artır / azalt	i++
+= -=	Bileşik atama	toplam += değer



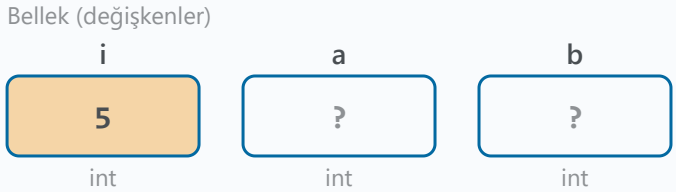

```
17 / 5 = 3
17 % 5 = 2
```

Sıfıra Bölme ve Özel Değerler

İfade	Sonuç	Açıklama
`1 / 0`	`ArithmeticException: / by zero`	tam sayı bölmesinde sıfıra bölme çalışma zamanı hatasıdır
`1.0 / 0`	`Infinity`	ondalıkta hata yok: sonsuz
`-1.0 / 0`	`-Infinity`	eksi sonsuz
`0.0 / 0`	`NaN`	Not a Number: tanımsız sonuç
`NaN == NaN`	`false`	NaN kendine bile eşit değildir
`NaN != NaN`	`true`	NaN kontrolü için `Double.isNaN(x)`
`-7 % 3`	`-1`	kalanın işareti bölünenin işaretidir
`5.5 % 2`	`1.5`	% kayan noktalı sayılarda da çalışır

Tam sayıda 1 / 0 programı durdurur; ondalıkta Infinity ve NaN sessizce üretilir ve sonraki işlemlere taşınır.

Artırma ve Azaltma: `i++` ile `++i`



i = 5.

Adım 1 / 7

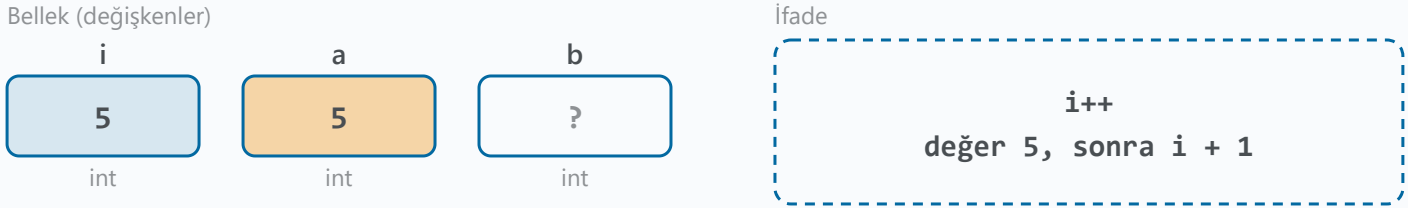
```
▶ int i = 5;
2 int a = i++;   // önce kullan, sonra artır
3 int b = ++i;   // önce artır, sonra kullan
4 i--;
5 System.out.println(i + " " + a + " " + b);
```

DEĞİŞKENLER

`i = 5`

ÇIKTI

Artırma ve Azaltma: `i++` ile `++i`



Sonek i++: ifadenin değeri i'nin **eski** değeridir → a = 5.

Adım 2 / 7

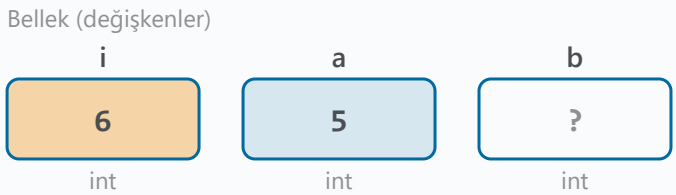
```
1 int i = 5;
▶ int a = i++;   // önce kullan, sonra artır
3 int b = ++i;   // önce artır, sonra kullan
4 i--;
5 System.out.println(i + " " + a + " " + b);
```

DEĞİŞKENLER

`i = 5` `a = 5`

ÇIKTI

Artırma ve Azaltma: `i++` ile `++i`



Ardından i artırılır: i = 6.

Adım 3 / 7

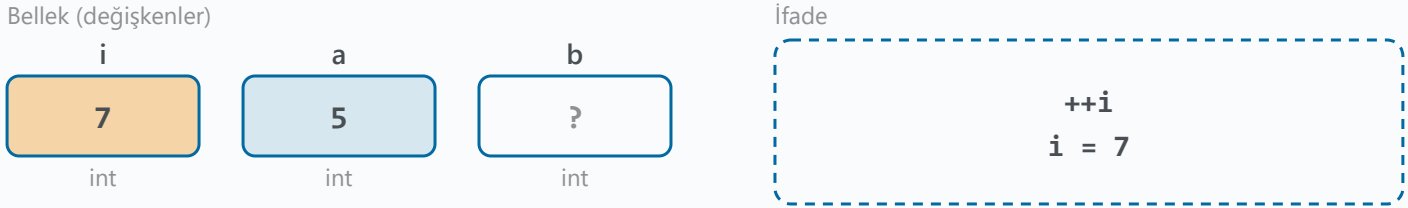
```
1 int i = 5;
▶ int a = i++;   // önce kullan, sonra artır
3 int b = ++i;   // önce artır, sonra kullan
4 i--;
5 System.out.println(i + " " + a + " " + b);
```

DEĞİŞKENLER

`i = 6` `a = 5`

ÇIKTI

Artırma ve Azaltma: `i++` ile `++i`



Önek ++i: önce i artırılır → i = 7.

Adım 4 / 7

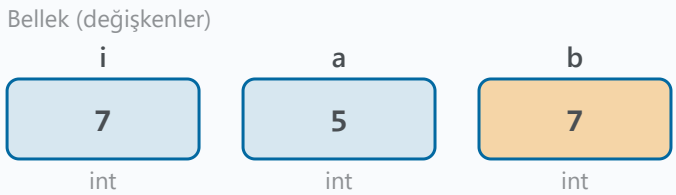
```
1 int i = 5;
2 int a = i++;   // önce kullan, sonra artır
▶ int b = ++i;   // önce artır, sonra kullan
4 i--;
5 System.out.println(i + " " + a + " " + b);
```

DEĞİŞKENLER

`i = 7` `a = 5`

ÇIKTI

Artırma ve Azaltma: `i++` ile `++i`



İfadenin değeri **yeni** değerdir → b = 7.

Adım 5 / 7

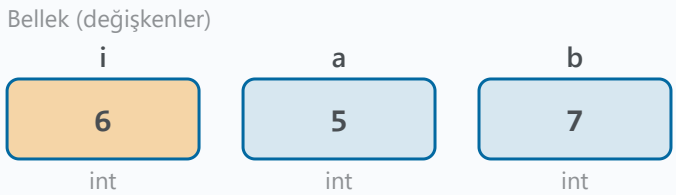
```
1 int i = 5;
2 int a = i++;   // önce kullan, sonra artır
▶ int b = ++i;   // önce artır, sonra kullan
4 i--;
5 System.out.println(i + " " + a + " " + b);
```

DEĞİŞKENLER

`i = 7` `a = 5` `b = 7`

ÇIKTI

Artırma ve Azaltma: `i++` ile `++i`



i-- tek başına bir deyim: önek/sonek farkı önemsiz → i = 6.

Adım 6 / 7

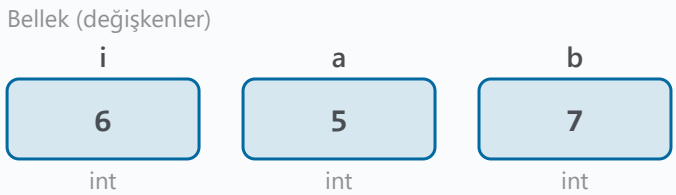
```
1 int i = 5;
2 int a = i++;   // önce kullan, sonra artır
3 int b = ++i;   // önce artır, sonra kullan
▶ i--;
5 System.out.println(i + " " + a + " " + b);
```

DEĞİŞKENLER

`i = 6` `a = 5` `b = 7`

ÇIKTI

Artırma ve Azaltma: `i++` ile `++i`



Çıktı: **6 5 7**. ++ ve -- tek başına bir deyim olarak kullanıldığında kod daha kolay okunur.

Adım 7 / 7

```
1 int i = 5;
2 int a = i++;   // önce kullan, sonra artır
3 int b = ++i;   // önce artır, sonra kullan
4 i--;
▶ System.out.println(i + " " + a + " " + b);
```

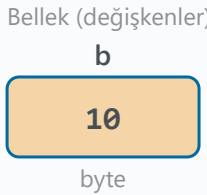
DEĞİŞKENLER

`i = 6` `a = 5` `b = 7`

ÇIKTI

```
6 5 7
```

Bileşik Atama Örtük Dönüşüm Yapar



b = 10: 8 bitlik byte kutusu.

Adım 1 / 6

```
▶ byte b = 10;
2 b += 5;
3 int i = 5;
4 i += 3.7;
5 // b = b + 5;   // derlenmez: int → byte
6 System.out.println(b + " " + i);
```

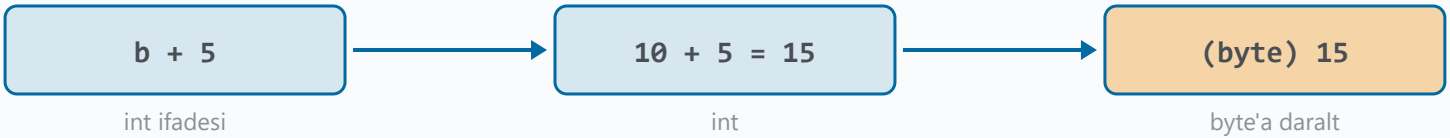
DEĞİŞKENLER

b = **10 (byte)**

ÇIKTI

Bileşik Atama Örtük Dönüşüm Yapar

b += 5 ≡ b = (byte)(b + 5)



b += 5 derleyicide `b = (byte)(b + 5)` olur: toplama int ile yapılır, sonuç **otomatik** byte'a daraltılır → b = 15.

Adım 2 / 6

```
1 byte b = 10;
▶ b += 5;
3 int i = 5;
4 i += 3.7;
5 // b = b + 5;   // derlenmez: int → byte
6 System.out.println(b + " " + i);
```

DEĞİŞKENLER

b = 15 (byte)

ÇIKTI

Bileşik Atama Örtük Dönüşüm Yapar



i = 5 (int).

Adım 3 / 6

```
1 byte b = 10;
2 b += 5;
▶ int i = 5;
4 i += 3.7;
5 // b = b + 5;   // derlenmez: int → byte
6 System.out.println(b + " " + i);
```

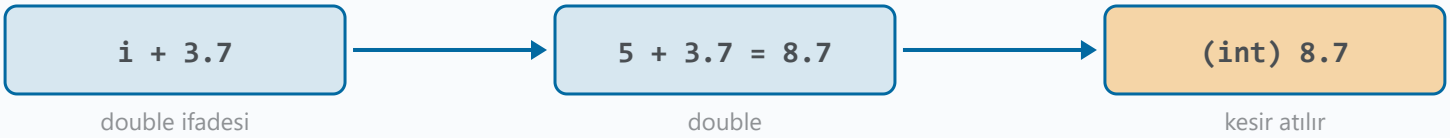
DEĞİŞKENLER

`b = 15 (byte)` `i = 5 (int)`

ÇIKTI

Bileşik Atama Örtük Dönüşüm Yapar

i += 3.7 ≡ i = (int)(i + 3.7)



i += 3.7 de örtük bir dönüşüm yapar: 8.7 hesaplanır, (int) ile kesir atılır → i = **8** (yuvarlama yok).

Adım 4 / 6

```
1 byte b = 10;
2 b += 5;
3 int i = 5;
▶ i += 3.7;
5 // b = b + 5;   // derlenmez: int → byte
6 System.out.println(b + " " + i);
```

DEĞİŞKENLER

`b = 15 (byte)` `i = 8 (int)`

ÇIKTI

Bileşik Atama Örtük Dönüşüm Yapar

`b = b + 5;`

`b + 5  →  int` (byte + int = int)

`error: incompatible types: possible lossy conversion from int to byte`

Bileşik atama olmadan `b = b + 5;` **derlenmez**: b + 5 int olur ve int → byte dönüşümü örtük yapılmaz. Bu satır yorum olarak duruyor.

Adım 5 / 6

```
1 byte b = 10;
2 b += 5;
3 int i = 5;
4 i += 3.7;
▶ // b = b + 5;   // derlenmez: int → byte
6 System.out.println(b + " " + i);
```

DEĞİŞKENLER

`b = 15 (byte)` `i = 8 (int)`

ÇIKTI

```
```

Bileşik Atama Örtük Dönüşüm Yapar



Çıktı: **15 8**.

Adım 6 / 6

```
1 byte b = 10;
2 b += 5;
3 int i = 5;
4 i += 3.7;
5 // b = b + 5;   // derlenmez: int → byte
▶ System.out.println(b + " " + i);
```

DEĞİŞKENLER

`b = 15 (byte)` `i = 8 (int)`

ÇIKTI

```
15 8
```

İŞLEÇLER

Karşılaştırma ve Mantık

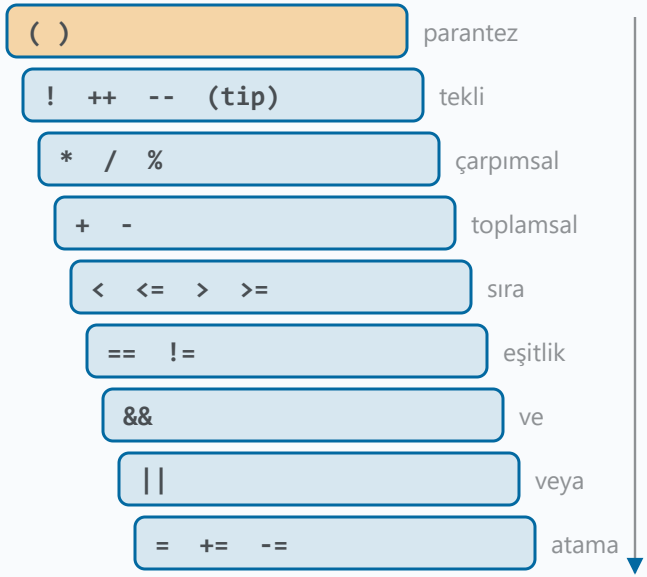
İşleç	Anlam	Örnek
`== !=`	Eşit / farklı	`a == b`
`< <= > >=`	Sıra karşılaştırması	`not >= 60`
`&&`	İki koşul da doğru	`yas >= 18 && belgeVar`
`||`	En az biri doğru	`haftaSonu || tatil`
`!`	Mantıksal değil	`!aktif`

p	q	p && q	p || q
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

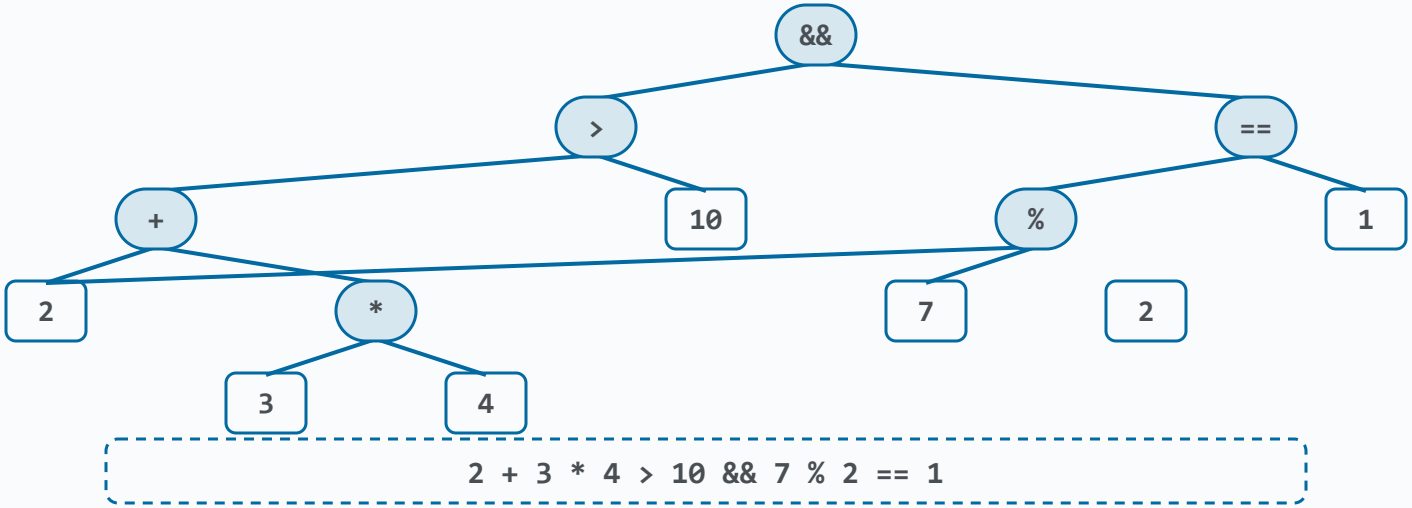
Karşılaştırma işleçlerinin sonucu her zaman boolean'dır.

İşleç Önceliği

- Parantez içi önce değerlendirilir; tekli `!` ve tip dönüşümü de önce gelir.
- Çarpma, bölme ve kalan; toplama ve çıkarmadan önce gelir.
- Karşılaştırmalar aritmetikten, `&&` / `||` karşılaştırmadan sonra değerlendirilir.
- Aynı düzeyde soldan sağa gidilir (atama hariç). Parantez niyeti görünür kılar.

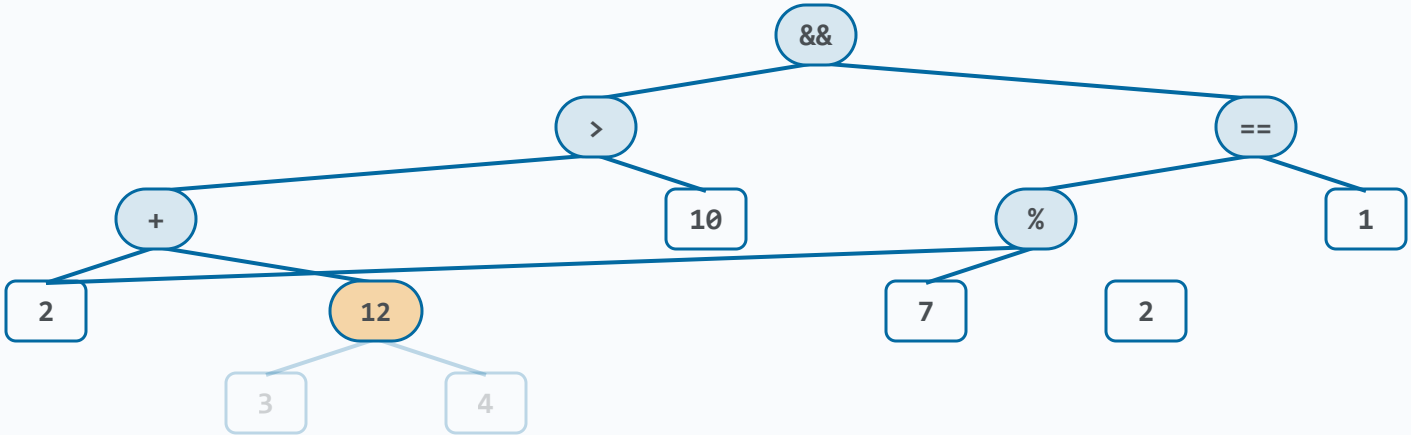


İfade Ağacı: `2 + 3 * 4 > 10 && 7 % 2 == 1`



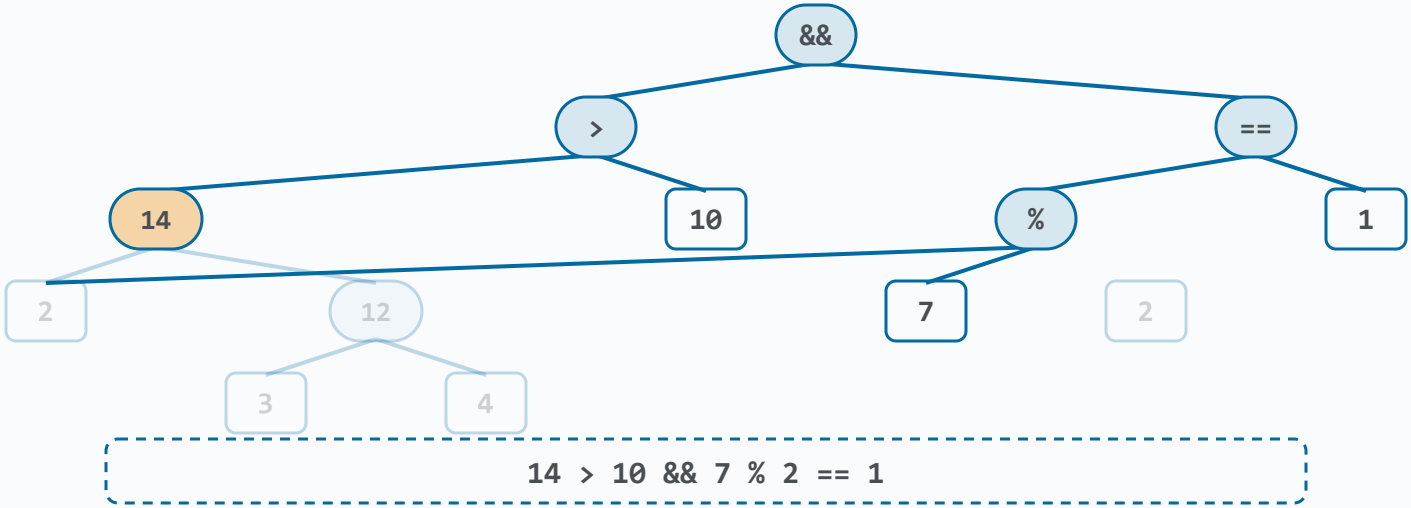
Derleyici ifadeyi önceliğe göre bir **ağaca** dönüştürür: en alttaki işlem en önce, kök (`&&`) en son hesaplanır.

İfade Ağacı: `2 + 3 * 4 > 10 && 7 % 2 == 1`



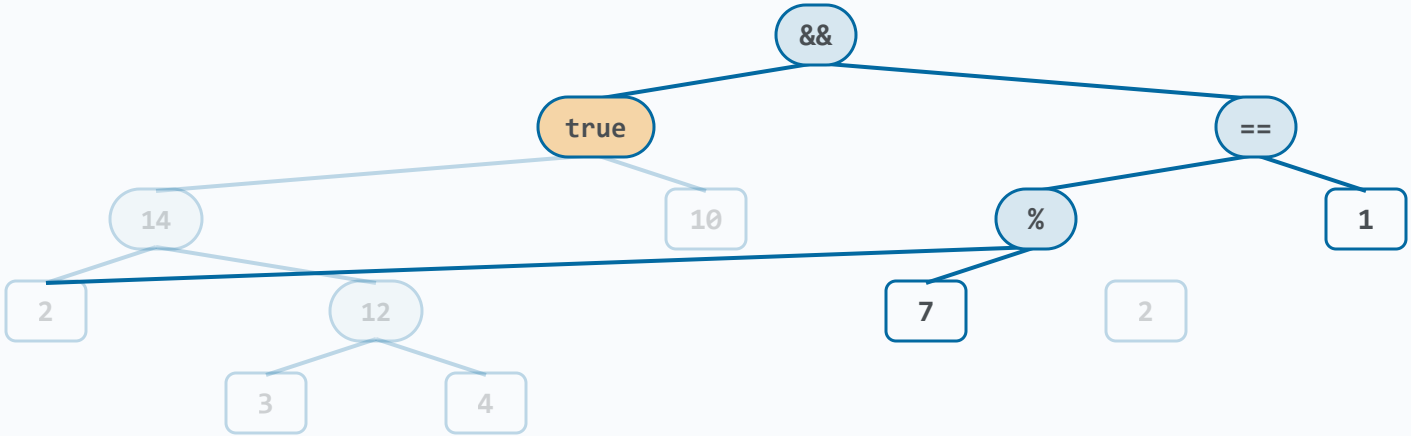
`*` , `+` 'dan önce gelir: 3 * 4 = **12**.

İfade Ağacı: `2 + 3 * 4 > 10 && 7 % 2 == 1`



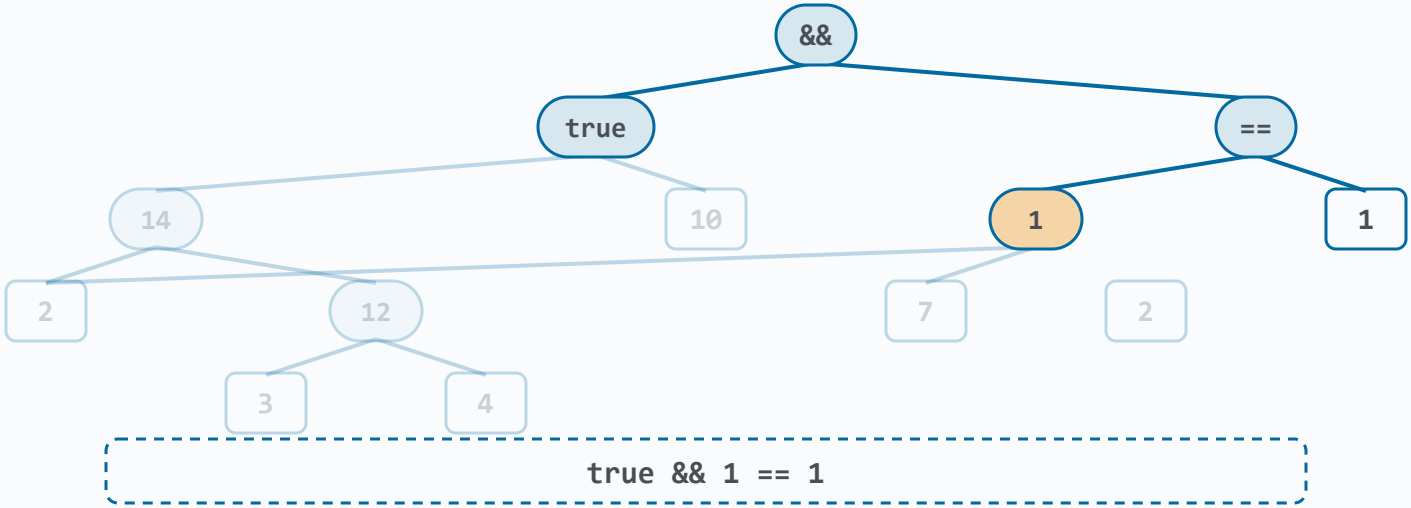
sonra toplama: 2 + 12 = **14**.

İfade Ağacı: `2 + 3 * 4 > 10 && 7 % 2 == 1`



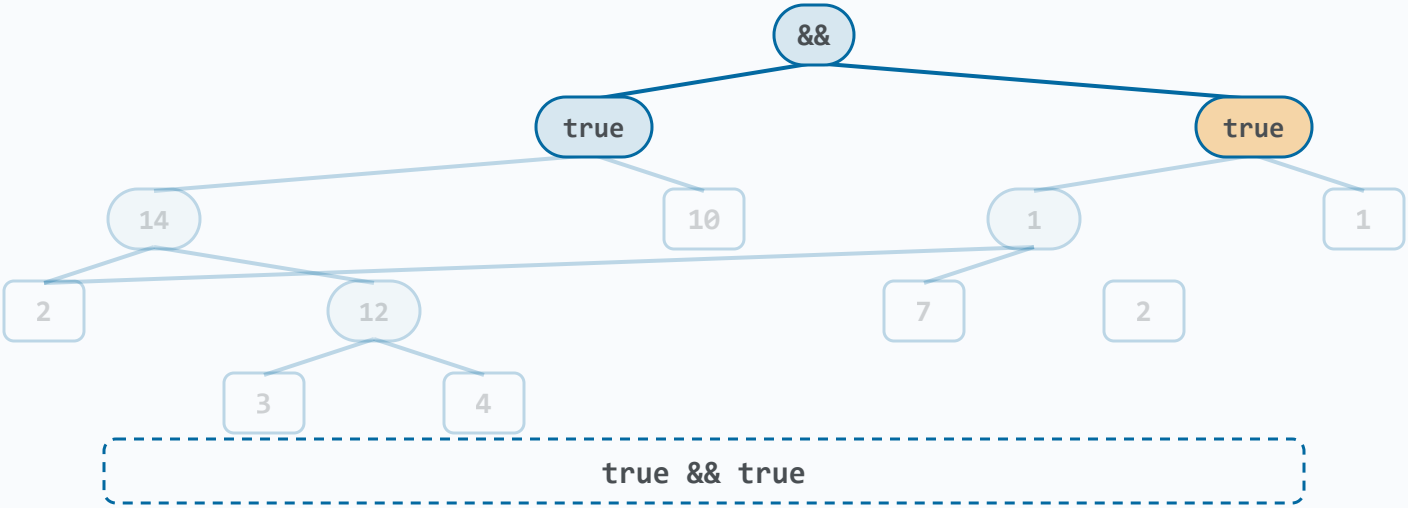
Karşılaştırma: 14 > 10 → **true** (boolean). Sol taraf true: `&&` sağ tarafa geçer.

İfade Ağacı: `2 + 3 * 4 > 10 && 7 % 2 == 1`



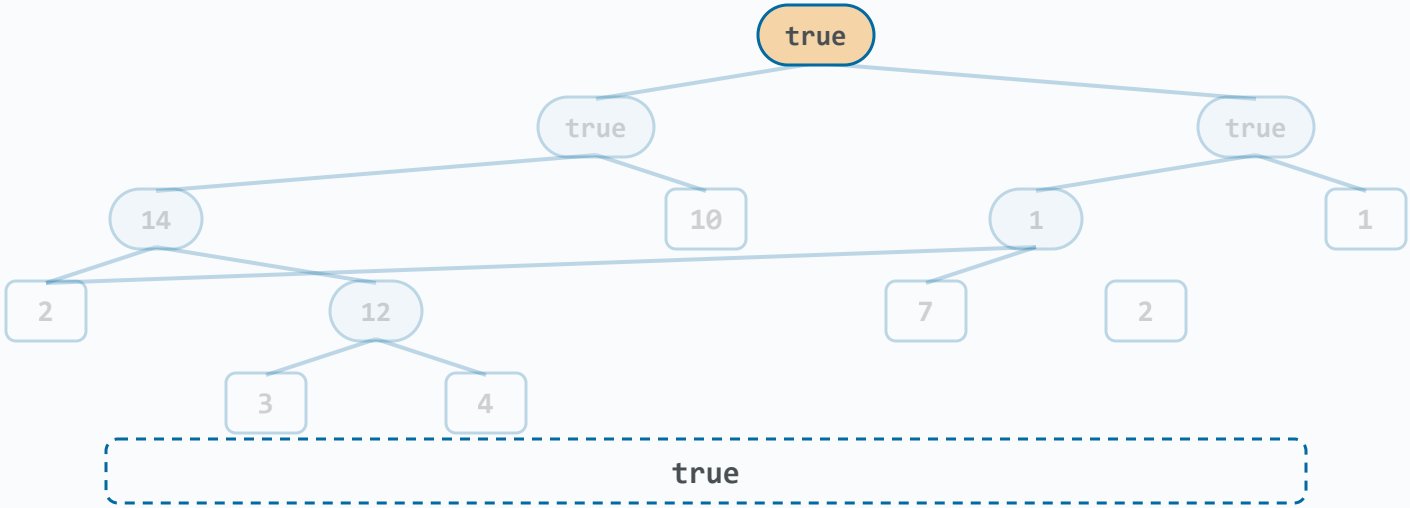
sağ tarafta önce kalan: 7 % 2 = **1**.

İfade Ağacı: `2 + 3 * 4 > 10 && 7 % 2 == 1`



Karşılaştırma: 1 == 1 → **true** (boolean).

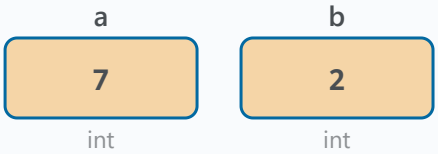
İfade Ağacı: `2 + 3 * 4 > 10 && 7 % 2 == 1`



Kök: true && true → **true**. Sol taraf true olduğu için sağ taraf da hesaplanmak zorundaydı.

Karma İfade: `a / b * 2.0 + a % b`

Bellek (değişkenler)



a = 7, b = 2 (int).

Adım 1 / 7

```
▶ int a = 7, b = 2;
2 double r = a / b * 2.0 + a % b;
3 System.out.println(r);
```

DEĞİŞKENLER

a = 7 b = 2

ÇIKTI

Karma İfade: `a / b * 2.0 + a % b`

`a / b * 2.0 + a % b`

önce / * % (soldan sağa), sonra +

`/`, `*`, `%` aynı öncelikte ve **soldan sağa** işlenir; `+` hepsinden sonra gelir.

Adım 2 / 7

```
1 int a = 7, b = 2;
▶ double r = a / b * 2.0 + a % b;
3 System.out.println(r);
```

DEĞİŞKENLER

`a = 7` `b = 2`

ÇIKTI

Karma İfade: `a / b * 2.0 + a % b`

3 * 2.0 + a % b

a / b = 7 / 2 = 3 (int / int → tam sayı bölmesi)

İlk: `a / b` iki int olduğundan **tam sayı bölmesi**: 7 / 2 = 3 (0.5 kesri atılır).

Adım 3 / 7

```
1 int a = 7, b = 2;
▶ double r = a / b * 2.0 + a % b;
3 System.out.println(r);
```

DEĞİŞKENLER

a = 7 b = 2

ÇIKTI

Karma İfade: `a / b * 2.0 + a % b`

`6.0 + a % b`

3 * 2.0 = 6.0 (int → double, sonuç double)

Sonra `3 * 2.0` : 3, double'a yükselir → **6.0**.

Adım 4 / 7

```
1 int a = 7, b = 2;
▶ double r = a / b * 2.0 + a % b;
3 System.out.println(r);
```

DEĞİŞKENLER

`a = 7` `b = 2`

ÇIKTI

Karma İfade: `a / b * 2.0 + a % b`

`6.0 + 1`

a % b = 7 % 2 = 1 (kalan, int)

`+` 'nın sağ tarafı: 7 % 2 = **1** (kalan).

Adım 5 / 7

```
1 int a = 7, b = 2;
▶ double r = a / b * 2.0 + a % b;
3 System.out.println(r);
```

DEĞİŞKENLER

`a = 7` `b = 2`

ÇIKTI

Karma İfade: `a / b * 2.0 + a % b`

7.0

6.0 + 1 = 7.0 (1 → 1.0, double)

Son: 6.0 + 1: int, double'a yükselir → **7.0**; r = 7.0 (double).

Adım 6 / 7

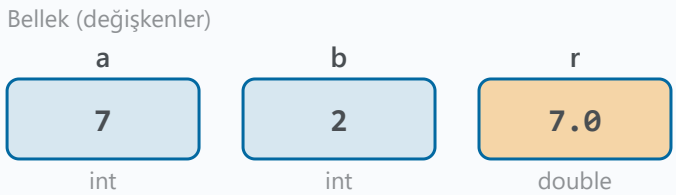
```
1 int a = 7, b = 2;
▶ double r = a / b * 2.0 + a % b;
3 System.out.println(r);
```

DEĞİŞKENLER

`a = 7` `b = 2` `r = 7.0 (double)`

ÇIKTI

Karma İfade: `a / b * 2.0 + a % b`



Çıktı: **7.0**. Tam sayı bölmesi yüzünden 3 × 2.0 = 6.0 çıktı; `a / 2.0` olsaydı 3.5 × 2.0 + 1 = 8.0 olurdu. Sıra ve tipler sonucu belirler.

Adım 7 / 7

```
1 int a = 7, b = 2;
2 double r = a / b * 2.0 + a % b;
▶ System.out.println(r);
```

DEĞİŞKENLER

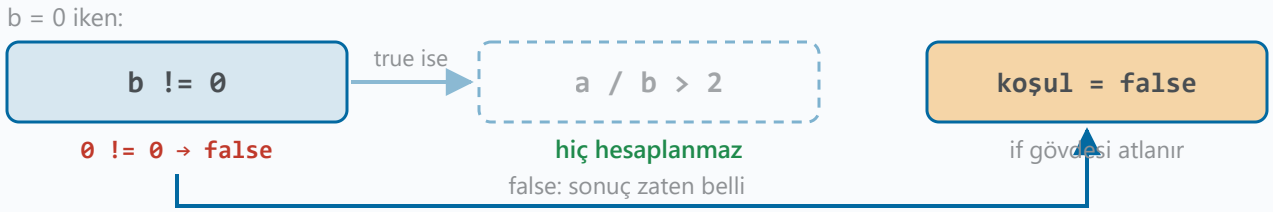
`a = 7` `b = 2` `r = 7.0 (double)`

ÇIKTI

```
7.0
```

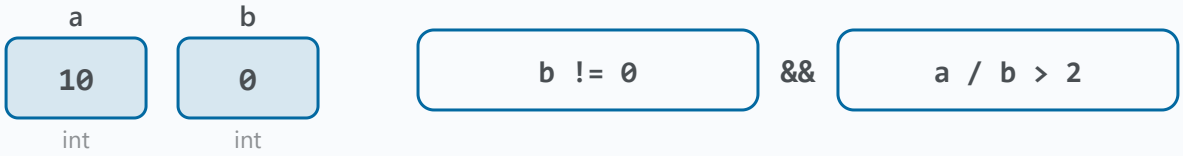
Kısa Devre Değerlendirme

```java
int a = 10, b = 0;
if (b != 0 && a / b > 2) {
    System.out.println("Oran büyük");
}
```



- `&&` ilk koşul false ise ikinci koşulu **çalıştırmaz**: sonuç zaten false.
- `b = 0` iken `a / b` çalışsaydı **ArithmeticException** (sıfıra bölme) oluşurdu; kısa devre bunu önler.
- `||` ilk koşul true ise ikincisine bakmaz. Bu yüzden koşulların **sırası** önemlidir.

Kısa Devre Adım Adım: b = 0 ve b = 2



a = 10, b = 0.

Adım 1 / 8

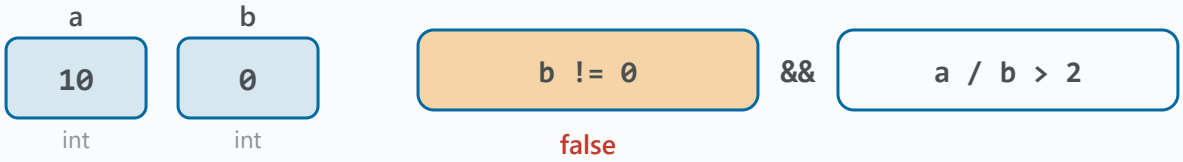
```
▶ int a = 10, b = 0;
2 if (b != 0 && a / b > 2) {
3     System.out.println("Oran büyük");
4 }
5 b = 2;
6 if (b != 0 && a / b > 2) {
7     System.out.println("Oran büyük");
8 }
```

DEĞİŞKENLER

a = 10 b = 0

ÇIKTI

Kısa Devre Adım Adım: b = 0 ve b = 2



Sol koşul: b != 0 → 0 != 0 → **false**.

Adım 2 / 8

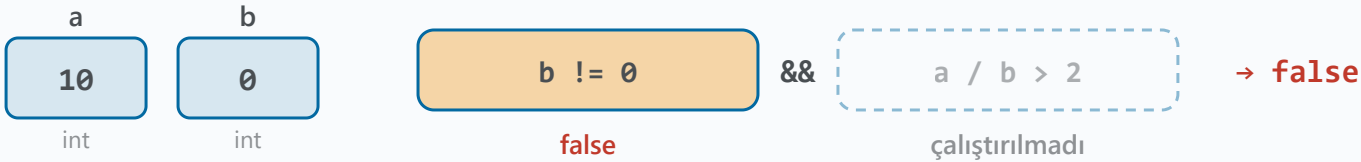
```
1 int a = 10, b = 0;
▶ if (b != 0 && a / b > 2) {
3     System.out.println("Oran büyük");
4 }
5 b = 2;
6 if (b != 0 && a / b > 2) {
7     System.out.println("Oran büyük");
8 }
```

DEĞİŞKENLER

`a = 10` `b = 0`

ÇIKTI

Kısa Devre Adım Adım: b = 0 ve b = 2



Kısa devre: false && (herhangi) → false. Sağ taraf **hiç çalışmaz**; a / b yapılmadığı için sıfıra bölme hatası da olmaz.

Adım 3 / 8

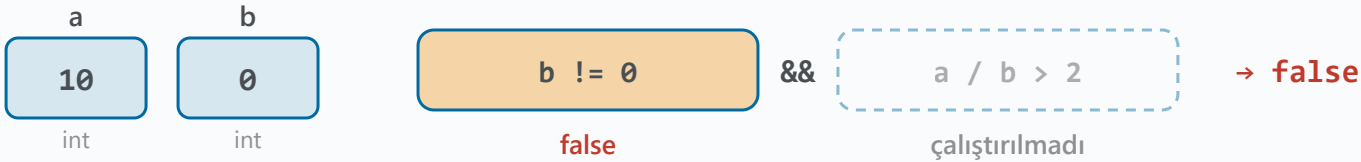
```
1 int a = 10, b = 0;
▶ if (b != 0 && a / b > 2) {
3     System.out.println("Oran büyük");
4 }
5 b = 2;
6 if (b != 0 && a / b > 2) {
7     System.out.println("Oran büyük");
8 }
```

DEĞİŞKENLER

a = 10 | b = 0

ÇIKTI

Kısa Devre Adım Adım: b = 0 ve b = 2



Koşul false → blok atlanır, hiçbir şey yazılmaz.

Adım 4 / 8

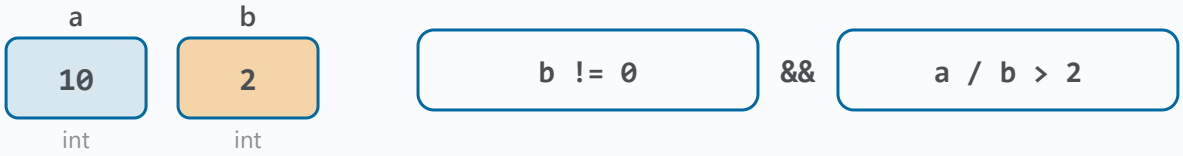
```
1 int a = 10, b = 0;
2 if (b != 0 && a / b > 2) {
3     System.out.println("Oran büyük");
▶ }
5 b = 2;
6 if (b != 0 && a / b > 2) {
7     System.out.println("Oran büyük");
8 }
```

DEĞİŞKENLER

a = 10 b = 0

ÇIKTI

Kısa Devre Adım Adım: b = 0 ve b = 2



b = 2 yapılır ve aynı if yeniden denenir.

Adım 5 / 8

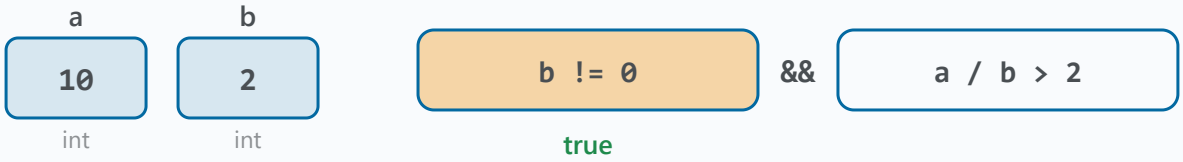
```
1 int a = 10, b = 0;
2 if (b != 0 && a / b > 2) {
3     System.out.println("Oran büyük");
4 }
▶ b = 2;
6 if (b != 0 && a / b > 2) {
7     System.out.println("Oran büyük");
8 }
```

DEĞİŞKENLER

a = 10 b = 2

ÇIKTI

Kısa Devre Adım Adım: b = 0 ve b = 2



Sol koşul: 2 != 0 → **true**. Sonuç henüz belli değil, sağ taraf da hesaplanmalı.

Adım 6 / 8

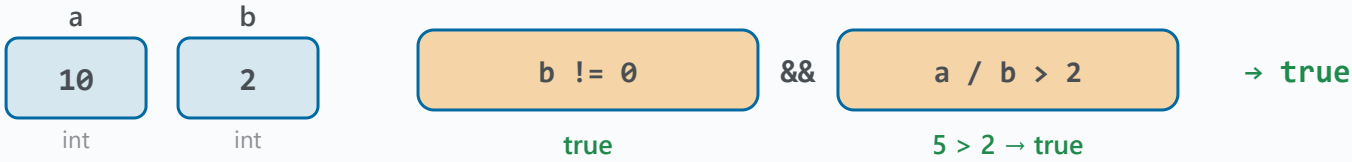
```
1 int a = 10, b = 0;
2 if (b != 0 && a / b > 2) {
3     System.out.println("Oran büyük");
4 }
5 b = 2;
▶ if (b != 0 && a / b > 2) {
7     System.out.println("Oran büyük");
8 }
```

DEĞİŞKENLER

a = 10 b = 2

ÇIKTI

Kısa Devre Adım Adım: b = 0 ve b = 2



Sağ koşul: a / b → 10 / 2 = 5; 5 > 2 → **true**. true && true → true.

Adım 7 / 8

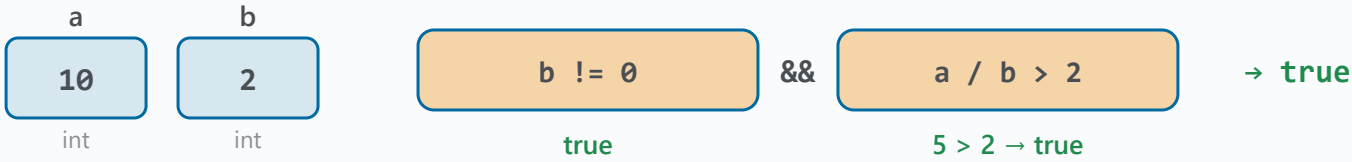
```
1 int a = 10, b = 0;
2 if (b != 0 && a / b > 2) {
3     System.out.println("Oran büyük");
4 }
5 b = 2;
▶ if (b != 0 && a / b > 2) {
7     System.out.println("Oran büyük");
8 }
```

DEĞİŞKENLER

a = 10 b = 2

ÇIKTI

Kısa Devre Adım Adım: b = 0 ve b = 2



Bloğa girilir: **Oran büyük**. Koşullar ters yazılsaydı (`a / b > 2 && b != 0`) b = 0 iken program çökerdi.

Adım 8 / 8

```
1 int a = 10, b = 0;
2 if (b != 0 && a / b > 2) {
3     System.out.println("Oran büyük");
4 }
5 b = 2;
6 if (b != 0 && a / b > 2) {
▶     System.out.println("Oran büyük");
8 }
```

DEĞİŞKENLER

`a = 10` `b = 2`

ÇIKTI

```
Oran büyük
```

Tipler Farklı Davranır

```
System.out.println(2 + 3);       // 5
System.out.println("2" + "3");   // 23
System.out.println("Sonuç: " + 2 + 3); // Sonuç: 23
System.out.println("Sonuç: " + (2 + 3)); // Sonuç: 5
```

- İşleç davranışı işlenen tiplerine bağlıdır.
- Parantez değerlendirme sırasını açıklaştırır.

`"Sonuç: " + 2 + 3` Adım Adım



2 + 3: iki int → sayısal **toplama**.

Adım 1 / 10

```
▶ System.out.println(2 + 3);
2 System.out.println("2" + "3");
3 System.out.println("Sonuç: " + 2 + 3);
4 System.out.println("Sonuç: " + (2 + 3));
```

ÇIKTI

VERİ · KODU ADIM ADIM ÇALIŞTIR

`"Sonuç: " + 2 + 3` Adım Adım

5

int

Çıktı: **5**.

Adım 2 / 10

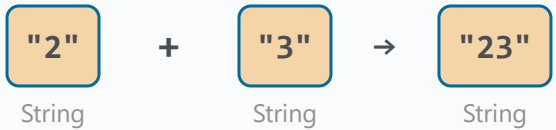
```
▶ System.out.println(2 + 3);
2 System.out.println("2" + "3");
3 System.out.println("Sonuç: " + 2 + 3);
4 System.out.println("Sonuç: " + (2 + 3));
```

ÇIKTI

```
5
```

VERİ · KODU ADIM ADIM ÇALIŞTIR

`"Sonuç: " + 2 + 3` Adım Adım



"2" + "3": işlenenlerden biri String → **birleştirme**: 3 metne eklenir.

Adım 3 / 10

```
1 System.out.println(2 + 3);
▶ System.out.println("2" + "3");
3 System.out.println("Sonuç: " + 2 + 3);
4 System.out.println("Sonuç: " + (2 + 3));
```

ÇIKTI

```
5
```

`"Sonuç: " + 2 + 3` Adım Adım



Çıktı: **23**.

Adım 4 / 10

```
1 System.out.println(2 + 3);
▶ System.out.println("2" + "3");
3 System.out.println("Sonuç: " + 2 + 3);
4 System.out.println("Sonuç: " + (2 + 3));
```

ÇIKTI

```
5
23
```

VERİ · KODU ADIM ADIM ÇALIŞTIR

`"Sonuç: " + 2 + 3` Adım Adım



"Sonuç: " + 2: işlenenlerden biri String → **birleştirme**: 2 metne eklenir. + soldan sağa değerlendirilir: önce ilk ikisi.

Adım 5 / 10

```
1 System.out.println(2 + 3);
2 System.out.println("2" + "3");
▶ System.out.println("Sonuç: " + 2 + 3);
4 System.out.println("Sonuç: " + (2 + 3));
```

ÇIKTI

```
5
23
```

`"Sonuç: " + 2 + 3` Adım Adım



"Sonuç: 2" + 3: işlenenlerden biri String → **birleştirme**: 3 metne eklenir.

Adım 6 / 10

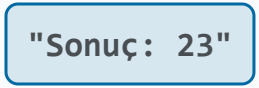
```
1 System.out.println(2 + 3);
2 System.out.println("2" + "3");
▶ System.out.println("Sonuç: " + 2 + 3);
4 System.out.println("Sonuç: " + (2 + 3));
```

ÇIKTI

```
5
23
```

VERİ · KODU ADIM ADIM ÇALIŞTIR

`"Sonuç: " + 2 + 3` Adım Adım



Çıktı: **Sonuç: 23**. 3 de metne eklendi; 2 + 3 hiç toplanmadı.

Adım 7 / 10

```
1 System.out.println(2 + 3);
2 System.out.println("2" + "3");
▶ System.out.println("Sonuç: " + 2 + 3);
4 System.out.println("Sonuç: " + (2 + 3));
```

ÇIKTI

```
5
23
Sonuç: 23
```

`"Sonuç: " + 2 + 3` Adım Adım



Parantez önce: 2 + 3, iki int → **toplama** = 5.

Adım 8 / 10

```
1 System.out.println(2 + 3);
2 System.out.println("2" + "3");
3 System.out.println("Sonuç: " + 2 + 3);
▶ System.out.println("Sonuç: " + (2 + 3));
```

ÇIKTI

```
5
23
Sonuç: 23
```

VERİ · KODU ADIM ADIM ÇALIŞTIR

`"Sonuç: " + 2 + 3` Adım Adım



Sonra "Sonuç: " + 5 → String varsa **birleştirme** → "Sonuç: 5".

Adım 9 / 10

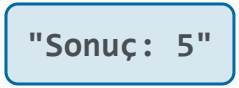
```
1 System.out.println(2 + 3);
2 System.out.println("2" + "3");
3 System.out.println("Sonuç: " + 2 + 3);
▶ System.out.println("Sonuç: " + (2 + 3));
```

ÇIKTI

```
5
23
Sonuç: 23
```

VERİ · KODU ADIM ADIM ÇALIŞTIR

`"Sonuç: " + 2 + 3` Adım Adım



Çıktı: **Sonuç: 5**. Parantez, toplamanın önce yapılmasını sağladı.

Adım 10 / 10

```
1 System.out.println(2 + 3);
2 System.out.println("2" + "3");
3 System.out.println("Sonuç: " + 2 + 3);
▶ System.out.println("Sonuç: " + (2 + 3));
```

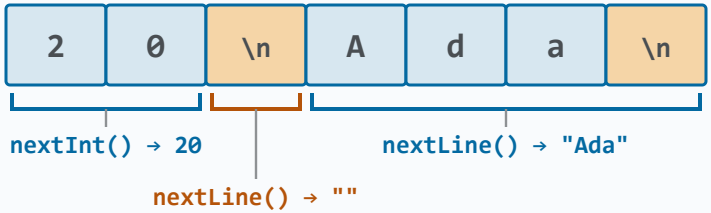
ÇIKTI

```
5
23
Sonuç: 23
Sonuç: 5
```

Scanner ile Girdi

```
Scanner klavye = new Scanner(System.in);
System.out.print("Yaş: ");
int yas = klavye.nextInt();
klavye.nextLine(); // satır sonunu tüket
System.out.print("Ad: ");
String ad = klavye.nextLine();
```

girdi tamponu
(kullanıcı: 20⏎ Ada⏎)

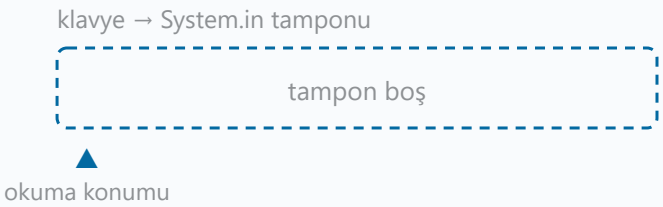


1. nextInt sayıyı alır, \n kalır
2. boş nextLine onu tüketir
3. sonraki nextLine adı okur

- Dosyanın başında `import java.util.Scanner;` gerekir.
- nextInt sayıyı, nextLine tüm satırı okur.
- Farklı okuma metotları birlikte kullanılırken kalan satır sonuna dikkat edilir.

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Scanner Girdi Akışı Adım Adım



Scanner nesnesi klavyeye (System.in) bağlanır. Tampon boş.

Adım 1 / 8

```
▶ Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
4 klavye.nextLine(); // satır sonunu tüket
5 System.out.print("Ad: ");
6 String ad = klavye.nextLine();
```

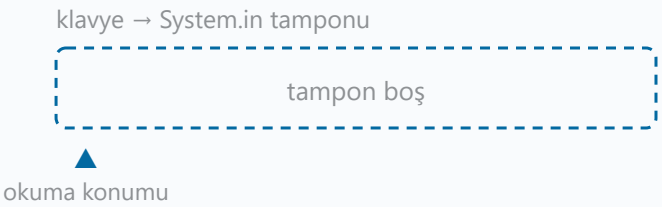
DEĞİŞKENLER

`klavye = Scanner`

ÇIKTI

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Scanner Girdi Akışı Adım Adım



`print` istemi yazar; satır sonu yok, imleç "Yaş: " arkasında.

Adım 2 / 8

```
1 Scanner klavye = new Scanner(System.in);
▶ System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
4 klavye.nextLine(); // satır sonunu tüket
5 System.out.print("Ad: ");
6 String ad = klavye.nextLine();
```

DEĞİŞKENLER

`klavye = Scanner`

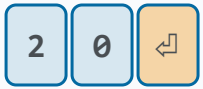
ÇIKTI

```
Yaş:
```

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Scanner Girdi Akışı Adım Adım

klavye → System.in tamponu



okuma konumu

⏳ kullanıcı yazıyor...

`nextInt()` tampon boş olduğu için **bekler**. Kullanıcı **20** yazıp Enter'a basar: tampona "20⏎" girer.

Adım 3 / 8

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
▶ int yas = klavye.nextInt();
4 klavye.nextLine(); // satır sonunu tüket
5 System.out.print("Ad: ");
6 String ad = klavye.nextLine();
```

DEĞİŞKENLER

`klavye = Scanner`

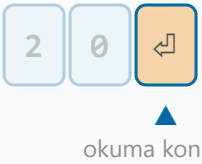
ÇIKTI

```
Yaş: 20
```

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Scanner Girdi Akışı Adım Adım

klavye → System.in tamponu



okunan: 20

nextInt yalnızca sayıyı (20) okur; ⏎ **tamponda kalır**.

Adım 4 / 8

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
▶ int yas = klavye.nextInt();
4 klavye.nextLine(); // satır sonunu tüket
5 System.out.print("Ad: ");
6 String ad = klavye.nextLine();
```

DEĞİŞKENLER

`klavye = Scanner` `yas = 20`

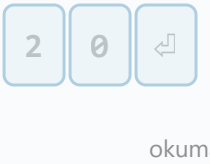
ÇIKTI

```
Yaş: 20
```

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Scanner Girdi Akışı Adım Adım

klavye → System.in tamponu



okuma konumu

okunan: ""

`nextLine()` ⏎'e kadar okur: boş dizgi ("") döner ve ⏎ tüketilir. Tampon temizlendi.

Adım 5 / 8

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
▶ klavye.nextLine(); // satır sonunu tüket
5 System.out.print("Ad: ");
6 String ad = klavye.nextLine();
```

DEĞİŞKENLER

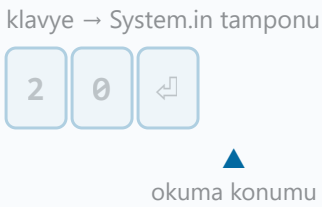
`klavye = Scanner` `yas = 20`

ÇIKTI

```
Yaş: 20
```

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Scanner Girdi Akışı Adım Adım



İstem "Ad: " yazılır.

Adım 6 / 8

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
4 klavye.nextLine(); // satır sonunu tüket
▶ System.out.print("Ad: ");
6 String ad = klavye.nextLine();
```

DEĞİŞKENLER

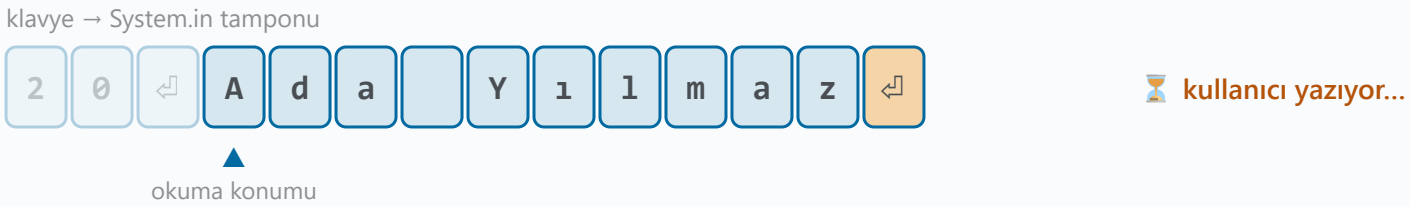
`klavye = Scanner` `yas = 20`

ÇIKTI

```
Yaş: 20
Ad:
```

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Scanner Girdi Akışı Adım Adım



nextLine tampon boş olduğu için bekler; kullanıcı **Ada Yılmaz** yazar ve Enter'a basar.

Adım 7 / 8

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
4 klavye.nextLine(); // satır sonunu tüket
5 System.out.print("Ad: ");
▶ String ad = klavye.nextLine();
```

DEĞİŞKENLER

`klavye = Scanner` `yas = 20`

ÇIKTI

```
Yaş: 20
Ad: Ada Yılmaz
```

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Scanner Girdi Akışı Adım Adım

klavye → System.in tamponu



okunan: "Ada Yılmaz"

nextLine tüm satırı (boşluk dahil) okur: ad = "Ada Yılmaz". ⏎ tüketilir, satıra dahil edilmez.

Adım 8 / 8

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
4 klavye.nextLine(); // satır sonunu tüket
5 System.out.print("Ad: ");
▶ String ad = klavye.nextLine();
```

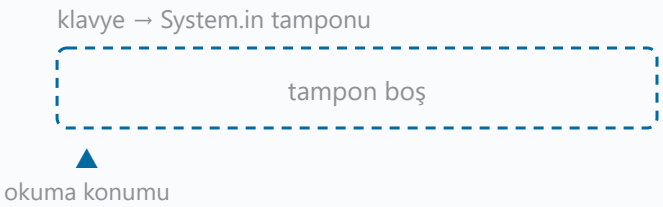
DEĞİŞKENLER

`klavye = Scanner` `yas = 20` `ad = "Ada Yılmaz"`

ÇIKTI

```
Yaş: 20
Ad: Ada Yılmaz
```

Satır Sonu Tüketilmezse



Scanner nesnesi klavyeye (System.in) bağlanır. Tampon boş.

Adım 1 / 7

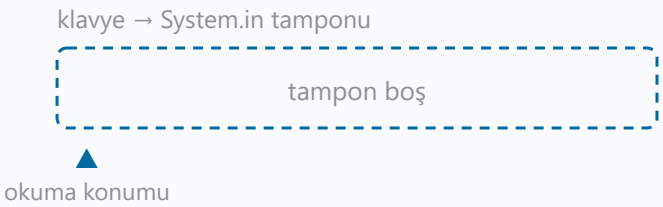
```
▶ Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
4 System.out.print("Ad: ");
5 String ad = klavye.nextLine();
6 System.out.println("[" + ad + "]");
```

DEĞİŞKENLER

`klavye = Scanner`

ÇIKTI

Satır Sonu Tüketilmezse



`print` istemi yazar; satır sonu yok, imleç "Yaş: " arkasında.

Adım 2 / 7

```
1 Scanner klavye = new Scanner(System.in);
▶ System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
4 System.out.print("Ad: ");
5 String ad = klavye.nextLine();
6 System.out.println("[" + ad + "]");
```

DEĞİŞKENLER

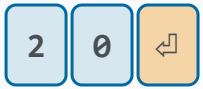
`klavye = Scanner`

ÇIKTI

```
Yaş:
```

Satır Sonu Tüketilmezse

klavye → System.in tamponu



okuma konumu

⌛ kullanıcı yazıyor...

`nextInt()` tampon boş olduğu için **bekler**. Kullanıcı **20** yazıp Enter'a basar: tampona "20⏎" girer.

Adım 3 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
▶ int yas = klavye.nextInt();
4 System.out.print("Ad: ");
5 String ad = klavye.nextLine();
6 System.out.println("[" + ad + "]");
```

DEĞİŞKENLER

`klavye = Scanner`

ÇIKTI

```
Yaş: 20
```

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Satır Sonu Tüketilmezse

klavye → System.in tamponu



okuma konumu

okunan: 20

nextInt yalnızca sayıyı (20) okur; ⏎ **tamponda kalır**.

Adım 4 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
▶ int yas = klavye.nextInt();
4 System.out.print("Ad: ");
5 String ad = klavye.nextLine();
6 System.out.println("[" + ad + "]");
```

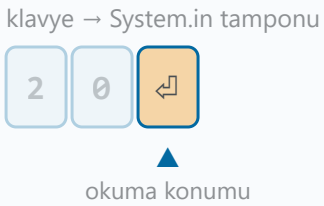
DEĞİŞKENLER

`klavye = Scanner` `yas = 20`

ÇIKTI

```
Yaş: 20
```

Satır Sonu Tüketilmezse



İstem "Ad: " yazılır.


```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
▶ System.out.print("Ad: ");
5 String ad = klavye.nextLine();
6 System.out.println("[" + ad + "]");
```

DEĞİŞKENLER

`klavye = Scanner` `yas = 20`

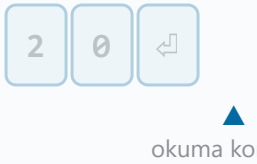
ÇIKTI

```
Yaş: 20
Ad:
```

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Satır Sonu Tüketilmezse

klavye → System.in tamponu



okunan: ""

Tamponda hâlâ ⏎ var: nextLine **beklemeden** boş dizgi döner. ad = "". Kullanıcıya ad sorulamadı!

Adım 6 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
4 System.out.print("Ad: ");
▶ String ad = klavye.nextLine();
6 System.out.println("[" + ad + "]");
```

DEĞİŞKENLER

klavye = Scanner | yas = 20 | ad = ""

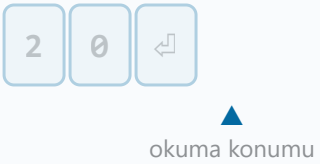
ÇIKTI

```
Yaş: 20
Ad:
```

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Satır Sonu Tüketilmezse

klavye → System.in tamponu



Çıktı **[]**: ad boş. Çözüm: nextInt'ten sonra bir `klavye.nextLine();` ile ⏎'i tüketmek.

Adım 7 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.print("Yaş: ");
3 int yas = klavye.nextInt();
4 System.out.print("Ad: ");
5 String ad = klavye.nextLine();
▶ System.out.println("[" + ad + "]");
```

DEĞİŞKENLER

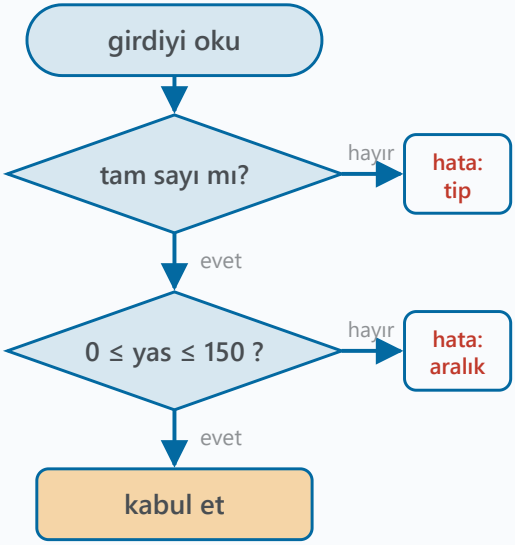
`klavye = Scanner` `yas = 20` `ad = ""`

ÇIKTI

```
Yaş: 20
Ad: []
```

Girdi Doğrulama

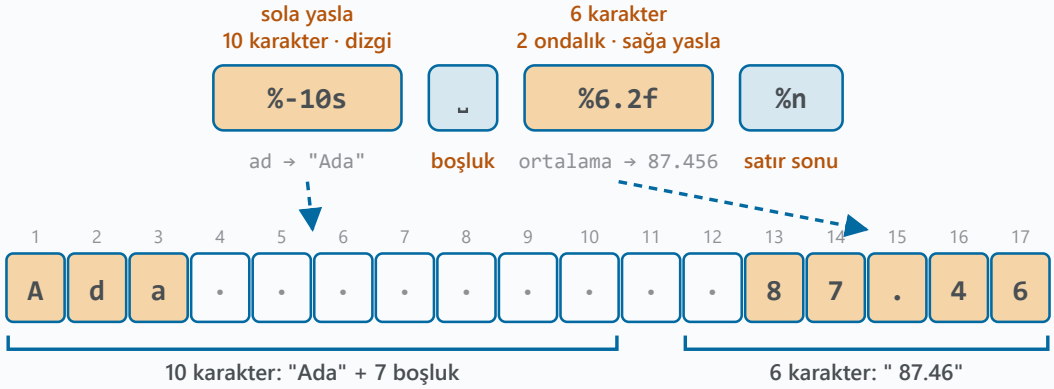
- Kullanıcının her zaman beklenen tipte değer gireceği varsayılmamalıdır.
- Aralık ve iş kuralı doğrulaması tip kontrolünden sonra yapılır.
- Hata iletisi beklenen biçimi açıkça söylemeli ve yeniden deneme yolu sunmalıdır.



ÇIKTI

Biçimli Çıktı: `printf`

```
String ad = "Ada";
double ortalama = 87.456;
System.out.printf("%-10s %6.2f%n", ad, ortalama);
```



87.456 → 87.46: yuvarlanır (kesilmez)

%s dizgi · %d tam sayı · %f kayan noktalı · %n satır sonu. Türkçe yerel ayarda ondalık ayırıcı virgüldür (87,46); derste nokta (`Locale.US`) kullanılır. Türkçe ayarda `nextDouble()` "37,5" bekler, "37.5" `InputMismatchException` verir.

ÇIKTI · KODU ADIM ADIM ÇALIŞTIR

printf Biçimini Adım Adım Çözümle

Biçim dizgisi: 3 biçim belirteci, 2 sabit karakter, 1 satır sonu



argümanlar sırayla: "Ada", 3.14159, 42

her % belirteci bir argümanı tüketir; | düz metindir

Biçim dizgisi soldan sağa taranır: **%…** biçim belirteçleri sırayla argümanlarla doldurulur, diğer karakterler olduğu gibi yazılır.

Adım 1 / 5

```
System.out.printf("%-8s|%6.2f|%05d%n",
                  "Ada", 3.14159, 42);
```

ÇIKTI

ÇIKTI · KODU ADIM ADIM ÇALIŞTIR

printf Biçimini Adım Adım Çözümle

%-8s ← "Ada"



genişlik 8, - = sola yaslı, 5 boşluk eklenir

%-8s: dizgi, 8 karakter genişliğinde; **-** sola yaslar → `"Ada "` (3 harf + 5 boşluk).

Adım 2 / 5

```
System.out.printf("%-8s|%6.2f|%05d%n",
                  "Ada", 3.14159, 42);
```

ÇIKTI

```
Ada
```

ÇIKTI · KODU ADIM ADIM ÇALIŞTIR

printf Biçimini Adım Adım Çözümle

| sabit, %6.2f ← 3.14159



genişlik 6, noktadan sonra 2 basamak, sağa yaslı

`|` aynen yazılır. **%6.2f**: 3.14159 → 2 basamağa yuvarlanır (3.14), 6 karaktere **sağa** yaslanır (2 boşluk). Türkçe yerel ayarda **3,14** yazılır.

Adım 3 / 5

```
▶ System.out.printf("%-8s|%6.2f|%05d%n",
▶                   "Ada", 3.14159, 42);
```

ÇIKTI

```
Ada     |   3.14
```

ÇIKTI · KODU ADIM ADIM ÇALIŞTIR

printf Biçimini Adım Adım Çözümle

| sabit, %05d ← 42



genişlik 5, 0 bayrağı = soldan sıfırla doldur

%05d: tam sayı, 5 karakter genişliğinde; **0** bayrağı boşluk yerine sıfır koyar → `00042`.

Adım 4 / 5

```
System.out.printf("%-8s|%6.2f|%05d%n",
                  "Ada", 3.14159, 42);
```

ÇIKTI

```
Ada     |  3.14|00042
```

ÇIKTI · KODU ADIM ADIM ÇALIŞTIR

printf Biçimini Adım Adım Çözümle

Sonuç

```
Ada      |  3.14|00042
```

%n: satır sonu · ondalık ayırıcı yerel ayara bağlı: Locale.US → 3.14, tr → 3,14

%n satır sonu ekler. Çıktı: `Ada  |  3.14|00042`. Ondalık ayırıcı yerel ayara bağlıdır; nokta için `Locale.US` verilir.

Adım 5 / 5

```
System.out.printf("%-8s|%6.2f|%05d%n",
                  "Ada", 3.14159, 42);
```

ÇIKTI

```
Ada      |  3.14|00042
```

ÖRNEKLER

Math Sınıfı

İfade	Sonuç	Açıklama
`Math.PI`	`3.141592653589793`	π sabiti
`Math.sqrt(16)`	`4.0`	karekök (double)
`Math.pow(2, 10)`	`1024.0`	üs alma (double)
`Math.abs(-7)`	`7`	mutlak değer
`Math.max(3, 8)`	`8`	büyük olan (min: küçük)
`Math.round(2.5)`	`3`	en yakın tam sayı (long)
`Math.round(-2.5)`	`-2`	yarım değer yukarı yuvarlanır



Math, java.lang paketindedir: import gerekmez. sqrt ve pow her zaman double döndürür.

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

İki Değişkeni Takas Etme



a = 5, b = 8. Hedef: değerleri **takas** etmek (a = 8, b = 5).

Adım 1 / 9

```
▶ int a = 5, b = 8;
2 a = b;                // yanlış deneme
3 b = a;
4 System.out.println(a + " " + b);
5 a = 5; b = 8;         // baştan
6 int gecici = a;       // doğru yöntem
7 a = b;
8 b = gecici;
9 System.out.println(a + " " + b);
```

DEĞİŞKENLER

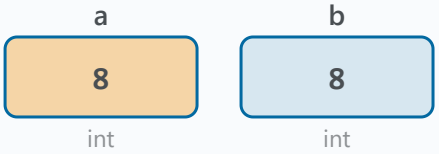
a = **5** b = **8**

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

İki Değişkeni Takas Etme

Bellek (değişkenler)



İfade

a ← b
a = 8

a = b → a = 8. a'nın eski değeri 5 **silindi**!

Adım 2 / 9

```
1 int a = 5, b = 8;
▶ a = b;                  // yanlış deneme
3 b = a;
4 System.out.println(a + " " + b);
5 a = 5; b = 8;           // baştan
6 int gecici = a;         // doğru yöntem
7 a = b;
8 b = gecici;
9 System.out.println(a + " " + b);
```

DEĞİŞKENLER

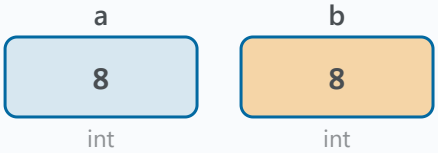
a = 8 b = 8

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

İki Değişkeni Takas Etme

Bellek (değişkenler)



İfade

b ← a
b = 8

b = a → b = 8. 5 artık hiçbir kutuda yok.

Adım 3 / 9

```
1 int a = 5, b = 8;
2 a = b;                // yanlış deneme
▶ b = a;
4 System.out.println(a + " " + b);
5 a = 5; b = 8;         // baştan
6 int gecici = a;       // doğru yöntem
7 a = b;
8 b = gecici;
9 System.out.println(a + " " + b);
```

DEĞİŞKENLER

a = 8 b = 8

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

İki Değişkeni Takas Etme



Çıktı **8 8**: takas başarısız. Bir değeri korumak için üçüncü bir kutu gerekir.

Adım 4 / 9

```
1 int a = 5, b = 8;
2 a = b;                // yanlış deneme
3 b = a;
▶ System.out.println(a + " " + b);
5 a = 5; b = 8;         // baştan
6 int gecici = a;       // doğru yöntem
7 a = b;
8 b = gecici;
9 System.out.println(a + " " + b);
```

DEĞİŞKENLER

`a = 8` `b = 8`

ÇIKTI

```
8 8
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

İki Değişkeni Takas Etme



Değerler yeniden 5 ve 8 yapıldı.

Adım 5 / 9

```
1 int a = 5, b = 8;
2 a = b;                // yanlış deneme
3 b = a;
4 System.out.println(a + " " + b);
▶ a = 5; b = 8;         // baştan
6 int gecici = a;       // doğru yöntem
7 a = b;
8 b = gecici;
9 System.out.println(a + " " + b);
```

DEĞİŞKENLER

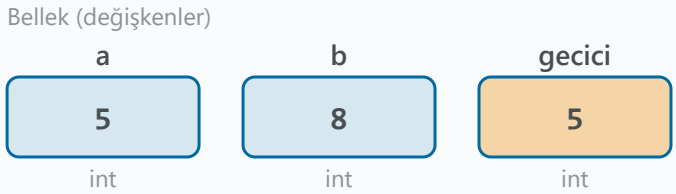
a = 5 b = 8

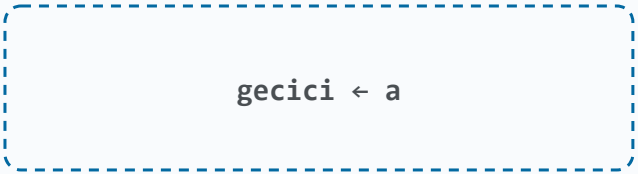
ÇIKTI

```
8 8
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

İki Değişkeni Takas Etme





gecici = a → gecici = 5: a'nın değeri güvende.

Adım 6 / 9

```
1 int a = 5, b = 8;
2 a = b;                  // yanlış deneme
3 b = a;
4 System.out.println(a + " " + b);
5 a = 5; b = 8;           // baştan
▶ int gecici = a;         // doğru yöntem
7 a = b;
8 b = gecici;
9 System.out.println(a + " " + b);
```

DEĞİŞKENLER

`a = 5` `b = 8` `gecici = 5`

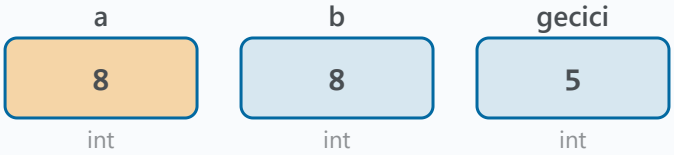
ÇIKTI

```
8 8
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

İki Değişkeni Takas Etme

Bellek (değişkenler)



İfade

a ← b

a = b → a = 8.

Adım 7 / 9

```
1 int a = 5, b = 8;
2 a = b;                 // yanlış deneme
3 b = a;
4 System.out.println(a + " " + b);
5 a = 5; b = 8;          // baştan
6 int gecici = a;        // doğru yöntem
▶ a = b;
8 b = gecici;
9 System.out.println(a + " " + b);
```

DEĞİŞKENLER

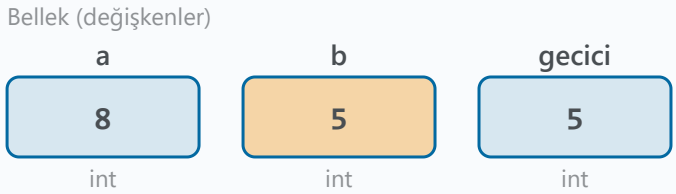
a = 8 | b = 8 | gecici = 5

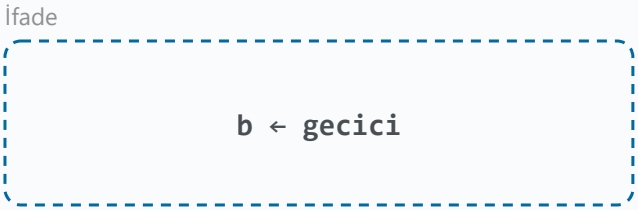
ÇIKTI

```
8 8
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

İki Değişkeni Takas Etme





b = gecici → b = 5.

Adım 8 / 9

```
1 int a = 5, b = 8;
2 a = b;                // yanlış deneme
3 b = a;
4 System.out.println(a + " " + b);
5 a = 5; b = 8;         // baştan
6 int gecici = a;       // doğru yöntem
7 a = b;
▶ b = gecici;
9 System.out.println(a + " " + b);
```

DEĞİŞKENLER

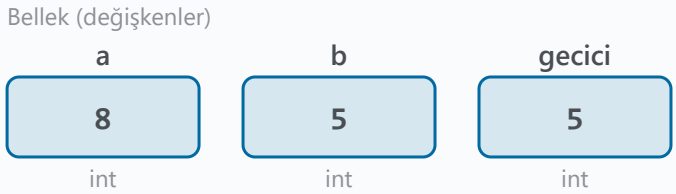
a = 8 | b = 5 | gecici = 5

ÇIKTI

```
8 8
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

İki Değişkeni Takas Etme



Çıktı **8 5**: takas tamam. 3 atama ve 1 geçici değişken gerekti.

Adım 9 / 9

```
1 int a = 5, b = 8;
2 a = b;                 // yanlış deneme
3 b = a;
4 System.out.println(a + " " + b);
5 a = 5; b = 8;          // baştan
6 int gecici = a;        // doğru yöntem
7 a = b;
8 b = gecici;
▶ System.out.println(a + " " + b);
```

DEĞİŞKENLER

`a = 8` `b = 5` `gecici = 5`

ÇIKTI

```
8 8
8 5
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Celsius → Fahrenheit: 100, 37, −40



1. çalıştırma: kullanıcı 100 girer; nextDouble → c = 100.0.

Adım 1 / 9

```
1 Scanner klavye = new Scanner(System.in);
▶ double c = klavye.nextDouble();
3 double f = c * 9 / 5 + 32;
4 System.out.printf("%.1f C = %.1f F%n", c, f);
```

DEĞİŞKENLER

c = 100.0

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Celsius → Fahrenheit: 100, 37, −40

Bellek (değişkenler)

c	f
100.0	?
double	double

İfade

```
c * 9 / 5 + 32
900.0 / 5 + 32
→ 212.0
```

Soldan sağa: c * 9 = 900.0, / 5 = 180.0, + 32 = **212.0**.

Adım 2 / 9

```
1 Scanner klavye = new Scanner(System.in);
2 double c = klavye.nextDouble();
▶ double f = c * 9 / 5 + 32;
4 System.out.printf("%.1f C = %.1f F%n", c, f);
```

DEĞİŞKENLER

c = 100.0

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Celsius → Fahrenheit: 100, 37, −40



printf %.1f ile bir ondalık basamak: **100.0 C = 212.0 F**. `9 / 5 * c` yazılsaydı 9 / 5 = 1 (tam sayı bölmesi) olurdu!

Adım 3 / 9

```
1 Scanner klavye = new Scanner(System.in);
2 double c = klavye.nextDouble();
3 double f = c * 9 / 5 + 32;
▶ System.out.printf("%.1f C = %.1f F%n", c, f);
```

DEĞİŞKENLER

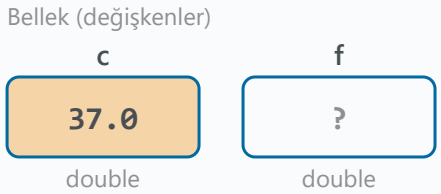
c = 100.0 f = 212.0

ÇIKTI

```
100.0 C = 212.0 F
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Celsius → Fahrenheit: 100, 37, −40



2. çalıştırma: kullanıcı 37 girer; nextDouble → c = 37.0.

Adım 4 / 9

```
1 Scanner klavye = new Scanner(System.in);
▶ double c = klavye.nextDouble();
3 double f = c * 9 / 5 + 32;
4 System.out.printf("%.1f C = %.1f F%n", c, f);
```

DEĞİŞKENLER

c = 37.0

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Celsius → Fahrenheit: 100, 37, −40

Bellek (değişkenler)

c	f
37.0	?
double	double

İfade

```
c * 9 / 5 + 32
333.0 / 5 + 32
→ 98.6
```

Soldan sağa: c * 9 = 333.0, / 5 = 66.6, + 32 = **98.6**.

Adım 5 / 9

```
1 Scanner klavye = new Scanner(System.in);
2 double c = klavye.nextDouble();
▶ double f = c * 9 / 5 + 32;
4 System.out.printf("%.1f C = %.1f F%n", c, f);
```

DEĞİŞKENLER

c = **37.0**

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Celsius → Fahrenheit: 100, 37, −40



printf %.1f ile bir ondalık basamak: **37.0 C = 98.6 F**.

Adım 6 / 9

```
1 Scanner klavye = new Scanner(System.in);
2 double c = klavye.nextDouble();
3 double f = c * 9 / 5 + 32;
▶ System.out.printf("%.1f C = %.1f F%n", c, f);
```

DEĞİŞKENLER

c = 37.0 f = 98.6

ÇIKTI

```
37.0 C = 98.6 F
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Celsius → Fahrenheit: 100, 37, −40



3. çalıştırma: kullanıcı -40 girer; nextDouble → c = -40.0.

Adım 7 / 9

```
1 Scanner klavye = new Scanner(System.in);
▶ double c = klavye.nextDouble();
3 double f = c * 9 / 5 + 32;
4 System.out.printf("%.1f C = %.1f F%n", c, f);
```

DEĞİŞKENLER

c = −40.0

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Celsius → Fahrenheit: 100, 37, −40

Bellek (değişkenler)

c	f
-40.0	?
double	double

İfade

```
c * 9 / 5 + 32
-360.0 / 5 + 32
→ -40.0
```

Soldan sağa: c * 9 = -360.0, / 5 = -72.0, + 32 = **-40.0**.

Adım 8 / 9

```
1 Scanner klavye = new Scanner(System.in);
2 double c = klavye.nextDouble();
▶ double f = c * 9 / 5 + 32;
4 System.out.printf("%.1f C = %.1f F%n", c, f);
```

DEĞİŞKENLER

c = −40.0

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Celsius → Fahrenheit: 100, 37, −40



printf %.1f ile bir ondalık basamak: **-40.0 C = -40.0 F**. −40 iki ölçekte aynıdır.

Adım 9 / 9

```
1 Scanner klavye = new Scanner(System.in);
2 double c = klavye.nextDouble();
3 double f = c * 9 / 5 + 32;
▶ System.out.printf("%.1f C = %.1f F%n", c, f);
```

DEĞİŞKENLER

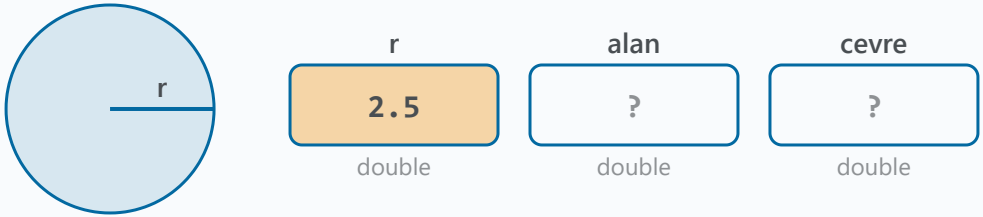
c = −40.0 f = −40.0

ÇIKTI

```
-40.0 C = -40.0 F
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Daire Alanı ve Çevresi: r = 2.5



Yarıçap r = 2.5.

Adım 1 / 5

```
▶ double r = 2.5;
2 double alan = Math.PI * r * r;
3 double cevre = 2 * Math.PI * r;
4 System.out.printf("Alan: %.2f%n", alan);
5 System.out.printf("Çevre: %.2f%n", cevre);
```

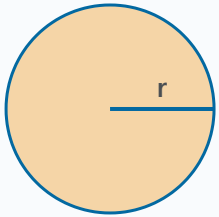
DEĞİŞKENLER

r = 2.5

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Daire Alanı ve Çevresi: r = 2.5



r	alan	cevre
2.5	19.634954	?
double	double	double

π · r²
≈ 19.635

Alan = π · r · r = 3.14159… · 2.5 · 2.5 = 19.634954084936208. Math.PI hazır sabittir.

Adım 2 / 5

```
1 double r = 2.5;
▶ double alan = Math.PI * r * r;
3 double cevre = 2 * Math.PI * r;
4 System.out.printf("Alan: %.2f%n", alan);
5 System.out.printf("Çevre: %.2f%n", cevre);
```

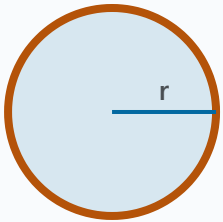
DEĞİŞKENLER

r = 2.5 | alan = 19.634954084936208

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Daire Alanı ve Çevresi: r = 2.5



r	alan	cevre
2.5	19.634954	15.707963
double	double	double

2 · π · r
≈ 15.708

Çevre = 2 · π · r = 15.707963267948966.

Adım 3 / 5

```
1 double r = 2.5;
2 double alan = Math.PI * r * r;
▶ double cevre = 2 * Math.PI * r;
4 System.out.printf("Alan: %.2f%n", alan);
5 System.out.printf("Çevre: %.2f%n", cevre);
```

DEĞİŞKENLER

r = 2.5 | alan = 19.634954084936208 | cevre = 15.707963267948966

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Daire Alanı ve Çevresi: r = 2.5



%.2f iki basamağa **yuvarlar**: 19.63. Değişkendeki değer değişmez.

Adım 4 / 5

```
1 double r = 2.5;
2 double alan = Math.PI * r * r;
3 double cevre = 2 * Math.PI * r;
▶ System.out.printf("Alan: %.2f%n", alan);
5 System.out.printf("Çevre: %.2f%n", cevre);
```

DEĞİŞKENLER

`r = 2.5` `alan = 19.634954084936208` `cevre = 15.707963267948966`

ÇIKTI

```
Alan: 19.63
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Daire Alanı ve Çevresi: r = 2.5



Çevre: **15.71** (15.7079… → 15.71).

Adım 5 / 5

```
1 double r = 2.5;
2 double alan = Math.PI * r * r;
3 double cevre = 2 * Math.PI * r;
4 System.out.printf("Alan: %.2f%n", alan);
▶ System.out.printf("Çevre: %.2f%n", cevre);
```

DEĞİŞKENLER

`r = 2.5` `alan = 19.634954084936208` `cevre = 15.707963267948966`

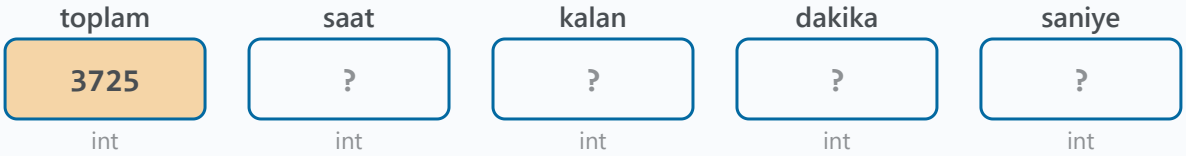
ÇIKTI

```
Alan: 19.63
Çevre: 15.71
```

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Saniye → saat:dakika:saniye (3725)

Bellek (değişkenler)



toplam = 3725 saniye.

Adım 1 / 6

```
▶ int toplam = 3725;
2 int saat = toplam / 3600;
3 int kalan = toplam % 3600;
4 int dakika = kalan / 60;
5 int saniye = kalan % 60;
6 System.out.printf("%d:%02d:%02d%n",
7                   saat, dakika, saniye);
```

DEĞİŞKENLER

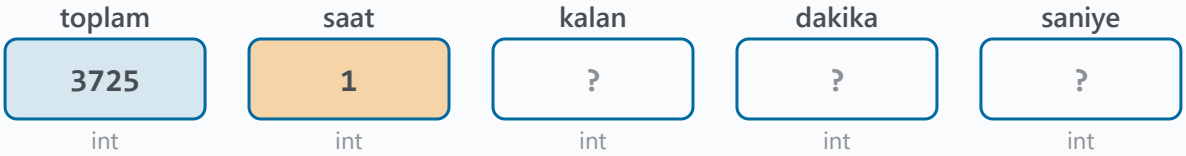
`toplam = 3725`

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Saniye → saat:dakika:saniye (3725)

Bellek (değişkenler)



saat = toplam / 3600 → 3725 / 3600 = **1** (tam sayı bölmesi: kaç tane tam saat).

Adım 2 / 6

```
1 int toplam = 3725;
▶ int saat = toplam / 3600;
3 int kalan = toplam % 3600;
4 int dakika = kalan / 60;
5 int saniye = kalan % 60;
6 System.out.printf("%d:%02d:%02d%n",
7                   saat, dakika, saniye);
```

DEĞİŞKENLER

`toplam = 3725` `saat = 1`

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Saniye → saat:dakika:saniye (3725)

Bellek (değişkenler)

toplam	saat	kalan	dakika	saniye
3725	1	125	?	?
int	int	int	int	int

kalan = toplam % 3600 → 3725 % 3600 = **125** (kalan: tam saatlerden artan saniye).

Adım 3 / 6

```
1 int toplam = 3725;
2 int saat = toplam / 3600;
▶ int kalan = toplam % 3600;
4 int dakika = kalan / 60;
5 int saniye = kalan % 60;
6 System.out.printf("%d:%02d:%02d%n",
7                   saat, dakika, saniye);
```

DEĞİŞKENLER

toplam = **3725** saat = **1** kalan = **125**

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Saniye → saat:dakika:saniye (3725)

Bellek (değişkenler)

toplam	saat	kalan	dakika	saniye
3725	1	125	2	?
int	int	int	int	int

dakika = kalan / 60 → 125 / 60 = **2** (tam sayı bölmesi: kaç tane tam dakika).

Adım 4 / 6

```
1 int toplam = 3725;
2 int saat = toplam / 3600;
3 int kalan = toplam % 3600;
▶ int dakika = kalan / 60;
5 int saniye = kalan % 60;
6 System.out.printf("%d:%02d:%02d%n",
7                   saat, dakika, saniye);
```

DEĞİŞKENLER

toplam = **3725** saat = **1** kalan = **125** dakika = **2**

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Saniye → saat:dakika:saniye (3725)

Bellek (değişkenler)

toplam	saat	kalan	dakika	saniye
3725	1	125	2	5
int	int	int	int	int

saniye = kalan % 60 → 125 % 60 = **5** (kalan: tam dakikalardan artan saniye).

Adım 5 / 6

```
1 int toplam = 3725;
2 int saat = toplam / 3600;
3 int kalan = toplam % 3600;
4 int dakika = kalan / 60;
▶ int saniye = kalan % 60;
6 System.out.printf("%d:%02d:%02d%n",
7                   saat, dakika, saniye);
```

DEĞİŞKENLER

toplam = **3725** | saat = **1** | kalan = **125** | dakika = **2** | saniye = **5**

ÇIKTI

ÖRNEK · KODU ADIM ADIM ÇALIŞTIR

Saniye → saat:dakika:saniye (3725)

Bellek (değişkenler)

toplam	saat	kalan	dakika	saniye
3725	1	125	2	5
int	int	int	int	int

%02d: en az 2 hane, boşluk yerine 0 ile doldur → **1:02:05**. Sağlama: 1·3600 + 2·60 + 5 = 3725.

Adım 6 / 6

```
1 int toplam = 3725;
2 int saat = toplam / 3600;
3 int kalan = toplam % 3600;
4 int dakika = kalan / 60;
5 int saniye = kalan % 60;
▶ System.out.printf("%d:%02d:%02d%n",
▶                   saat, dakika, saniye);
```

DEĞİŞKENLER

toplam = **3725** | saat = **1** | kalan = **125** | dakika = **2** | saniye = **5**

ÇIKTI

```
1:02:05
```

Yaygın Hatalar

Hata	Örnek	Ne olur?
= ile == karışır	`if (x = 5)`	Derleme hatası: int, boolean değil
Tam sayı bölmesi	`double o = 7 / 2;`	o = 3.0, 3.5 değil
Kapsam dışı kullanım	`{ int k = 1; }` `k++;`	k blok dışında tanımsız
Yanlış noktalı virgül	`if (x > 0);`	Boş deyim: koşul hiçbir şeyi kontrol etmez
String ve ==	`s1 == s2`	İçeriği değil, referansı karşılaştırır → `s1.equals(s2)`

ÖZET

Bölüm Özeti

- Değişken, belirli tipteki bir değere anlamlı bir ad verir ve okunmadan önce değer almalıdır.
- İlkel tipler farklı değer kümeleri ve bellek gereksinimleri taşır.
- Genişletme otomatik yapılır; daraltma açık dönüşüm ister ve bilgi kaybettirebilir.
- İşleç önceliği ve işlenen tipleri ifadenin sonucunu doğrudan belirler.
- Kullanıcı girdisi hem tip hem iş kuralı açısından doğrulanmalıdır.

ÖZET

Kontrol Soruları

1. int ve double bölmesi nasıl farklı sonuç verir?
2. Daraltıcı dönüşüm neden açıkça yazılır?
3. "Sonuç: " + 2 + 3 ifadesi neden Sonuç: 23 yazdırır?
4. && işleci kısa devreyi nasıl kullanır?
5. nextInt sonrasında nextLine kullanırken neye dikkat edilir?

ÖZET

Alıştırmalar

1. Kenarları klavyeden okunan dikdörtgenin alanını ve çevresini yazdırın.
2. Kilometreyi mile çevirin (1 mil = 1,609 km); sonucu `%.2f` ile yazdırın.
3. Üç notun ortalamasını hesaplayın. `(n1 + n2 + n3) / 3` neden hatalı olabilir? Düzeltin.
4. Dört basamaklı bir sayının basamakları toplamını `%` ve `/` ile bulun (1234 → 10).
5. Kuruş cinsinden bir tutarı lira ve kuruşa ayırın (1575 → 15 lira 75 kuruş).
6. Takası geçici değişken olmadan `a = a + b; b = a - b; a = a - b;` ile izleyin. Taşma riski var mı?

ÖZET

Kaynaklar

- Deitel ve Deitel, Java: How to Program, Early Objects, 11. baskı, Pearson, 2018.
- Liang, Introduction to Java Programming and Data Structures, 13. baskı, Pearson, 2024.
- Oracle, The Java Tutorials: **Variables**, **Operators**.
- Oracle, Java Language Specification SE 21: **Bölüm 4, Types, Values, and Variables**; **Bölüm 15, Expressions**.
- Oracle, Java SE 21 API: **Scanner**, **Math**.
- Sedgewick ve Wayne, Computer Science: An Interdisciplinary Approach, Addison-Wesley, 2016: **1.2 Built-in Types of Data**.

Son

Değişkenler, Veri Tipleri ve İşleçler

Sonraki sunum: Koşullu İfadeler · Çalışma soruları · Simülatör: İfade Değerlendirici · Terim sözlüğü