Bölüm 11: Dizgiler

Bilgisayar ve Programlamaya Giriş

String · İndis · Metotlar · Karşılaştırma · Biçimleme

Sercan KÜLCÜ · Giresun Üniversitesi

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü: https://sercankulcu.github.io/files/computer/slides/Bolum_11_Dizgiler.html

GİRİŞ

Öğrenme Çıktıları

Temsil

String ve char ayrımını açıklamak.

İşleme

İndis ve temel metotları kullanmak.

Karşılaştırma

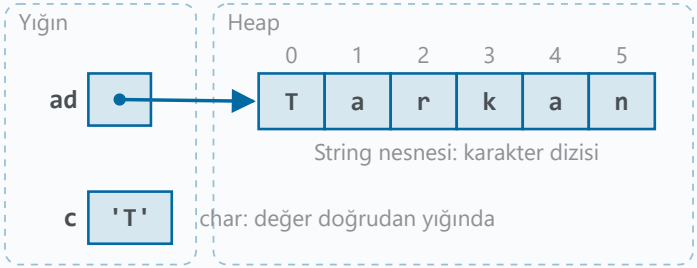
İçerik ve referans karşılaştırmasını ayırmak.

Çıktı

printf ile biçimli metin üretmek.

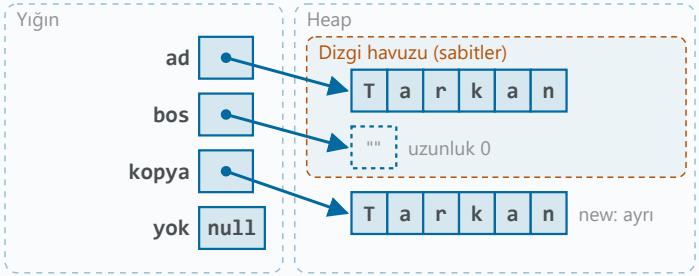
String Nedir?

- `String`, karakter dizisini temsil eden `java.lang` sınıfıdır (import gerekmez).
- Dizgi sabitleri çift tırnakla (`"Tarkan"`), `char` tek tırnakla (`'T'`) yazılır.
- `String` ilkel tip değil, **değiştirilemez** bir nesnedir; değişken nesnenin referansını tutar.



String Oluşturma

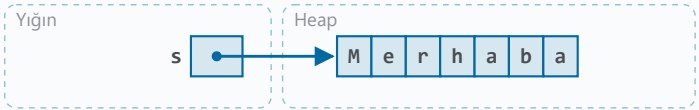
```
String ad = "Tarkan";
String bos = "";
String kopya = new String(ad);
String yok = null;
```



- Çoğunlukla `new` kullanmadan dizgi sabiti tercih edilir.
- `""` uzunluğu 0 olan gerçek bir nesnedir; `null` ise hiçbir nesneyi göstermez (`yok.length()` → `NullPointerException`).

Değiştirilemezlik

- String oluşturulduktan sonra içeriği **değişmez**.
- Birleştirme ve dönüştürme işlemleri **yeni** String üretir; eski nesne olduğu gibi kalır.
- Sonucu kullanmak için dönen değer bir değişkene atanmalıdır.
- Hiçbir değişkenin göstermediği nesneyi çöp toplayıcı (garbage collector) temizler.

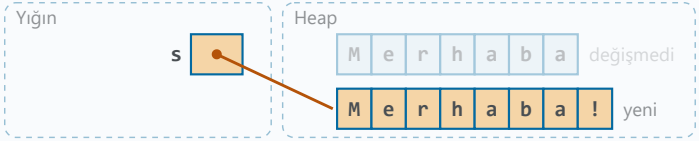



```
String s = "Merhaba";
```

Adım 1 / 2

Değiştirilemezlik

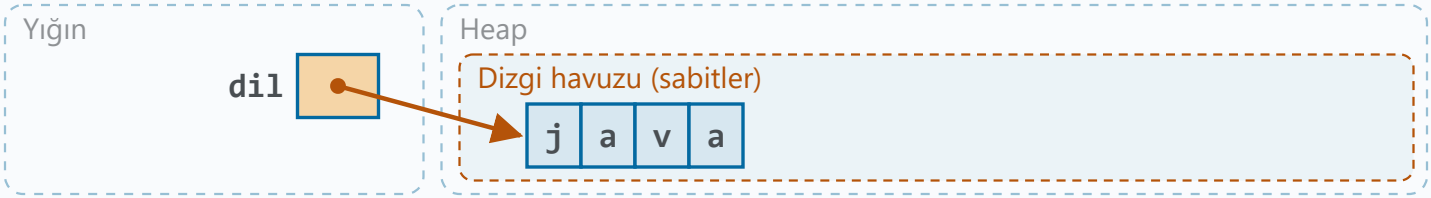
- String oluşturulduktan sonra içeriği **değişmez**.
- Birleştirme ve dönüştürme işlemleri **yeni** String üretir; eski nesne olduğu gibi kalır.
- Sonucu kullanmak için dönen değer bir değişkene atanmalıdır.
- Hiçbir değişkenin göstermediği nesneyi çöp toplayıcı (garbage collector) temizler.



`s = s + "!";` → yeni nesne; ok ona taşındı, eski nesne aynı.

Adım 2 / 2

Değiştirilemezlik: `toUpperCase()`



`dil`, havuzdaki `"java"` nesnesini gösterir.

Adım 1 / 5

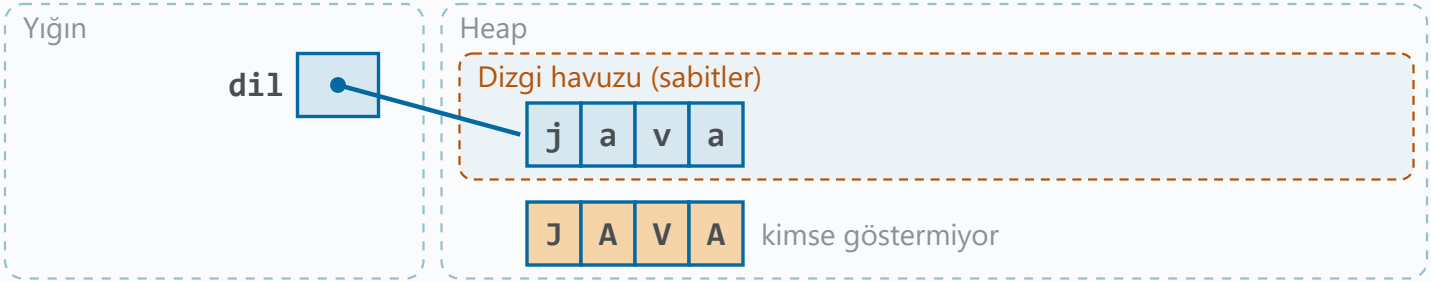
```
▶ String dil = "java";
2 dil.toUpperCase();
3 System.out.println(dil); // java
4 dil = dil.toUpperCase();
5 System.out.println(dil); // JAVA
```

DEĞİŞKENLER

`dil = "java"`

ÇIKTI

Değiştirilemezlik: `toUpperCase()`



`toUpperCase()` **yeni** bir `"JAVA"` nesnesi üretir; ama sonuç hiçbir değişkene atanmadı → kaybolur. `dil` hâlâ eski nesnede.

Adım 2 / 5

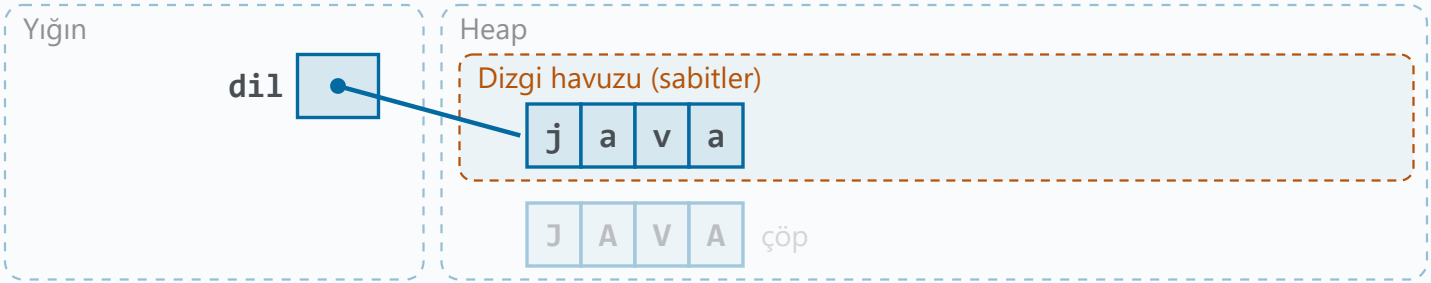
```
1 String dil = "java";
▶ dil.toUpperCase();
3 System.out.println(dil); // java
4 dil = dil.toUpperCase();
5 System.out.println(dil); // JAVA
```

DEĞİŞKENLER

`dil = "java"`

ÇIKTI

Değiştirilemezlik: `toUpperCase()`



Eski nesne değişmediği için **java** yazar.

Adım 3 / 5

```
1 String dil = "java";
2 dil.toUpperCase();
▶ System.out.println(dil); // java
4 dil = dil.toUpperCase();
5 System.out.println(dil); // JAVA
```

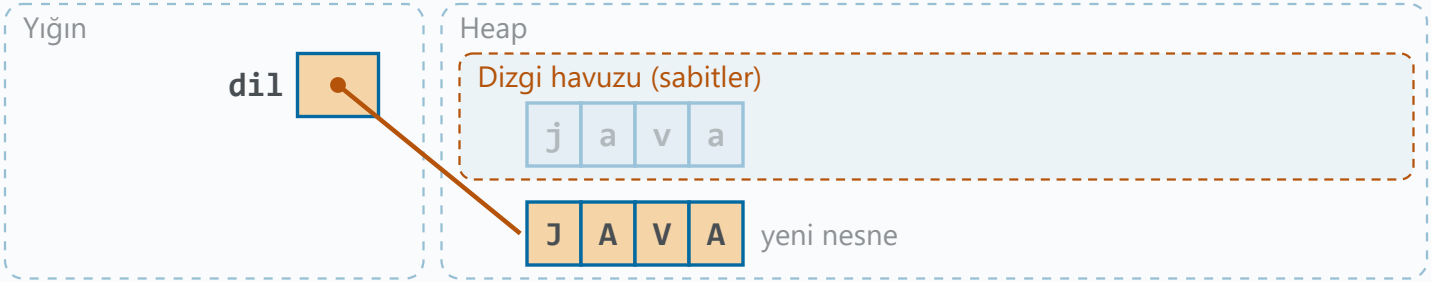
DEĞİŞKENLER

`dil = "java"`

ÇIKTI

```
java
```

Değiştirilemezlik: `toUpperCase()`



Bu kez dönen yeni nesne `dil`'e **atanır**: ok yeni nesneye taşınır. `"java"` nesnesi yine aynen durur.

Adım 4 / 5

```
1 String dil = "java";
2 dil.toUpperCase();
3 System.out.println(dil); // java
▶ dil = dil.toUpperCase();
5 System.out.println(dil); // JAVA
```

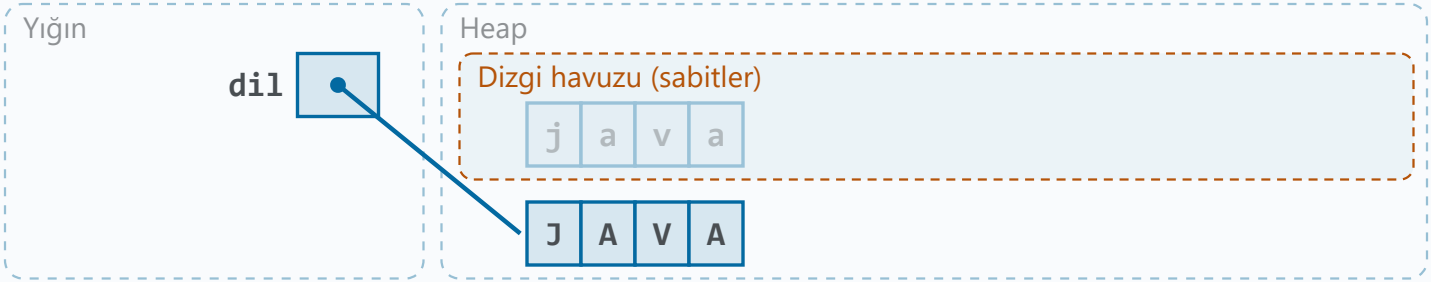
DEĞİŞKENLER

`dil = "JAVA"`

ÇIKTI

```
java
```

Değiştirilemezlik: `toUpperCase()`



JAVA yazar. Değişen nesne değil, değişkenin gösterdiği yerdir.

Adım 5 / 5

```
1 String dil = "java";
2 dil.toUpperCase();
3 System.out.println(dil); // java
4 dil = dil.toUpperCase();
▶ System.out.println(dil); // JAVA
```

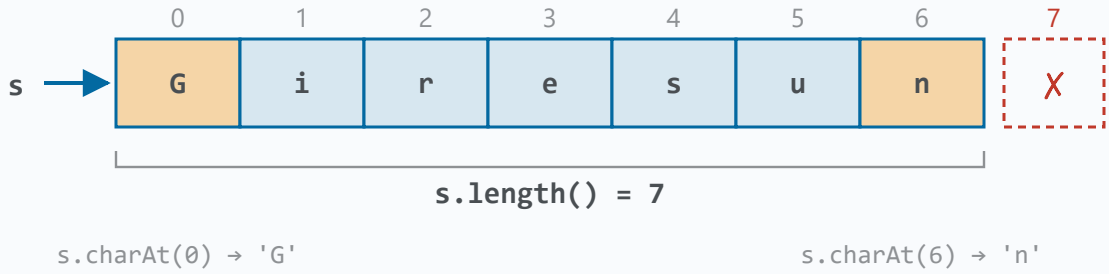
DEĞİŞKENLER

`dil = "JAVA"`

ÇIKTI

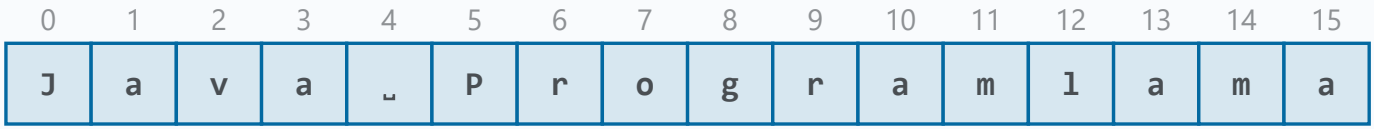
```
java
JAVA
```

İndisleme



- İlk karakter **0**, son karakter **length() − 1** indisindedir; `length()` bir metottur (dizilerdeki `length` alanından farklı).
- `charAt(i)` o konumdaki `char` değerini döndürür; `charAt(7)` → `StringIndexOutOfBoundsException`.

Temel Metotlar



16 karakterlik dizgi (boşluk da bir karakterdir: indis 4).

Adım 1 / 6

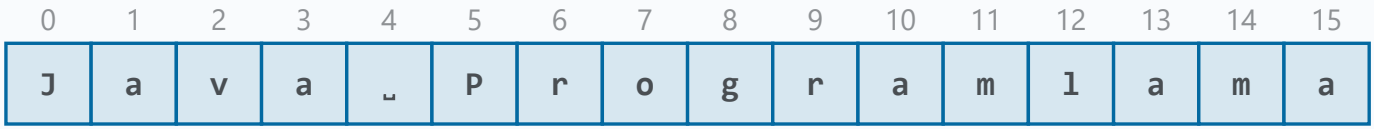
```
▶ String s = "Java Programlama";
2 int uzunluk = s.length();
3 int konum = s.indexOf("Programlama");
4 String alt = s.substring(5, 16);
5 boolean icerir = s.contains("Java");
6 System.out.println(alt + " " + konum);
```

DEĞİŞKENLER

`s = "Java Programlama"`

ÇIKTI

Temel Metotlar



`length()` → **16**; indisler 0 … 15.

Adım 2 / 6

```
1 String s = "Java Programlama";
▶ int uzunluk = s.length();
3 int konum = s.indexOf("Programlama");
4 String alt = s.substring(5, 16);
5 boolean icerir = s.contains("Java");
6 System.out.println(alt + " " + konum);
```

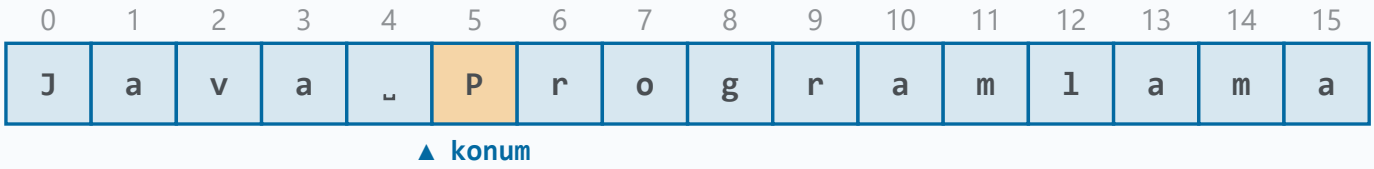
DEĞİŞKENLER

`s = "Java Programlama"`

`uzunluk = 16`

ÇIKTI

Temel Metotlar



`indexOf` aranan metnin **ilk karakterinin** indisini verir → **5** (yoksa −1).

Adım 3 / 6

```
1 String s = "Java Programlama";
2 int uzunluk = s.length();
▶ int konum = s.indexOf("Programlama");
4 String alt = s.substring(5, 16);
5 boolean icerir = s.contains("Java");
6 System.out.println(alt + " " + konum);
```

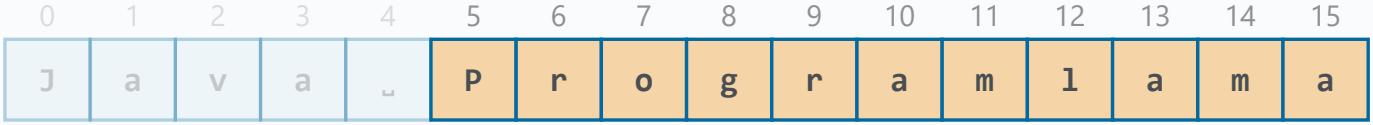
DEĞİŞKENLER

`s = "Java Programlama"`

`uzunluk = 16` `konum = 5`

ÇIKTI

Temel Metotlar



`substring(5, 16)` : 5 **dahil**, 16 **hariç** → 11 karakter: "Programlama".

Adım 4 / 6

```
1 String s = "Java Programlama";
2 int uzunluk = s.length();
3 int konum = s.indexOf("Programlama");
▶ String alt = s.substring(5, 16);
5 boolean icerir = s.contains("Java");
6 System.out.println(alt + " " + konum);
```

DEĞİŞKENLER

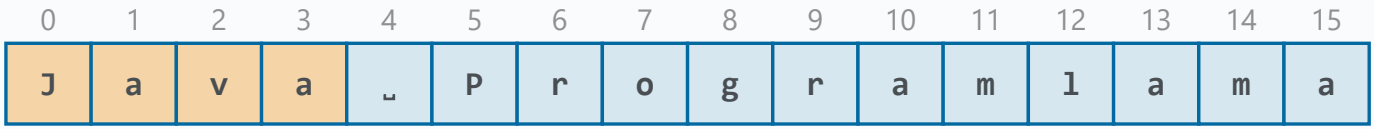
`s = "Java Programlama"`

`uzunluk = 16` `konum = 5`

`alt = "Programlama"`

ÇIKTI

Temel Metotlar



`contains("Java")` → **true**. Hiçbir metot `s` 'yi değiştirmedi.

Adım 5 / 6

```
1 String s = "Java Programlama";
2 int uzunluk = s.length();
3 int konum = s.indexOf("Programlama");
4 String alt = s.substring(5, 16);
▶ boolean icerir = s.contains("Java");
6 System.out.println(alt + " " + konum);
```

DEĞİŞKENLER

`s = "Java Programlama"`

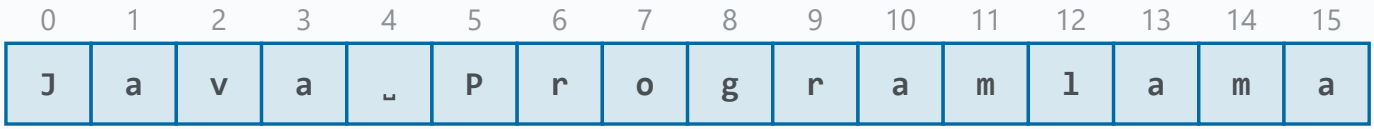
`uzunluk = 16` `konum = 5`

`alt = "Programlama"`

`icerir = true`

ÇIKTI

Temel Metotlar



Sonuçlar birleştirilip yazdırılır.

Adım 6 / 6

```
1 String s = "Java Programlama";
2 int uzunluk = s.length();
3 int konum = s.indexOf("Programlama");
4 String alt = s.substring(5, 16);
5 boolean icerir = s.contains("Java");
▶ System.out.println(alt + " " + konum);
```

DEĞİŞKENLER

`s = "Java Programlama"`

`uzunluk = 16` `konum = 5`

`alt = "Programlama"`

`icerir = true`

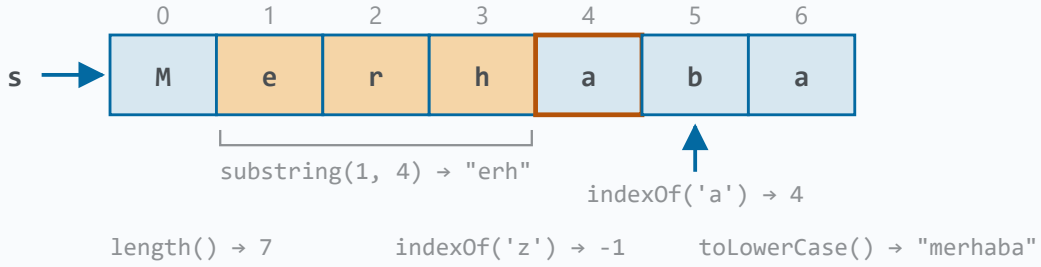
ÇIKTI

```
Programlama 5
```

METOTLAR

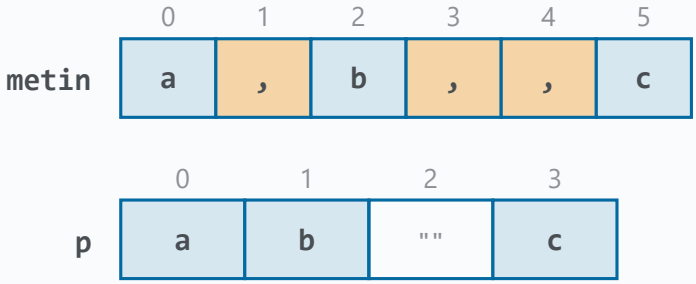
Sık String Metotları

Metot	Sonuç
length()	Karakter/kod birimi sayısı
charAt(i)	i konumundaki char
substring(a,b)	a dahil, b hariç alt dizgi
indexOf(x)	İlk konum veya -1
toLowerCase()	Küçük harfli yeni String
trim() / strip()	Kenar boşlukları kaldırılmış yeni String



METOTLAR · KODU ADIM ADIM ÇALIŞTIR

`split` ve `replace` Adım Adım



virgüllerden 4 parça

`split(",")` metni virgüllerde keser: **"a", "b", "", "c"**. Ardışık iki virgülün arası **boş dizgi** olur ve dizide yer alır.

Adım 1 / 6

```
▶ String[] p = "a,b,,c".split(",");
2 System.out.println(Arrays.toString(p) + " " + p.length);
  // [a, b, , c] 4
3 String s = "ali veli";
4 String t = s.replace('l', 'L');
5 System.out.println(t); // aLi veLi
6 System.out.println(s); // ali veli
```

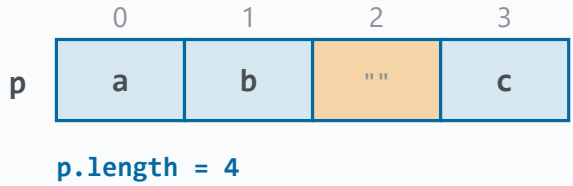
DEĞİŞKENLER

p = [a, b, , c]

ÇIKTI

METOTLAR · KODU ADIM ADIM ÇALIŞTIR

`split` ve `replace` Adım Adım



`Arrays.toString(p)` → **[a, b, , c]**, `p.length` → **4**. (Dizinin *sonundaki* boş parçalar atılır: `"a,b,,".split(",")` yalnızca 2 parça verir.)

Adım 2 / 6

```
1 String[] p = "a,b,,c".split(",");
▶ System.out.println(Arrays.toString(p) + " " + p.length);
  // [a, b, , c] 4
3 String s = "ali veli";
4 String t = s.replace('l', 'L');
5 System.out.println(t); // aLi veLi
6 System.out.println(s); // ali veli
```

DEĞİŞKENLER

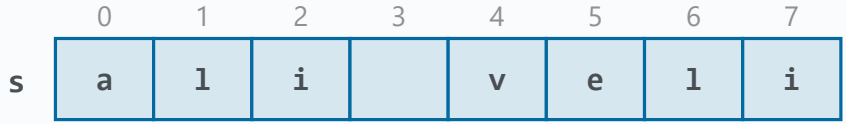
p = **[a, b, , c]**

ÇIKTI

```
[a, b, , c] 4
```

METOTLAR · KODU ADIM ADIM ÇALIŞTIR

`split` ve `replace` Adım Adım



`s` = "ali veli": 8 karakter, 3. indis boşluk.

Adım 3 / 6

```
1 String[] p = "a,b,,c".split(",");
2 System.out.println(Arrays.toString(p) + " " + p.length);
  // [a, b, , c] 4
▶ String s = "ali veli";
4 String t = s.replace('l', 'L');
5 System.out.println(t); // aLi veLi
6 System.out.println(s); // ali veli
```

DEĞİŞKENLER

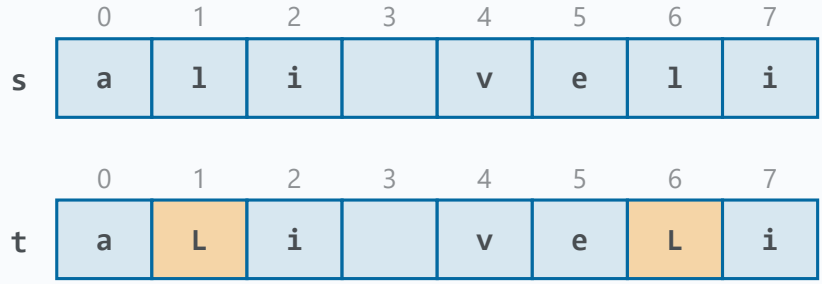
`p = [a, b, , c]` `s = "ali veli"`

ÇIKTI

```
[a, b, , c] 4
```

METOTLAR · KODU ADIM ADIM ÇALIŞTIR

`split` ve `replace` Adım Adım



`replace('l','L')`: yeni nesne

`replace('l', 'L')` her **'l'** karakterini **'L'** yapar (indis 1 ve 6) ve **yeni** bir String döndürür; `s` ise bu satırda değişmez.

Adım 4 / 6

```
1 String[] p = "a,b,,c".split(",");
2 System.out.println(Arrays.toString(p) + " " + p.length);
  // [a, b, , c] 4
3 String s = "ali veli";
▶ String t = s.replace('l', 'L');
5 System.out.println(t); // aLi veLi
6 System.out.println(s); // ali veli
```

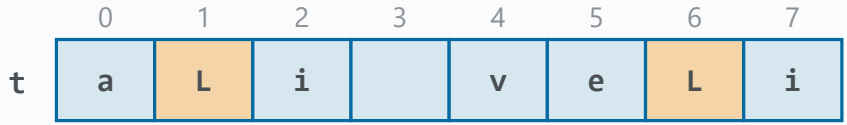
DEĞİŞKENLER

`p = [a, b, , c]` `s = "ali veli"` `t = "aLi veLi"`

ÇIKTI

```
[a, b, , c] 4
```

`split` ve `replace` Adım Adım



`t` yazdırılır: **aLi veLi**.

Adım 5 / 6

```
1 String[] p = "a,b,,c".split(",");
2 System.out.println(Arrays.toString(p) + " " + p.length);
  // [a, b, , c] 4
3 String s = "ali veli";
4 String t = s.replace('l', 'L');
▶ System.out.println(t); // aLi veLi
6 System.out.println(s); // ali veli
```

DEĞİŞKENLER

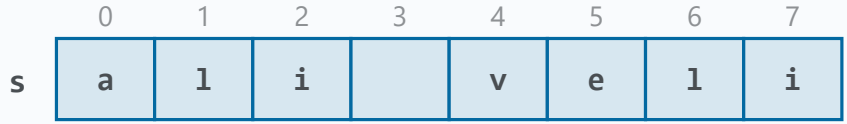
`p = [a, b, , c]` `s = "ali veli"` `t = "aLi veLi"`

ÇIKTI

```
[a, b, , c] 4
aLi veLi
```

METOTLAR · KODU ADIM ADIM ÇALIŞTIR

`split` ve `replace` Adım Adım



`s` hâlâ **ali veli**: String değiştirilemezdir, sonuç ancak dönen değerde bulunur.

Adım 6 / 6

```
1 String[] p = "a,b,,c".split(",");
2 System.out.println(Arrays.toString(p) + " " + p.length);
  // [a, b, , c] 4
3 String s = "ali veli";
4 String t = s.replace('l', 'L');
5 System.out.println(t); // aLi veLi
▶ System.out.println(s); // ali veli
```

DEĞİŞKENLER

`p = [a, b, , c]` `s = "ali veli"` `t = "aLi veLi"`

ÇIKTI

```
[a, b, , c] 4
aLi veLi
ali veli
```

METOTLAR

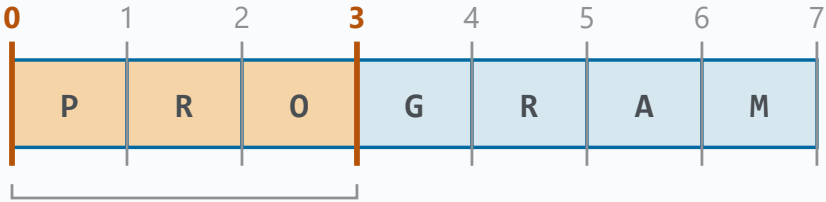
replace, replaceAll, join, valueOf, format

Çağrı	Sonuç
`"banana".replace('a', 'o')`	`bonono`
`"banana".replace("an", "AN")`	`bANANa`
`"a1b22c333".replaceAll("[0-9]+", "#")`	`a#b#c#`
`"a1b22c333".replaceFirst("[0-9]+", "#")`	`a#b22c333`
`"Ada Can Ece".replaceAll(" +", " ")`	`Ada Can Ece`
`String.join(", ", "Ada", "Can", "Ece")`	`Ada, Can, Ece`
`String.valueOf(3.5)` / `String.valueOf(new char[]{'h','i'})`	`3.5` / `hi`
`String.format(Locale.US, "%.2f", 3.14159)`	`3.14`
`String.format(Locale.US, "%5d|%-5d|%05d", 42, 42, 42)`	`  42|42 |00042`
`String.format(Locale.US, "%,d", 1234567)`	`1,234,567`

`replace` düz metin, `replaceAll` düzenli ifade arar; hepsi **yeni String** döndürür. Yerel ayar verilmezse `%.2f` Türkçe ayarlı bilgisayarda `3,14` yazar.

METOTLAR

Alt Dizgi Sınırları: `s = "PROGRAM"`



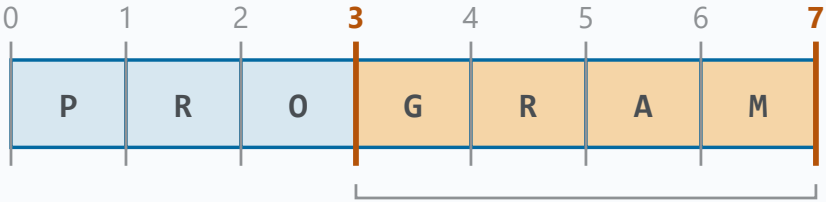
`s.substring(0, 3) → "PRO"`

İndisler karakterlerin **arasındaki çizgiler** gibi düşünülebilir: 0 ile 3 arası "PRO". Uzunluk = 3 − 0.

Adım 1 / 5

- `substring(a, b)` : **a dahil, b hariç**; sonuç uzunluğu **b − a**.
- Geçerli aralık: 0 ≤ a ≤ b ≤ length().

Alt Dizgi Sınırları: `s = "PROGRAM"`

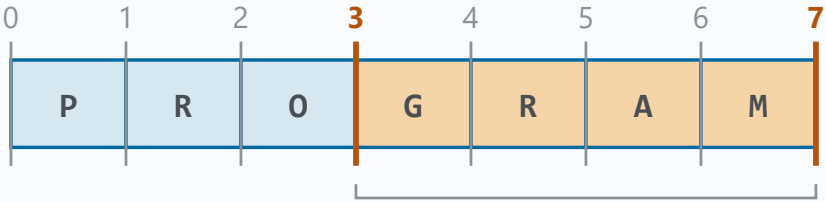


`s.substring(3, 7)` → `"GRAM"`

Bitiş `length()` = 7 olabilir: son karakter dahil edilir.

- `substring(a, b)` : **a dahil, b hariç**; sonuç uzunluğu **b − a**.
- Geçerli aralık: 0 ≤ a ≤ b ≤ length().

Alt Dizgi Sınırları: `s = "PROGRAM"`

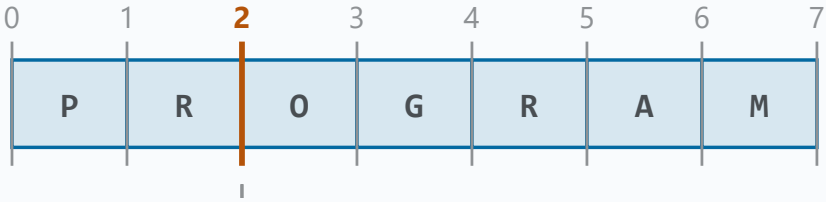


`s.substring(3) → "GRAM"`

Tek parametreli sürüm: başlangıçtan **sona kadar**.

- `substring(a, b)` : **a dahil, b hariç**; sonuç uzunluğu **b − a**.
- Geçerli aralık: 0 ≤ a ≤ b ≤ length().

Alt Dizgi Sınırları: `s = "PROGRAM"`

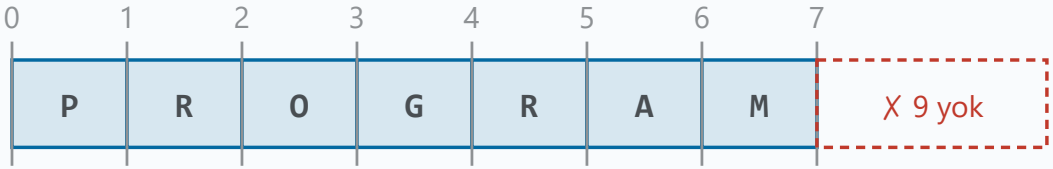


`s.substring(2, 2)` → `""`

Başlangıç = bitiş → boş dizgi `""` (hata değil).

- `substring(a, b)` : **a dahil, b hariç**; sonuç uzunluğu **b − a**.
- Geçerli aralık: 0 ≤ a ≤ b ≤ length().

Alt Dizgi Sınırları: `s = "PROGRAM"`

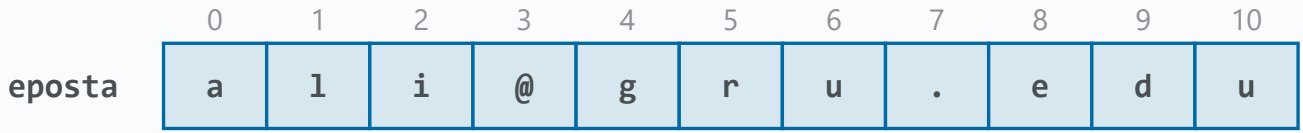


`s.substring(5, 9)` → `StringIndexOutOfBoundsException`

Bitiş > length() ya da başlangıç > bitiş → çalışma zamanı hatası.

- `substring(a, b)` : **a dahil, b hariç**; sonuç uzunluğu **b − a**.
- Geçerli aralık: 0 ≤ a ≤ b ≤ length().

Uygulama: E-postayı Parçalama



11 karakterlik e-posta adresi; kullanıcı adını ve alan adını ayıracağız.

Adım 1 / 7

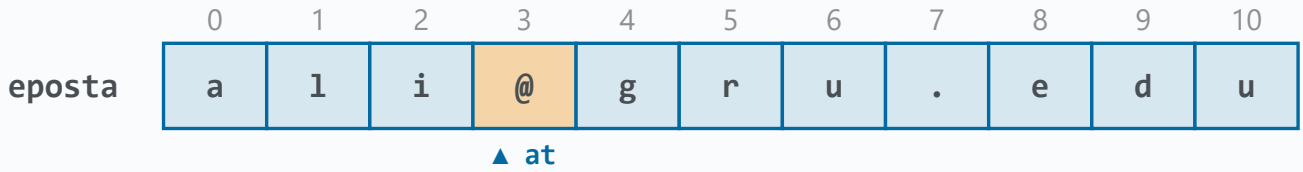
```
String eposta = "ali@gru.edu";
int at = eposta.indexOf('@');
String kullanici = eposta.substring(0, at);
String alan = eposta.substring(at + 1);
char ilk = eposta.charAt(0);
System.out.println(kullanici + " / " + alan);
System.out.println(Character.toUpperCase(ilk));
```

DEĞİŞKENLER

eposta = "ali@gru.edu"

ÇIKTI

Uygulama: E-postayı Parçalama



`indexOf('@')` → **3**: ayırıcının konumu.

Adım 2 / 7

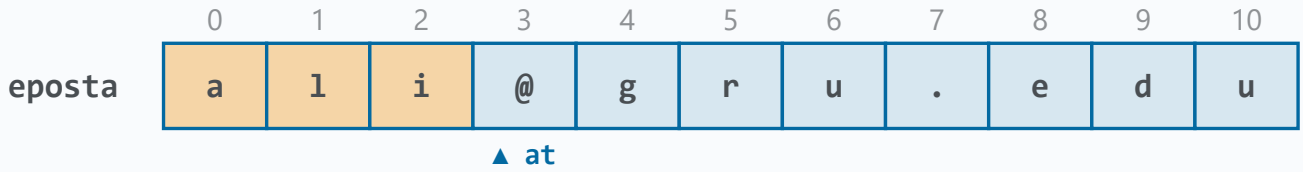
```
1 String eposta = "ali@gru.edu";
▶ int at = eposta.indexOf('@');
3 String kullanici = eposta.substring(0, at);
4 String alan = eposta.substring(at + 1);
5 char ilk = eposta.charAt(0);
6 System.out.println(kullanici + " / " + alan);
7 System.out.println(Character.toUpperCase(ilk));
```

DEĞİŞKENLER

`eposta = "ali@gru.edu"` `at = 3`

ÇIKTI

Uygulama: E-postayı Parçalama



`substring(0, 3)` : 0 dahil, 3 (@) hariç → "ali".

Adım 3 / 7

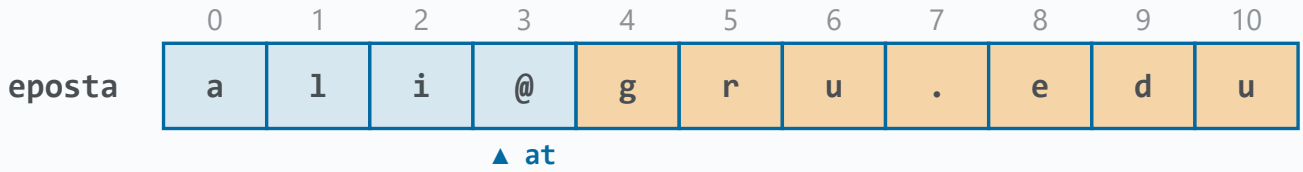
```
1 String eposta = "ali@gru.edu";
2 int at = eposta.indexOf('@');
▶ String kullanici = eposta.substring(0, at);
4 String alan = eposta.substring(at + 1);
5 char ilk = eposta.charAt(0);
6 System.out.println(kullanici + " / " + alan);
7 System.out.println(Character.toUpperCase(ilk));
```

DEĞİŞKENLER

`eposta = "ali@gru.edu"` `at = 3` `kullanici = "ali"`

ÇIKTI

Uygulama: E-postayı Parçalama



`substring(4)` : @'den sonraki karakterden sona kadar → "gru.edu".

Adım 4 / 7

```
1 String eposta = "ali@gru.edu";
2 int at = eposta.indexOf('@');
3 String kullanici = eposta.substring(0, at);
▶ String alan = eposta.substring(at + 1);
5 char ilk = eposta.charAt(0);
6 System.out.println(kullanici + " / " + alan);
7 System.out.println(Character.toUpperCase(ilk));
```

DEĞİŞKENLER

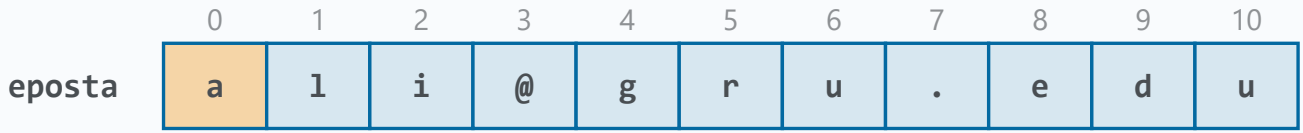
`eposta = "ali@gru.edu"` `at = 3`

`kullanici = "ali"`

`alan = "gru.edu"`

ÇIKTI

Uygulama: E-postayı Parçalama



`charAt(0)` bir `char` döndürür: **'a'** (tek tırnak).

Adım 5 / 7

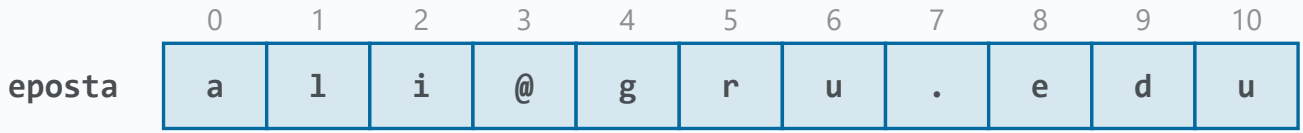
```
1 String eposta = "ali@gru.edu";
2 int at = eposta.indexOf('@');
3 String kullanici = eposta.substring(0, at);
4 String alan = eposta.substring(at + 1);
▶ char ilk = eposta.charAt(0);
6 System.out.println(kullanici + " / " + alan);
7 System.out.println(Character.toUpperCase(ilk));
```

DEĞİŞKENLER

`eposta = "ali@gru.edu"` `at = 3` `kullanici = "ali"` `alan = "gru.edu"` `ilk = 'a'`

ÇIKTI

Uygulama: E-postayı Parçalama



Parçalar birleştirilerek yazdırılır.

Adım 6 / 7

```
1 String eposta = "ali@gru.edu";
2 int at = eposta.indexOf('@');
3 String kullanici = eposta.substring(0, at);
4 String alan = eposta.substring(at + 1);
5 char ilk = eposta.charAt(0);
▶ System.out.println(kullanici + " / " + alan);
7 System.out.println(Character.toUpperCase(ilk));
```

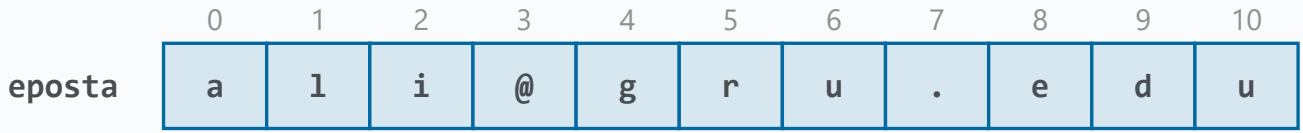
DEĞİŞKENLER

`eposta = "ali@gru.edu"` `at = 3` `kullanici = "ali"` `alan = "gru.edu"` `ilk = 'a'`

ÇIKTI

```
ali / gru.edu
```

Uygulama: E-postayı Parçalama



`Character.toUpperCase` char'ı büyük harfe çevirir → **A**. `eposta` hiç değişmedi.

Adım 7 / 7

```
1 String eposta = "ali@gru.edu";
2 int at = eposta.indexOf('@');
3 String kullanici = eposta.substring(0, at);
4 String alan = eposta.substring(at + 1);
5 char ilk = eposta.charAt(0);
6 System.out.println(kullanici + " / " + alan);
▶ System.out.println(Character.toUpperCase(ilk));
```

DEĞİŞKENLER

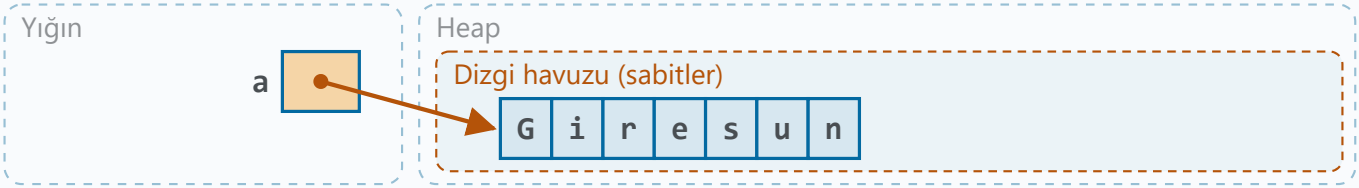
`eposta = "ali@gru.edu"` `at = 3` `kullanici = "ali"` `alan = "gru.edu"` `ilk = 'a'`

ÇIKTI

```
ali / gru.edu
A
```

KARŞILAŞTIRMA · KODU ADIM ADIM ÇALIŞTIR

`==` ile `equals` Farkı



Sabit `"Giresun"` dizgi havuzuna konur; `a` onu gösterir.

Adım 1 / 6

```
▶ String a = "Giresun";
2 String b = new String("Giresun");
3 String c = "Giresun";
4 System.out.println(a == b);      // false
5 System.out.println(a == c);      // true
6 System.out.println(a.equals(b)); // true
```

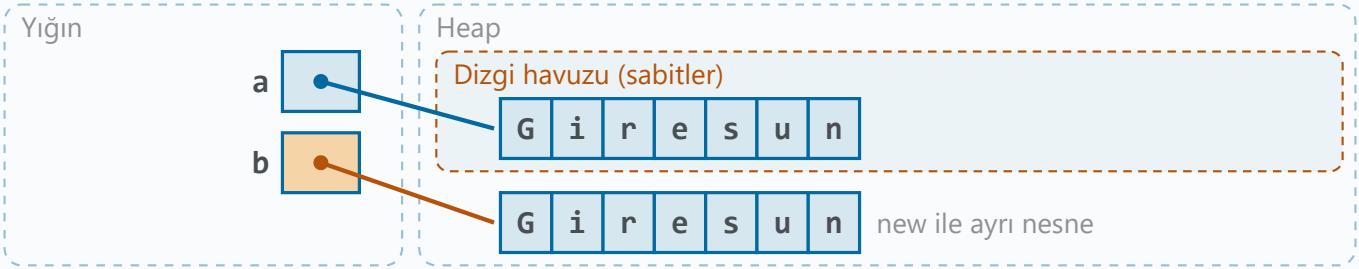
DEĞİŞKENLER

`a = → havuz`

ÇIKTI

KARŞILAŞTIRMA · KODU ADIM ADIM ÇALIŞTIR

== ile equals Farkı



`new String(...)` içerik aynı olsa da heap'te **ayrı** bir nesne açar.

Adım 2 / 6

```
1 String a = "Giresun";
▶ String b = new String("Giresun");
3 String c = "Giresun";
4 System.out.println(a == b);      // false
5 System.out.println(a == c);      // true
6 System.out.println(a.equals(b)); // true
```

DEĞİŞKENLER

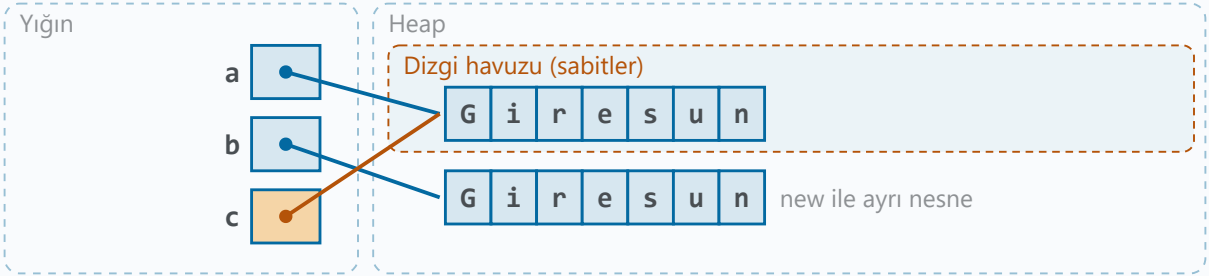
a = → havuz

b = → yeni nesne

ÇIKTI

KARŞILAŞTIRMA · KODU ADIM ADIM ÇALIŞTIR

== ile equals Farkı



Aynı sabit tekrar yazıldı: havuzdaki **mevcut** nesne paylaşılır → c ile a aynı oku taşır.

Adım 3 / 6

```
1 String a = "Giresun";
2 String b = new String("Giresun");
▶ String c = "Giresun";
4 System.out.println(a == b);      // false
5 System.out.println(a == c);      // true
6 System.out.println(a.equals(b)); // true
```

DEĞİŞKENLER

a = → havuz

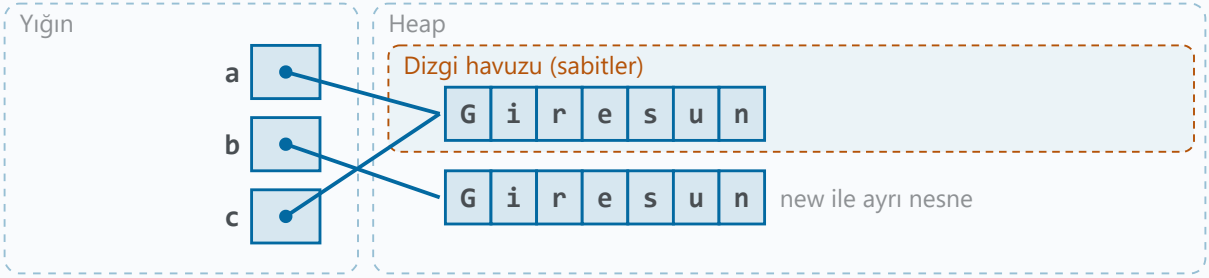
b = → yeni nesne

c = → havuz

ÇIKTI

KARŞILAŞTIRMA · KODU ADIM ADIM ÇALIŞTIR

== ile equals Farkı



`a == b` : iki ok **farklı** nesneye gidiyor → **false** (içeriğe bakılmaz).

Adım 4 / 6

```
1 String a = "Giresun";
2 String b = new String("Giresun");
3 String c = "Giresun";
▶ System.out.println(a == b);       // false
5 System.out.println(a == c);       // true
6 System.out.println(a.equals(b)); // true
```

DEĞİŞKENLER

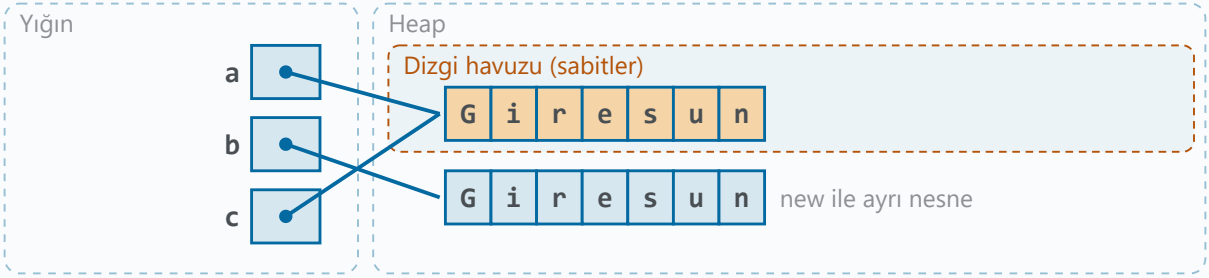
`a = → havuz` `b = → yeni nesne` `c = → havuz`

ÇIKTI

```
false
```

KARŞILAŞTIRMA · KODU ADIM ADIM ÇALIŞTIR

== ile equals Farkı



`a == c` : iki ok **aynı** nesnede → **true**.

Adım 5 / 6

```
1 String a = "Giresun";
2 String b = new String("Giresun");
3 String c = "Giresun";
4 System.out.println(a == b);       // false
▶ System.out.println(a == c);       // true
6 System.out.println(a.equals(b)); // true
```

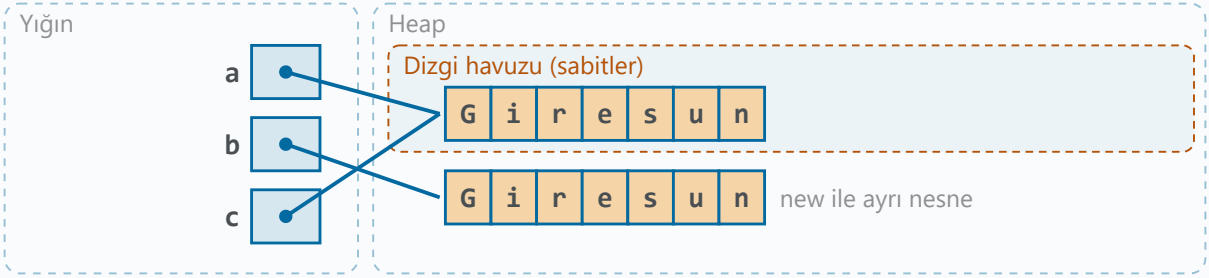
DEĞİŞKENLER

`a = → havuz` `b = → yeni nesne` `c = → havuz`

ÇIKTI

```
false
true
```

== ile equals Farkı



`equals` karakterleri tek tek karşılaştırır: G=G, i=i, … → **true**. İçerik için daima `equals`.

Adım 6 / 6

```
1 String a = "Giresun";
2 String b = new String("Giresun");
3 String c = "Giresun";
4 System.out.println(a == b);      // false
5 System.out.println(a == c);      // true
▶ System.out.println(a.equals(b)); // true
```

DEĞİŞKENLER

`a = → havuz` `b = → yeni nesne` `c = → havuz`

ÇIKTI

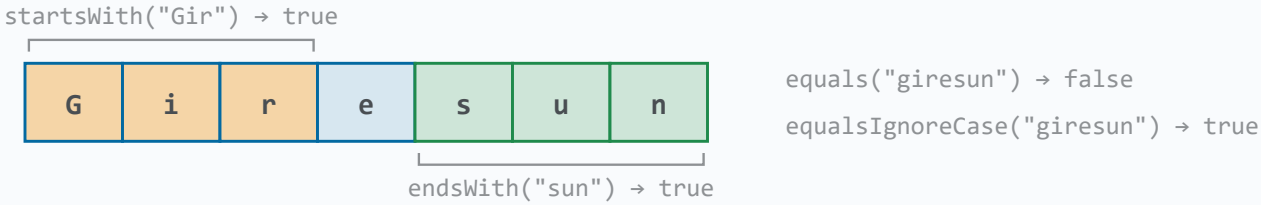
```
false
true
true
```

KARŞILAŞTIRMA

Karşılaştırma Metotları

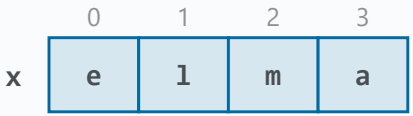
Metot	Amaç	Örnek
`equals`	İçerik eşitliği	`"abc".equals("abc") → true`
`equalsIgnoreCase`	Büyük-küçük harfi yok sayan eşitlik	`"Java".equalsIgnoreCase("JAVA") → true`
`startsWith`	Belirli metinle başlama	`"Giresun".startsWith("Gir") → true`
`endsWith`	Belirli metinle bitme	`"Giresun".endsWith("sun") → true`
`compareTo`	Sözlüksel sıra: <0, 0, >0	`"elma".compareTo("armut") → pozitif`

compareTo Unicode değerlerine göre çalışır: büyük harfler küçüklerden önce gelir ve Türkçe harfler (ç, ğ, ı, ö, ş, ü) alfabe sırasına uymayabilir.



KARŞILAŞTIRMA · KODU ADIM ADIM ÇALIŞTIR

`compareTo`: Karakter ve Uzunluk Farkı



Önce `x` = "elma" (4 karakter). Karşılaştırma bu karakterlerin Unicode kodlarına göre yapılır.

Adım 1 / 6

```
▶ String x = "elma";
2 String y = "armut";
3 System.out.println(x.compareTo(y)); // 4
4 String k = "kedi";
5 System.out.println(k.compareTo("kediler")); // -3
```

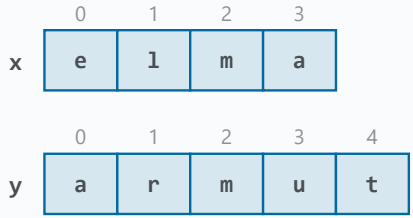
DEĞİŞKENLER

`x = "elma"`

ÇIKTI

KARŞILAŞTIRMA · KODU ADIM ADIM ÇALIŞTIR

`compareTo`: Karakter ve Uzunluk Farkı



`y` = "armut" (5 karakter). Sıradaki satırda `x.compareTo(y)` hesaplanacak.

Adım 2 / 6

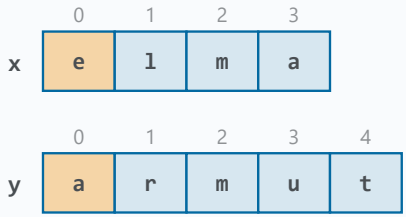
```
1 String x = "elma";
▶ String y = "armut";
3 System.out.println(x.compareTo(y)); // 4
4 String k = "kedi";
5 System.out.println(k.compareTo("kediler")); // -3
```

DEĞİŞKENLER

x = "elma" y = "armut"

ÇIKTI

`compareTo`: Karakter ve Uzunluk Farkı



`'e' (101) - 'a' (97) = 4`

İlk farklı konum 0: **'e' - 'a' = 101 - 97 = 4**. Pozitif: "elma" sözlükte "armut"tan sonra gelir.

Adım 3 / 6

```
1 String x = "elma";
2 String y = "armut";
▶ System.out.println(x.compareTo(y)); // 4
4 String k = "kedi";
5 System.out.println(k.compareTo("kediler")); // -3
```

DEĞİŞKENLER

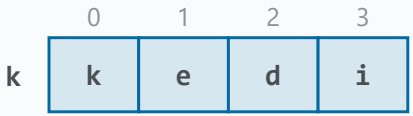
`x = "elma"` `y = "armut"`

ÇIKTI

```
4
```

KARŞILAŞTIRMA · KODU ADIM ADIM ÇALIŞTIR

`compareTo`: Karakter ve Uzunluk Farkı



`k` = "kedi" (4 karakter). Şimdi bir dizgi diğerinin başlangıcıysa ne olduğuna bakalım.

Adım 4 / 6

```
1 String x = "elma";
2 String y = "armut";
3 System.out.println(x.compareTo(y)); // 4
▶ String k = "kedi";
5 System.out.println(k.compareTo("kediler")); // -3
```

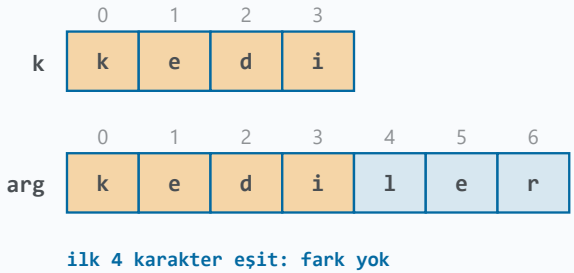
DEĞİŞKENLER

`x = "elma"` `y = "armut"` `k = "kedi"`

ÇIKTI

```
4
```

`compareTo`: Karakter ve Uzunluk Farkı



`k.compareTo("kediler")` : ilk 4 karakter **aynı** ve "kedi" bitiyor; sonuç **uzunluk farkıdır**.

Adım 5 / 6

```
1 String x = "elma";
2 String y = "armut";
3 System.out.println(x.compareTo(y)); // 4
4 String k = "kedi";
▶ System.out.println(k.compareTo("kediler")); // -3
```

DEĞİŞKENLER

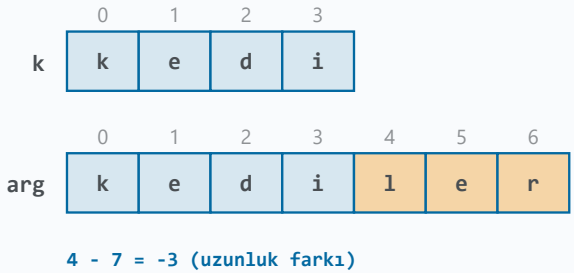
`x = "elma"` `y = "armut"` `k = "kedi"`

ÇIKTI

```
4
```

KARŞILAŞTIRMA · KODU ADIM ADIM ÇALIŞTIR

`compareTo`: Karakter ve Uzunluk Farkı



4 - 7 = -3: kısa olan "kedi" önce gelir. Kural: farklı karakter varsa karakter farkı, yoksa uzunluk farkı.

Adım 6 / 6

```
1 String x = "elma";
2 String y = "armut";
3 System.out.println(x.compareTo(y)); // 4
4 String k = "kedi";
▶ System.out.println(k.compareTo("kediler")); // -3
```

DEĞİŞKENLER

`x = "elma"` `y = "armut"` `k = "kedi"`

ÇIKTI

```
4
-3
```

KARŞILAŞTIRMA

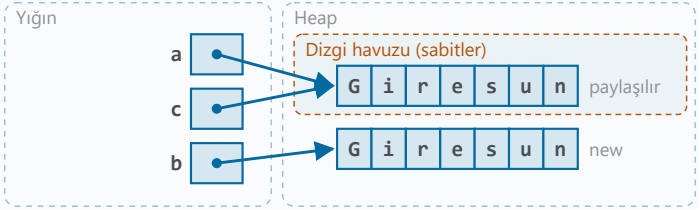
Türkçe Karakter ve Locale

Çağrı (tr = Locale.forLanguageTag("tr"))	Sonuç
`"TITLE".toLowerCase(Locale.ENGLISH)`	`title`
`"TITLE".toLowerCase(Locale.forLanguageTag("tr"))`	`tıtle`
`"i".toUpperCase(tr)`	`İ`
`"i".toUpperCase(Locale.ENGLISH)`	`I`
`"I".toLowerCase(tr)`	`ı`
`"ISPARTA".toLowerCase(tr)`	`ısparta`
`"ısparta".toUpperCase(tr)`	`ISPARTA`

- Parametresiz `toLowerCase()` makinenin **varsayılan diline** bağlıdır: aynı kod başka bilgisayarda başka sonuç verebilir.
- Anahtar sözcük, dosya uzantısı gibi teknik metinde `Locale.ENGLISH`, Türkçe metinde `Locale.forLanguageTag("tr")` verilir.
- `"TITLE".toLowerCase(tr).equals("title")` → `false`; `"title".equalsIgnoreCase("TITLE")` → `true`.

Dizgi Havuzu

- Aynı dizgi sabitleri havuzdaki **tek bir** nesneyi paylaşır (bellek tasarrufu; değiştirilemezlik sayesinde güvenli).
- `new String(...)` her zaman ayrı nesne oluşturur.
- Bir String'in havuzdan mı geldiği koda bakınca her zaman anlaşılmaz; bu yüzden içerik karşılaştırmasında `==` değil **`equals`** kullanılır.



BELLEK

Sabit Birleştirme ve intern

```
String a = "a", b = "b";
String r = a + b;
final String f = "a";
System.out.println(("a" + "b") == "ab");  // true
System.out.println((f + "b") == "ab");    // true
System.out.println(r == "ab");            // false
System.out.println(r.equals("ab"));       // true
System.out.println(r.intern() == "ab");   // true
```

Neden?

Sabit ifadeleri (`"a" + "b"` , `final` değişken) derleyici birleştirir; sonuç havuzdaki `"ab"` 'dir. `a + b` çalışma zamanında **yeni nesne** üretir; `intern()` havuzdaki nesneyi verir.

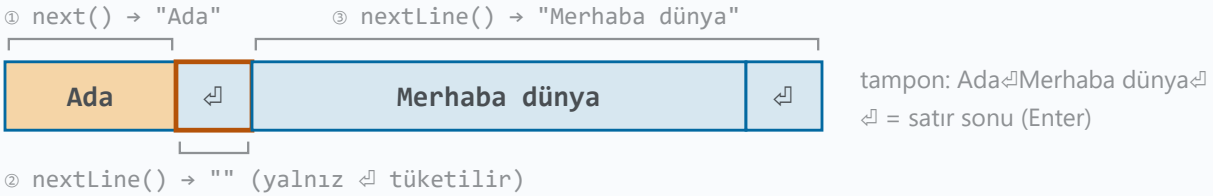
Null güvenli karşılaştırma

`String x = null;` iken `"sabit".equals(x)` → `false` , ama `x.equals("sabit")` → **NullPointerException**. Bu yüzden sabit değer sola yazılır.

Scanner ile Metin Girdisi

```
import java.util.Scanner;
// ...
Scanner klavye = new Scanner(System.in);
String kelime = klavye.next();
klavye.nextLine();
String satir = klavye.nextLine();
```

- `next()` bir kelime okur: baştaki boşlukları atlar, ilk boşlukta/satır sonunda durur.
- `nextLine()` satır sonuna (⏎) kadar her şeyi okur ve ⏎'ı tüketir.
- `next()` veya `nextInt()` sonrası tamponda kalan ⏎, bir `nextLine()` ile ayrıca tüketilmelidir.



GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: `next()` ve `nextLine()`

Klavye tamponu (System.in)



▲ okuma konumu

Scanner klavye akışına bağlanır; tampon şimdilik boş.

Adım 1 / 7

```
▶ Scanner klavye = new Scanner(System.in);
2 String kelime = klavye.next();
3 klavye.nextLine();
4 String satir = klavye.nextLine();
5 System.out.println(kelime + " | " + satir);
```

DEĞİŞKENLER

—

ÇIKTI

3. satır olmasaydı 4. satırdaki `nextLine()` kalan ⏎'ı okuyup hemen **boş** satır döndürürdü.

Kod İzleme: `next()` ve `nextLine()`

Klavye tamponu (System.in)



`next()` okuyacak bir şey bulamaz ve **bekler**. Kullanıcı **Ada** yazıp Enter'a basar → tampona "Ada⏎" girer.

Adım 2 / 7

```
1 Scanner klavye = new Scanner(System.in);
▶ String kelime = klavye.next();
3 klavye.nextLine();
4 String satir = klavye.nextLine();
5 System.out.println(kelime + " | " + satir);
```

DEĞİŞKENLER

—

ÇIKTI

3. satır olmasaydı 4. satırdaki `nextLine()` kalan ⏎'ı okuyup hemen **boş** satır döndürürdü.

Kod İzleme: `next()` ve `nextLine()`

Klavye tamponu (System.in)



▲ okuma konumu

`next()` boşluk/satır sonuna kadar okur → "Ada". ⏎ **tamponda kalır.**

Adım 3 / 7

```
1 Scanner klavye = new Scanner(System.in);
▶ String kelime = klavye.next();
3 klavye.nextLine();
4 String satir = klavye.nextLine();
5 System.out.println(kelime + " | " + satir);
```

DEĞİŞKENLER

`kelime = "Ada"`

ÇIKTI

3. satır olmasaydı 4. satırdaki `nextLine()` kalan ⏎'ı okuyup hemen **boş** satır döndürürdü.

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: `next()` ve `nextLine()`

Klavye tamponu (System.in)



▲ okuma konumu

`nextLine()` ⏎'a kadar okur → "" (boş) döner ve ⏎'ı tüketir. Artık tampon temiz.

Adım 4 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 String kelime = klavye.next();
▶ klavye.nextLine();
4 String satir = klavye.nextLine();
5 System.out.println(kelime + " | " + satir);
```

DEĞİŞKENLER

`kelime = "Ada"`

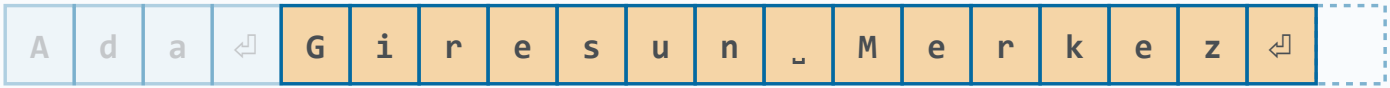
ÇIKTI

3. satır olmasaydı 4. satırdaki `nextLine()` kalan ⏎'ı okuyup hemen **boş** satır döndürürdü.

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: `next()` ve `nextLine()`

Klavye tamponu (System.in)



Tampon boş → `nextLine()` bekler. Kullanıcı **Giresun Merkez** yazıp Enter'a basar.

Adım 5 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 String kelime = klavye.next();
3 klavye.nextLine();
▶ String satir = klavye.nextLine();
5 System.out.println(kelime + " | " + satir);
```

DEĞİŞKENLER

`kelime = "Ada"`

ÇIKTI

3. satır olmasaydı 4. satırdaki `nextLine()` kalan ⏎'ı okuyup hemen **boş** satır döndürürdü.

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: `next()` ve `nextLine()`

Klavye tamponu (System.in)



Satırın tamamı (boşluk dahil) okunur → "Giresun Merkez"; ⏎ tüketilir ama sonuca eklenmez.

Adım 6 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 String kelime = klavye.next();
3 klavye.nextLine();
▶ String satir = klavye.nextLine();
5 System.out.println(kelime + " | " + satir);
```

DEĞİŞKENLER

`kelime = "Ada"`

`satir = "Giresun Merkez"`

ÇIKTI

3. satır olmasaydı 4. satırdaki `nextLine()` kalan ⏎'ı okuyup hemen **boş** satır döndürürdü.

GİRDİ · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: `next()` ve `nextLine()`

Klavye tamponu (System.in)



İki değer yazdırılır.

Adım 7 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 String kelime = klavye.next();
3 klavye.nextLine();
4 String satir = klavye.nextLine();
▶ System.out.println(kelime + " | " + satir);
```

DEĞİŞKENLER

`kelime = "Ada"`

`satir = "Giresun Merkez"`

ÇIKTI

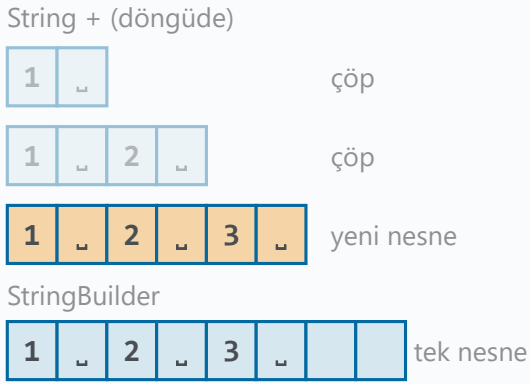
```
Ada | Giresun Merkez
```

3. satır olmasaydı 4. satırdaki `nextLine()` kalan ⏎'ı okuyup hemen **boş** satır döndürürdü.

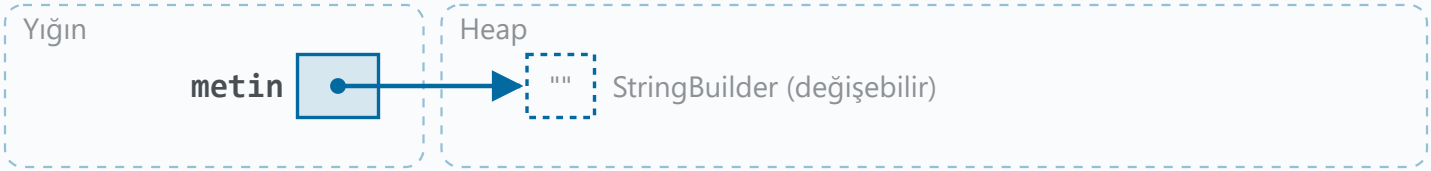
PERFORMANS

Birleştirme ve StringBuilder

- `+` az sayıda dizgi birleştirme için okunaklıdır.
- Döngüde `s = s + x` her turda **yeni** String oluşturur ve eskiyi kopyalar.
- `StringBuilder` değiştirilebilir bir tampondur: `append` aynı nesneye ekler.
- `toString()` tamponun o anki içeriğinden son String'i üretir.



Kod İzleme: `StringBuilder` ile Birleştirme



Boş, değiştirilebilir bir tampon oluşturulur.

Adım 1 / 10

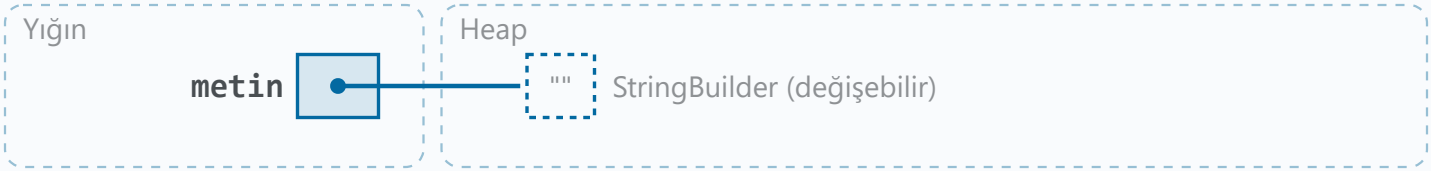
```
▶ StringBuilder metin = new StringBuilder();
2 for (int i = 1; i <= 3; i++) {
3     metin.append(i).append(' ');
4 }
5 String sonuc = metin.toString();
6 System.out.println(sonuc + "|");
```

DEĞİŞKENLER

`metin = ""`

ÇIKTI

Kod İzleme: `StringBuilder` ile Birleştirme



1 <= 3 → true

Adım 2 / 10

```
1 StringBuilder metin = new StringBuilder();
▶ for (int i = 1; i <= 3; i++) {
3     metin.append(i).append(' ');
4 }
5 String sonuc = metin.toString();
6 System.out.println(sonuc + "|");
```

DEĞİŞKENLER

`metin = ""` `i = 1`

ÇIKTI

Kod İzleme: `StringBuilder` ile Birleştirme



`append(1)` ve `append(' ')` **aynı nesneye** ekler (yeni String yok) → "1 ".

Adım 3 / 10

```
1 StringBuilder metin = new StringBuilder();
2 for (int i = 1; i <= 3; i++) {
▶     metin.append(i).append(' ');
4 }
5 String sonuc = metin.toString();
6 System.out.println(sonuc + "|");
```

DEĞİŞKENLER

`metin = "1 "` `i = 1`

ÇIKTI

Kod İzleme: `StringBuilder` ile Birleştirme



`i++` → **2 <= 3 → true**

Adım 4 / 10

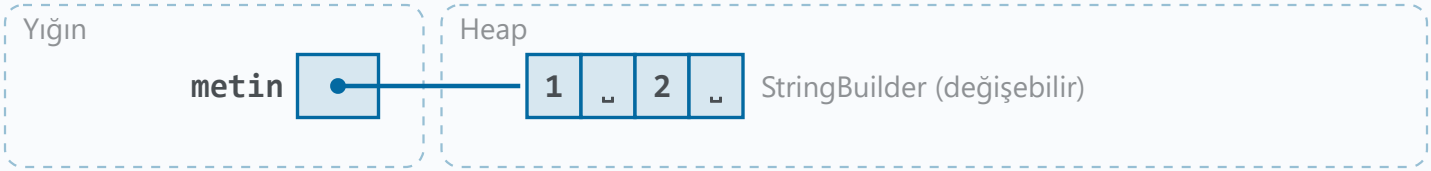
```
1 StringBuilder metin = new StringBuilder();
▶ for (int i = 1; i <= 3; i++) {
3     metin.append(i).append(' ');
4 }
5 String sonuc = metin.toString();
6 System.out.println(sonuc + "|");
```

DEĞİŞKENLER

`metin = "1 "` `i = 2`

ÇIKTI

Kod İzleme: `StringBuilder` ile Birleştirme



`append(2)` ve `append(' ')` **aynı nesneye** ekler (yeni String yok) → "1 2 ".

Adım 5 / 10

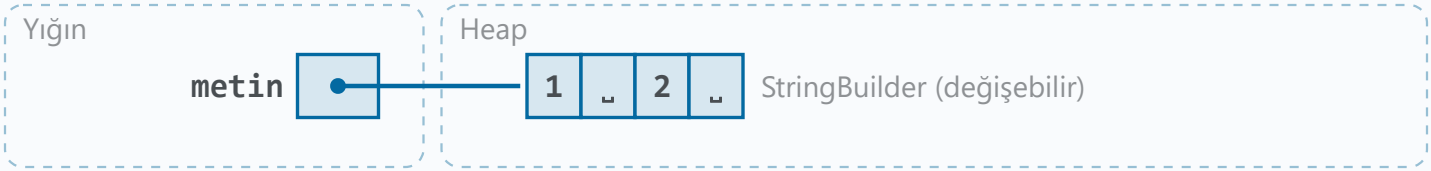
```
1 StringBuilder metin = new StringBuilder();
2 for (int i = 1; i <= 3; i++) {
▶     metin.append(i).append(' ');
4 }
5 String sonuc = metin.toString();
6 System.out.println(sonuc + "|");
```

DEĞİŞKENLER

`metin = "1 2 "` `i = 2`

ÇIKTI

Kod İzleme: `StringBuilder` ile Birleştirme



`i++` → 3 <= 3 → true

Adım 6 / 10

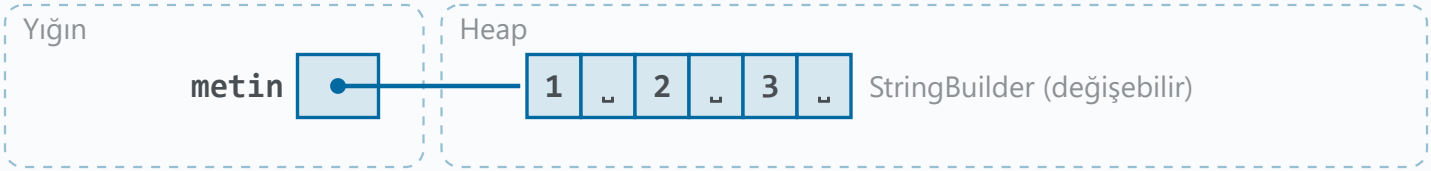
```
1 StringBuilder metin = new StringBuilder();
▶ for (int i = 1; i <= 3; i++) {
3     metin.append(i).append(' ');
4 }
5 String sonuc = metin.toString();
6 System.out.println(sonuc + "|");
```

DEĞİŞKENLER

`metin = "1 2 "` `i = 3`

ÇIKTI

Kod İzleme: `StringBuilder` ile Birleştirme



`append(3)` ve `append(' ')` **aynı nesneye** ekler (yeni String yok) → "1 2 3 ".

Adım 7 / 10

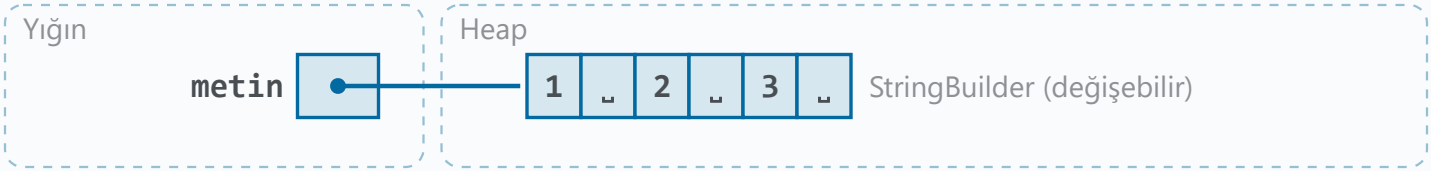
```
1 StringBuilder metin = new StringBuilder();
2 for (int i = 1; i <= 3; i++) {
▶     metin.append(i).append(' ');
4 }
5 String sonuc = metin.toString();
6 System.out.println(sonuc + "|");
```

DEĞİŞKENLER

`metin = "1 2 3 "` `i = 3`

ÇIKTI

Kod İzleme: `StringBuilder` ile Birleştirme



`i++` → **4 <= 3** → **false**: döngü biter.

Adım 8 / 10

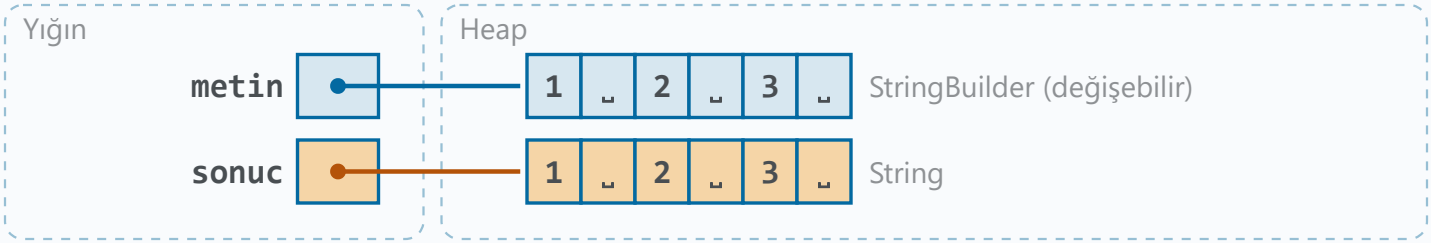
```
1 StringBuilder metin = new StringBuilder();
▶ for (int i = 1; i <= 3; i++) {
3     metin.append(i).append(' ');
4 }
5 String sonuc = metin.toString();
6 System.out.println(sonuc + "|");
```

DEĞİŞKENLER

`metin = "1 2 3 "` `i = 4`

ÇIKTI

Kod İzleme: `StringBuilder` ile Birleştirme



`toString()` tampondan tek bir String üretir.

Adım 9 / 10

```
1 StringBuilder metin = new StringBuilder();
2 for (int i = 1; i <= 3; i++) {
3     metin.append(i).append(' ');
4 }
▶ String sonuc = metin.toString();
6 System.out.println(sonuc + "|");
```

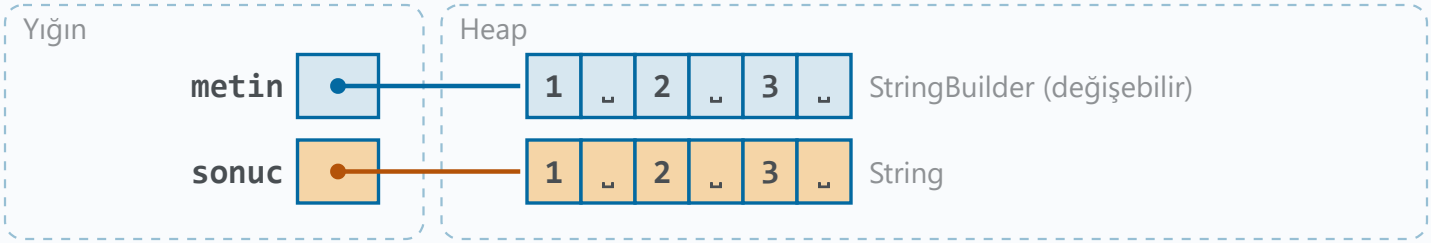
DEĞİŞKENLER

`metin = "1 2 3 "`

`sonuc = "1 2 3 "`

ÇIKTI

Kod İzleme: `StringBuilder` ile Birleştirme



Sondaki boşluk görünsün diye `|` eklendi.

Adım 10 / 10

```
1 StringBuilder metin = new StringBuilder();
2 for (int i = 1; i <= 3; i++) {
3     metin.append(i).append(' ');
4 }
5 String sonuc = metin.toString();
▶ System.out.println(sonuc + "|");
```

DEĞİŞKENLER

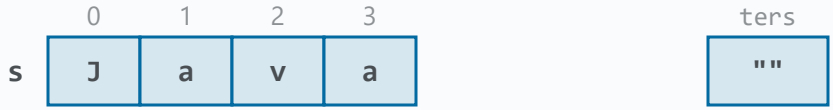
`metin = "1 2 3 "`

`sonuc = "1 2 3 "`

ÇIKTI

```
1 2 3 |
```

Dizgiyi Ters Çevirme



Dizgiyi sondan başa doğru okuyup yeni bir dizgiye ekleyeceğiz.

Adım 1 / 9

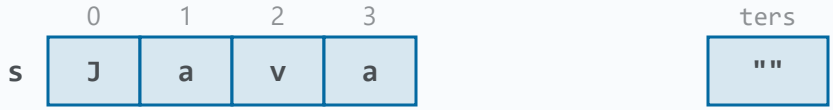
```
▶ String s = "Java";
2 String ters = "";
3 for (int i = s.length() - 1; i >= 0; i--)
4   ters += s.charAt(i);
5 System.out.println(ters);
6 String t2 = new StringBuilder(s).reverse()
7                                 .toString();
8 System.out.println(t2);
```

DEĞİŞKENLER

`s = "Java"`

ÇIKTI

Dizgiyi Ters Çevirme



Boş dizgiyle başlanır.

Adım 2 / 9

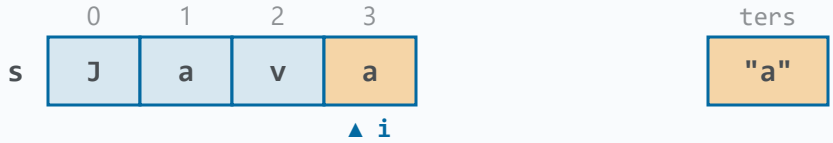
```
1 String s = "Java";
▶ String ters = "";
3 for (int i = s.length() - 1; i >= 0; i--)
4   ters += s.charAt(i);
5 System.out.println(ters);
6 String t2 = new StringBuilder(s).reverse()
7                                 .toString();
8 System.out.println(t2);
```

DEĞİŞKENLER

`s = "Java"` `ters = ""`

ÇIKTI

Dizgiyi Ters Çevirme



i = 3: `s.charAt(3)` = 'a' eklenir → "a" (her += yeni String üretir).

Adım 3 / 9

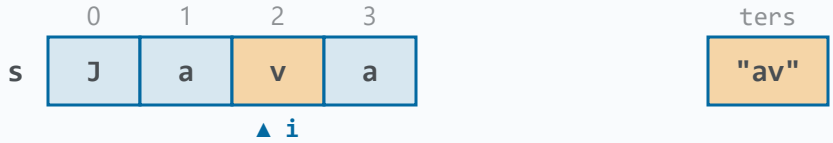
```
1 String s = "Java";
2 String ters = "";
▶ for (int i = s.length() - 1; i >= 0; i--)
▶   ters += s.charAt(i);
5 System.out.println(ters);
6 String t2 = new StringBuilder(s).reverse()
7                                 .toString();
8 System.out.println(t2);
```

DEĞİŞKENLER

`s = "Java"` `ters = "a"` `i = 3`

ÇIKTI

Dizgiyi Ters Çevirme



i = 2: `s.charAt(2)` = 'v' eklenir → "av" (her += yeni String üretir).

Adım 4 / 9

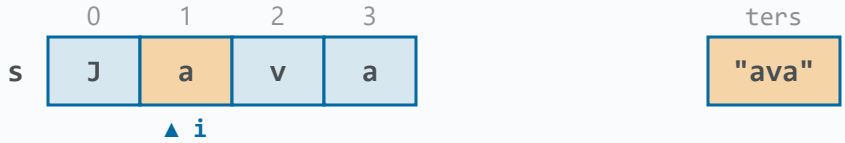
```
1 String s = "Java";
2 String ters = "";
▶ for (int i = s.length() - 1; i >= 0; i--)
▶   ters += s.charAt(i);
5 System.out.println(ters);
6 String t2 = new StringBuilder(s).reverse()
7                                 .toString();
8 System.out.println(t2);
```

DEĞİŞKENLER

`s = "Java"` `ters = "av"` `i = 2`

ÇIKTI

Dizgiyi Ters Çevirme



i = 1: `s.charAt(1)` = 'a' eklenir → "ava" (her += yeni String üretir).

Adım 5 / 9

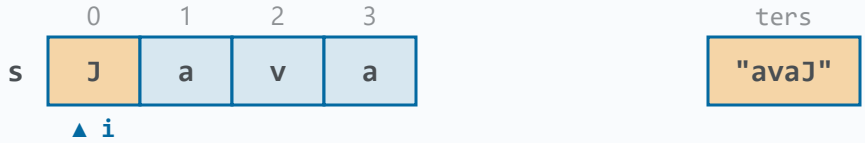
```
1 String s = "Java";
2 String ters = "";
▶ for (int i = s.length() - 1; i >= 0; i--)
▶   ters += s.charAt(i);
5 System.out.println(ters);
6 String t2 = new StringBuilder(s).reverse()
7                                 .toString();
8 System.out.println(t2);
```

DEĞİŞKENLER

`s = "Java"` `ters = "ava"` `i = 1`

ÇIKTI

Dizgiyi Ters Çevirme



i = 0: `s.charAt(0)` = 'J' eklenir → "avaJ" (her += yeni String üretir).

Adım 6 / 9

```
1 String s = "Java";
2 String ters = "";
▶ for (int i = s.length() - 1; i >= 0; i--)
▶   ters += s.charAt(i);
5 System.out.println(ters);
6 String t2 = new StringBuilder(s).reverse()
7                                 .toString();
8 System.out.println(t2);
```

DEĞİŞKENLER

`s = "Java"` `ters = "avaJ"` `i = 0`

ÇIKTI

Dizgiyi Ters Çevirme



i = −1 → döngü biter; sonuç yazdırılır.

Adım 7 / 9

```
1 String s = "Java";
2 String ters = "";
▶ for (int i = s.length() - 1; i >= 0; i--)
4   ters += s.charAt(i);
▶ System.out.println(ters);
6 String t2 = new StringBuilder(s).reverse()
7                                 .toString();
8 System.out.println(t2);
```

DEĞİŞKENLER

`s = "Java"` `ters = "avaJ"`

ÇIKTI

```
avaJ
```

Dizgiyi Ters Çevirme



Kısa yol: `StringBuilder`'ın hazır `reverse()` metodu aynı sonucu verir.

Adım 8 / 9

```
1 String s = "Java";
2 String ters = "";
3 for (int i = s.length() - 1; i >= 0; i--)
4   ters += s.charAt(i);
5 System.out.println(ters);
▶ String t2 = new StringBuilder(s).reverse()
▶                                 .toString();
8 System.out.println(t2);
```

DEĞİŞKENLER

`s = "Java"` `ters = "avaJ"` `t2 = "avaJ"`

ÇIKTI

```
avaJ
```

Dizgiyi Ters Çevirme



İki yöntem de aynı çıktıyı üretir; `s` hiç değişmedi.

Adım 9 / 9

```
1 String s = "Java";
2 String ters = "";
3 for (int i = s.length() - 1; i >= 0; i--)
4   ters += s.charAt(i);
5 System.out.println(ters);
6 String t2 = new StringBuilder(s).reverse()
7                                 .toString();
▶ System.out.println(t2);
```

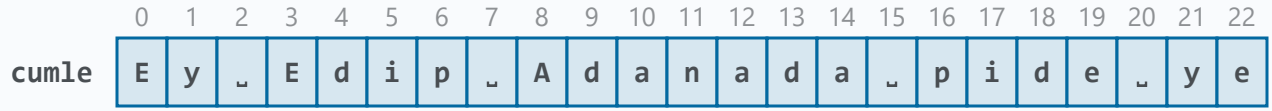
DEĞİŞKENLER

`s = "Java"` `ters = "avaJ"` `t2 = "avaJ"`

ÇIKTI

```
avaJ
avaJ
```

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



Cümle büyük harf ve boşluk içeriyor; doğrudan karşılaştırırsak 'E' ≠ 'e' olur.

Adım 1 / 13

```
▶ String cumle = "Ey Edip Adanada pide ye";
2 String t = cumle.toLowerCase().replace(" ", "");
3 int sol = 0, sag = t.length() - 1;
4 boolean palindrom = true;
5 while (sol < sag && palindrom) {
6   if (t.charAt(sol) != t.charAt(sag))
7     palindrom = false;
8   sol++; sag--;
9 }
10 System.out.println(palindrom);
```

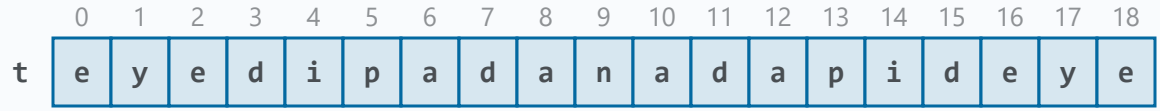
DEĞİŞKENLER

cumle	"Ey Edip Adanada pide ye"
=	

ÇIKTI

```
```

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



`toLowerCase()` ve `replace(" ", "")` → "eyedipadanadapideye" (19 karakter).

Adım 2 / 13

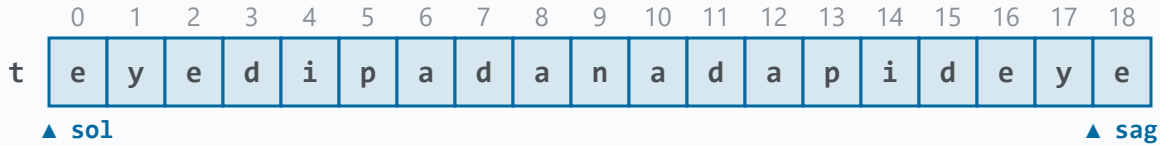
```
1  String cumle = "Ey Edip Adanada pide ye";
▶  String t = cumle.toLowerCase().replace(" ", "");
3  int sol = 0, sag = t.length() - 1;
4  boolean palindrom = true;
5  while (sol < sag && palindrom) {
6    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

`t = "eyedipadanadapideye"`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



İki uç işaretçi: sol = 0, sag = 18. Aksi kanıtlanana kadar palindrom = true.

Adım 3 / 13

```
1  String cumle = "Ey Edip Adanada pide ye";
2  String t = cumle.toLowerCase().replace(" ", "");
▶  int sol = 0, sag = t.length() - 1;
▶  boolean palindrom = true;
5  while (sol < sag && palindrom) {
6    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

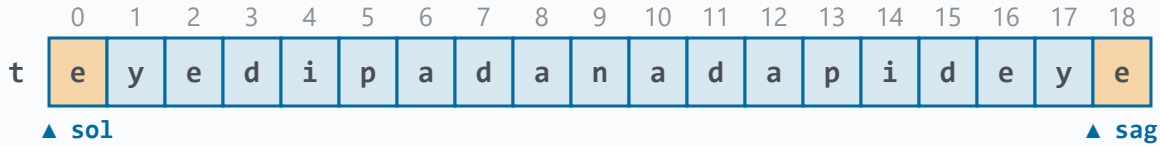
`t = "eyedipadanadapideye"`

`sol = 0` `sag = 18`

`palindrom = true`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



'e' ile 'e' → **eşit**, içe doğru ilerlenir.

Adım 4 / 13

```
1  String cumle = "Ey Edip Adanada pide ye";
2  String t = cumle.toLowerCase().replace(" ", "");
3  int sol = 0, sag = t.length() - 1;
4  boolean palindrom = true;
▶  while (sol < sag && palindrom) {
▶    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

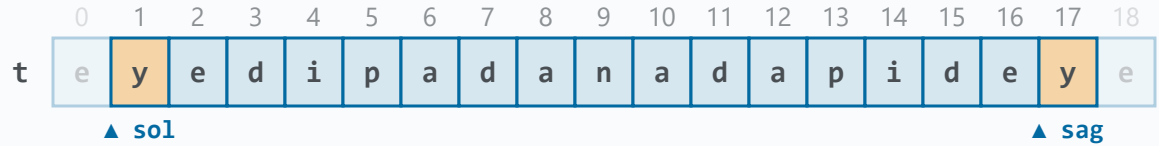
`t = "eyedipadanadapideye"`

`sol = 0` `sag = 18`

`palindrom = true`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



'y' ile 'y' → **eşit**, içe doğru ilerlenir.

Adım 5 / 13

```
1  String cumle = "Ey Edip Adanada pide ye";
2  String t = cumle.toLowerCase().replace(" ", "");
3  int sol = 0, sag = t.length() - 1;
4  boolean palindrom = true;
▶  while (sol < sag && palindrom) {
▶    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

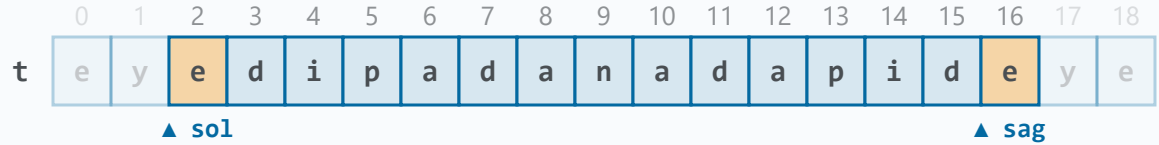
`t = "eyedipadanadapideye"`

`sol = 1` `sag = 17`

`palindrom = true`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



'e' ile 'e' → **eşit**, içe doğru ilerlenir.

Adım 6 / 13

```
1  String cumle = "Ey Edip Adanada pide ye";
2  String t = cumle.toLowerCase().replace(" ", "");
3  int sol = 0, sag = t.length() - 1;
4  boolean palindrom = true;
▶  while (sol < sag && palindrom) {
▶    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

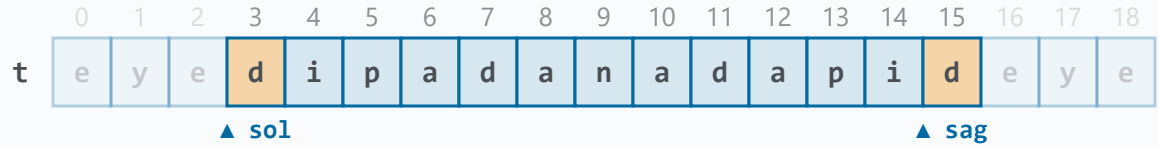
`t = "eyedipadanadapideye"`

`sol = 2` `sag = 16`

`palindrom = true`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



'd' ile 'd' → **eşit**, içe doğru ilerlenir.

Adım 7 / 13

```
1  String cumle = "Ey Edip Adanada pide ye";
2  String t = cumle.toLowerCase().replace(" ", "");
3  int sol = 0, sag = t.length() - 1;
4  boolean palindrom = true;
▶  while (sol < sag && palindrom) {
▶    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

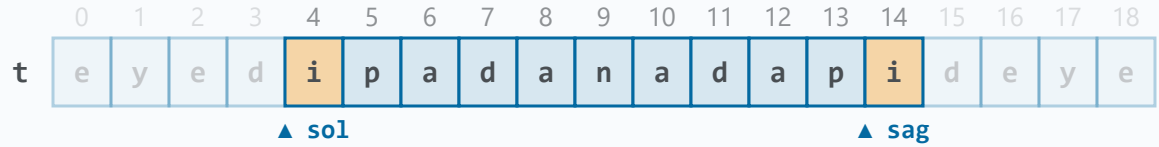
`t = "eyedipadanadapideye"`

`sol = 3` `sag = 15`

`palindrom = true`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



'i' ile 'i' → **eşit**, içe doğru ilerlenir.

Adım 8 / 13

```
1  String cumle = "Ey Edip Adanada pide ye";
2  String t = cumle.toLowerCase().replace(" ", "");
3  int sol = 0, sag = t.length() - 1;
4  boolean palindrom = true;
▶  while (sol < sag && palindrom) {
▶    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

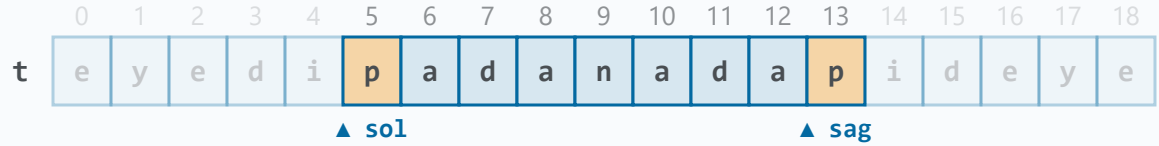
`t = "eyedipadanadapideye"`

`sol = 4` `sag = 14`

`palindrom = true`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



'p' ile 'p' → **eşit**, içe doğru ilerlenir.

Adım 9 / 13

```
1  String cumle = "Ey Edip Adanada pide ye";
2  String t = cumle.toLowerCase().replace(" ", "");
3  int sol = 0, sag = t.length() - 1;
4  boolean palindrom = true;
▶  while (sol < sag && palindrom) {
▶    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

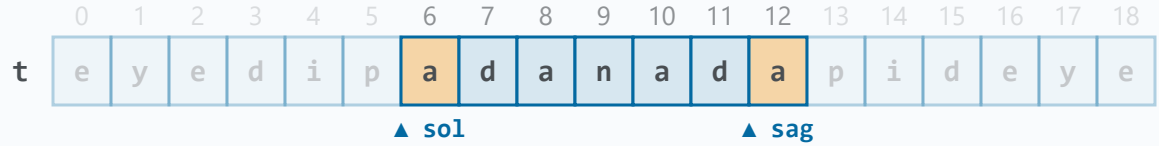
`t = "eyedipadanadapideye"`

`sol = 5` `sag = 13`

`palindrom = true`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



'a' ile 'a' → **eşit**, içe doğru ilerlenir.

Adım 10 / 13

```
1  String cumle = "Ey Edip Adanada pide ye";
2  String t = cumle.toLowerCase().replace(" ", "");
3  int sol = 0, sag = t.length() - 1;
4  boolean palindrom = true;
▶  while (sol < sag && palindrom) {
▶    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

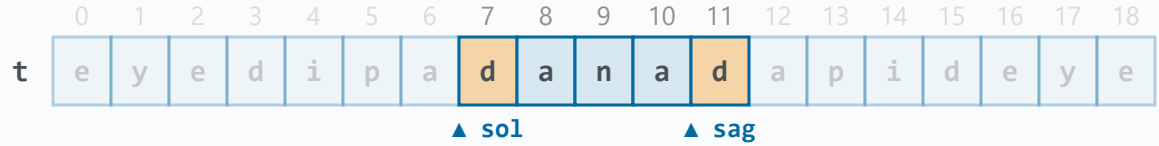
`t = "eyedipadanadapideye"`

`sol = 6` `sag = 12`

`palindrom = true`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



'd' ile 'd' → **eşit**, içe doğru ilerlenir.

Adım 11 / 13

```
1  String cumle = "Ey Edip Adanada pide ye";
2  String t = cumle.toLowerCase().replace(" ", "");
3  int sol = 0, sag = t.length() - 1;
4  boolean palindrom = true;
▶  while (sol < sag && palindrom) {
▶    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

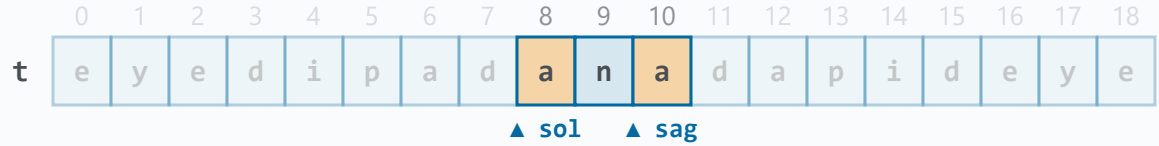
`t = "eyedipadanadapideye"`

`sol = 7` `sag = 11`

`palindrom = true`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



'a' ile 'a' → **eşit**, içe doğru ilerlenir.

Adım 12 / 13

```
1  String cumle = "Ey Edip Adanada pide ye";
2  String t = cumle.toLowerCase().replace(" ", "");
3  int sol = 0, sag = t.length() - 1;
4  boolean palindrom = true;
▶  while (sol < sag && palindrom) {
▶    if (t.charAt(sol) != t.charAt(sag))
7      palindrom = false;
8    sol++; sag--;
9  }
10 System.out.println(palindrom);
```

DEĞİŞKENLER

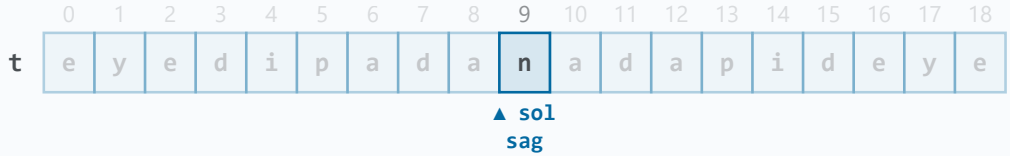
`t = "eyedipadanadapideye"`

`sol = 8` `sag = 10`

`palindrom = true`

ÇIKTI

Palindrom: Harf Büyüklüğü ve Boşluk Yok Sayılarak



sol = 9, sag = 9: işaretçiler buluştu, tüm çiftler eşit → **true**.

Adım 13 / 13

```
1 String cumle = "Ey Edip Adanada pide ye";
2 String t = cumle.toLowerCase().replace(" ", "");
3 int sol = 0, sag = t.length() - 1;
4 boolean palindrom = true;
5 while (sol < sag && palindrom) {
6   if (t.charAt(sol) != t.charAt(sag))
7     palindrom = false;
8   sol++; sag--;
9 }
▶ System.out.println(palindrom);
```

DEĞİŞKENLER

`t = "eyedipadanadapideye"`

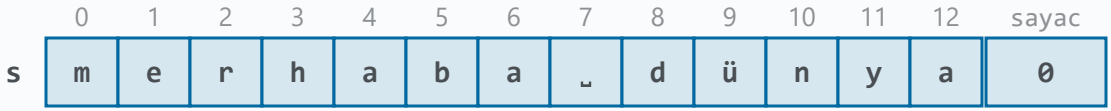
`sol = 9` `sag = 9`

`palindrom = true`

ÇIKTI

```
true
```

Sesli Harf Sayma



Önce hepsi küçük harfe çevrilir; `sesliler` dizgisi Türkçe sesli harfleri tutar.

Adım 1 / 15

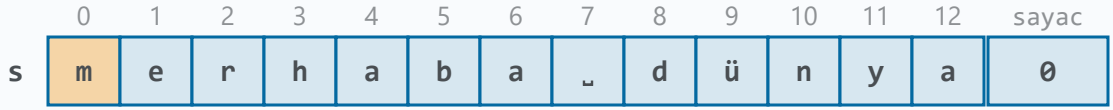
```
String s = "Merhaba Dünya".toLowerCase();
String sesliler = "aeıioöuü";
int sayac = 0;
for (int i = 0; i < s.length(); i++) {
  char c = s.charAt(i);
  if (sesliler.indexOf(c) >= 0) sayac++;
}
System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 0`

ÇIKTI

Sesli Harf Sayma



'm': `indexOf` → -1 → sesli değil.

Adım 2 / 15

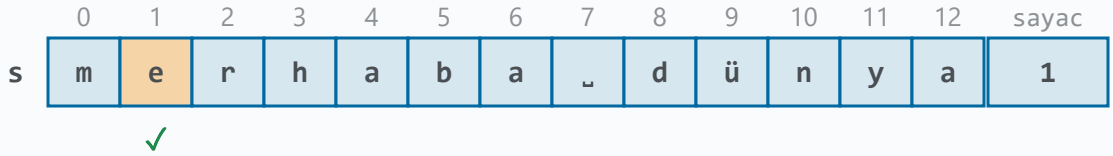
```
1  String s = "Merhaba Dünya".toLowerCase();
2  String sesliler = "aeıioöuü";
3  int sayac = 0;
4  for (int i = 0; i < s.length(); i++) {
▶    char c = s.charAt(i);
▶    if (sesliler.indexOf(c) >= 0) sayac++;
7  }
8  System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 0` `i = 0` `c = 'm'`

ÇIKTI

Sesli Harf Sayma



'e': `indexOf` → 1 ≥ 0 → sesli, sayac = 1.

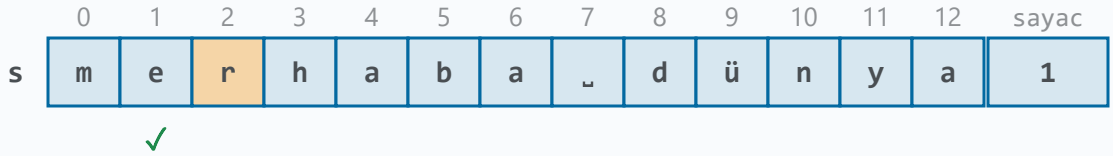

```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 1` `i = 1` `c = 'e'`

ÇIKTI

Sesli Harf Sayma



'r': `indexOf` → -1 → sesli değil.

Adım 4 / 15

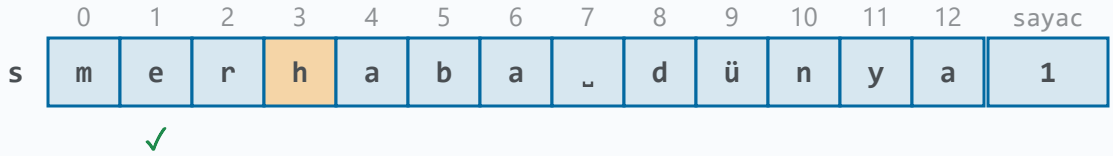
```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 1` `i = 2` `c = 'r'`

ÇIKTI

Sesli Harf Sayma



'h': `indexOf` → -1 → sesli değil.

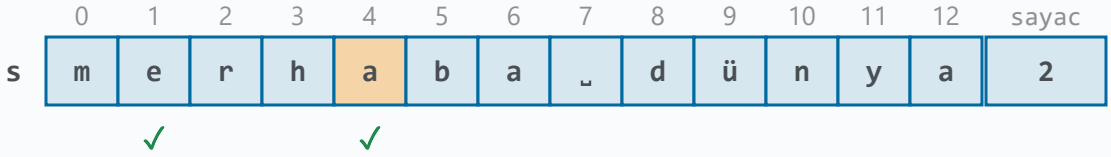

```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 1` `i = 3` `c = 'h'`

ÇIKTI

Sesli Harf Sayma



'a': `indexOf` → 0 ≥ 0 → sesli, sayac = 2.

Adım 6 / 15

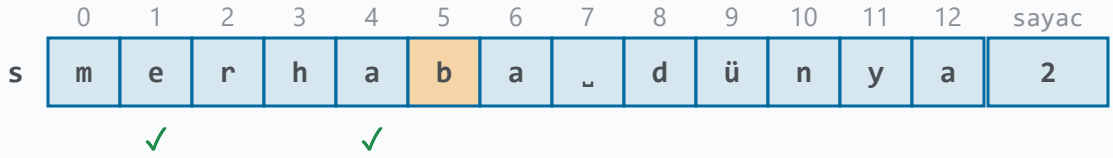
```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 2` `i = 4` `c = 'a'`

ÇIKTI

Sesli Harf Sayma



'b': `indexOf` → -1 → sesli değil.

Adım 7 / 15

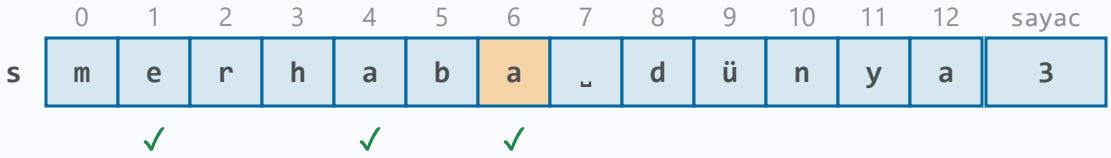
```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 2` `i = 5` `c = 'b'`

ÇIKTI

Sesli Harf Sayma



'a': `indexOf` → 0 ≥ 0 → sesli, sayac = 3.

Adım 8 / 15

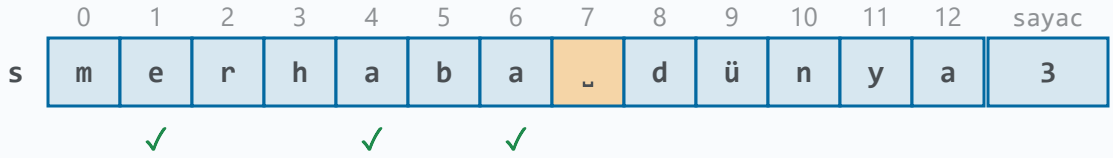
```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 3` `i = 6` `c = 'a'`

ÇIKTI

Sesli Harf Sayma



'␣': `indexOf` → -1 → sesli değil.

Adım 9 / 15

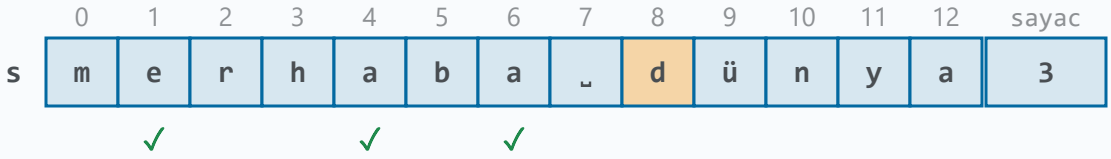
```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 3` `i = 7` `c = ' '`

ÇIKTI

Sesli Harf Sayma



'd': `indexOf` → -1 → sesli değil.

Adım 10 / 15

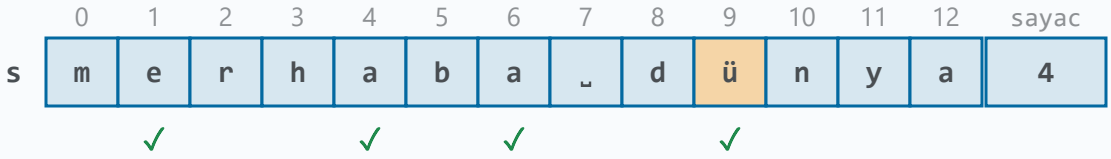
```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 3` `i = 8` `c = 'd'`

ÇIKTI

Sesli Harf Sayma



'ü': `indexOf` → 7 ≥ 0 → sesli, sayac = 4.

Adım 11 / 15

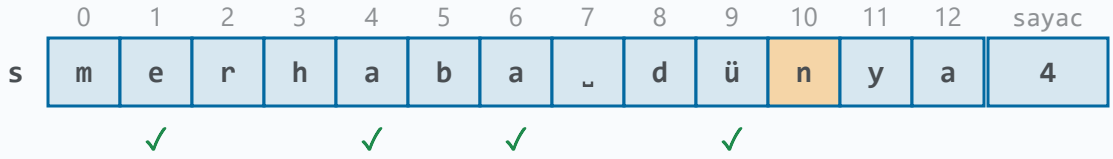
```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 4` `i = 9` `c = 'ü'`

ÇIKTI

Sesli Harf Sayma



'n': `indexOf` → -1 → sesli değil.

Adım 12 / 15

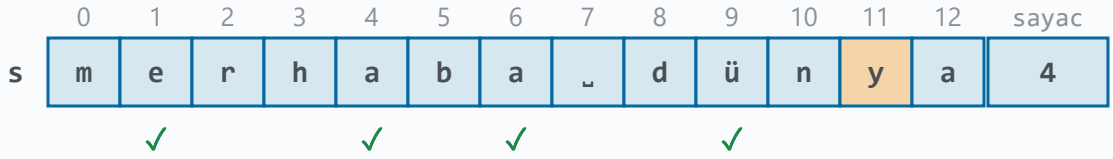
```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 4` `i = 10` `c = 'n'`

ÇIKTI

Sesli Harf Sayma



'y': `indexOf` → -1 → sesli değil.

Adım 13 / 15

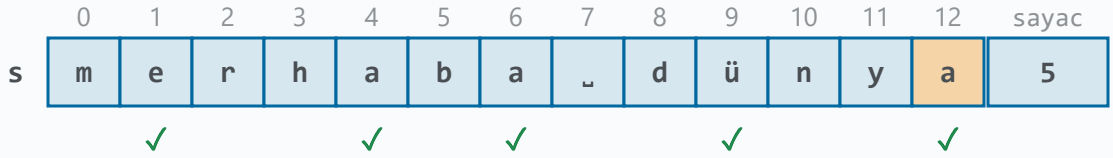
```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 4` `i = 11` `c = 'y'`

ÇIKTI

Sesli Harf Sayma



'a': `indexOf` → 0 ≥ 0 → sesli, sayac = 5.

Adım 14 / 15

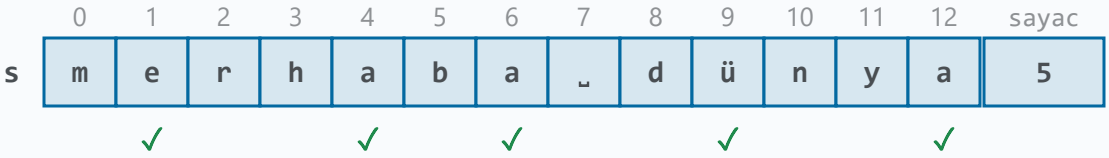
```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
▶   char c = s.charAt(i);
▶   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
8 System.out.println(sayac);
```

DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 5` `i = 12` `c = 'a'`

ÇIKTI

Sesli Harf Sayma



13 karakter tarandı: **5** sesli harf.

Adım 15 / 15

```
1 String s = "Merhaba Dünya".toLowerCase();
2 String sesliler = "aeıioöuü";
3 int sayac = 0;
4 for (int i = 0; i < s.length(); i++) {
5   char c = s.charAt(i);
6   if (sesliler.indexOf(c) >= 0) sayac++;
7 }
▶ System.out.println(sayac);
```

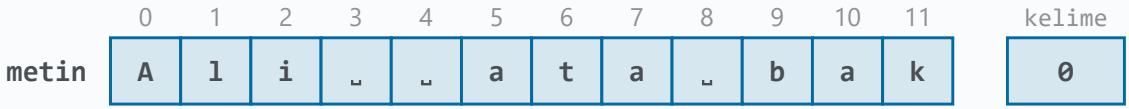
DEĞİŞKENLER

`s = "merhaba dünya"` `sayac = 5`

ÇIKTI

```
5
```

Kelime Sayma (boşluk geçişleri)



Kelime = boşluk olmayan karakterlerin kesintisiz dizisi. **Boşluktan harfe geçişleri** sayacağız.

Adım 1 / 14

```
String metin = "Ali  ata bak";
int kelime = 0;
boolean icinde = false;
for (int i = 0; i < metin.length(); i++) {
  char c = metin.charAt(i);
  if (c != ' ' && !icinde) {
    kelime++;
    icinde = true;
  } else if (c == ' ') icinde = false;
}
System.out.println(kelime);
```

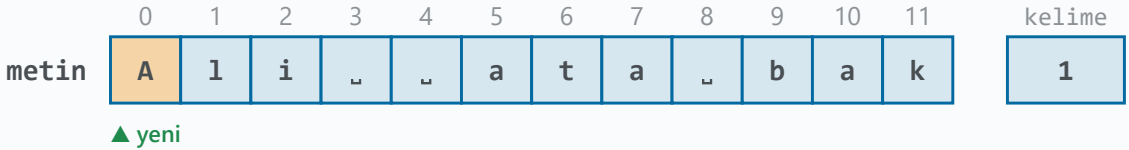
DEĞİŞKENLER

metin = "Ali ata bak"

kelime = 0 icinde = false

ÇIKTI

Kelime Sayma (boşluk geçişleri)



'A' harf ve kelime dışındaydık → **yeni kelime**: kelime = 1.

Adım 2 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
▶    if (c != ' ' && !icinde) {
▶      kelime++;
▶      icinde = true;
9    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

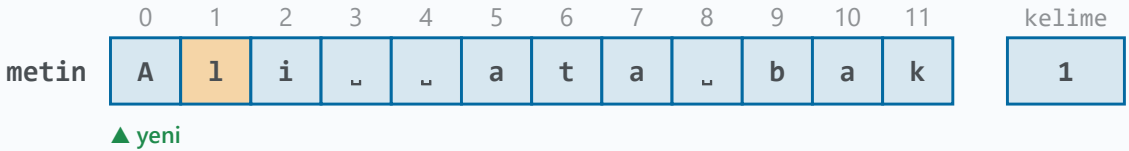
`metin = "Ali ata bak"`

`kelime = 1` `icinde = true`

`i = 0`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



'l' aynı kelimenin devamı.

Adım 3 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
▶    if (c != ' ' && !icinde) {
7      kelime++;
8      icinde = true;
9    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

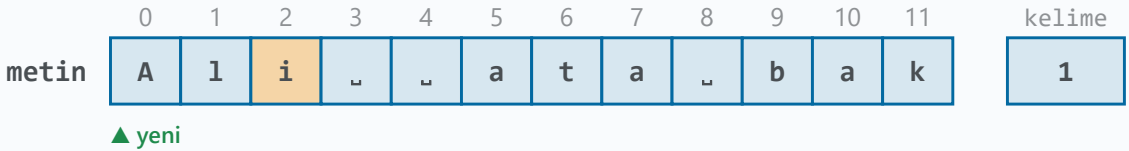
`metin = "Ali ata bak"`

`kelime = 1` `icinde = true`

`i = 1`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



'i' aynı kelimenin devamı.

Adım 4 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
▶    if (c != ' ' && !icinde) {
7      kelime++;
8      icinde = true;
9    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

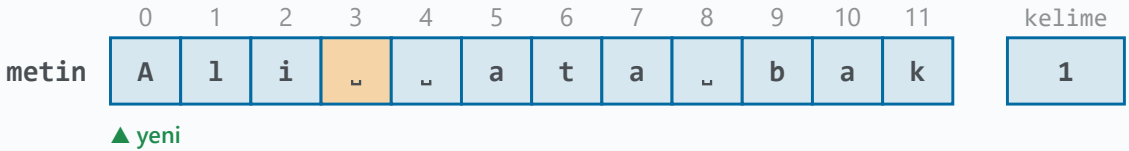
`metin = "Ali ata bak"`

`kelime = 1` `icinde = true`

`i = 2`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



Boşluk → kelime dışına çıkıldı.

Adım 5 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
6    if (c != ' ' && !icinde) {
7      kelime++;
8      icinde = true;
▶    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

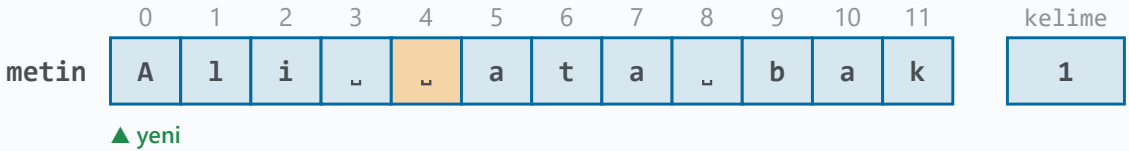
`metin = "Ali ata bak"`

`kelime = 1` `icinde = false`

`i = 3`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



Yine boşluk (çift boşluk) → sayılmaz.

Adım 6 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
6    if (c != ' ' && !icinde) {
7      kelime++;
8      icinde = true;
▶    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

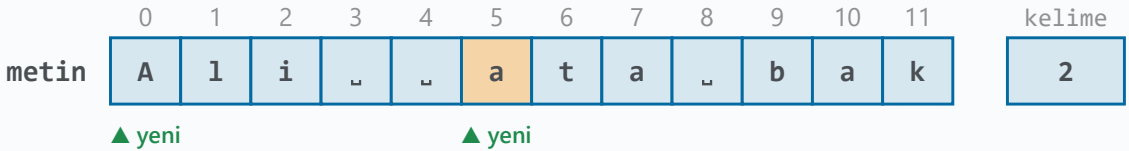
`metin = "Ali ata bak"`

`kelime = 1` `icinde = false`

`i = 4`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



'a' harf ve kelime dışındaydık → **yeni kelime**: kelime = 2.

Adım 7 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
▶    if (c != ' ' && !icinde) {
▶      kelime++;
▶      icinde = true;
9    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

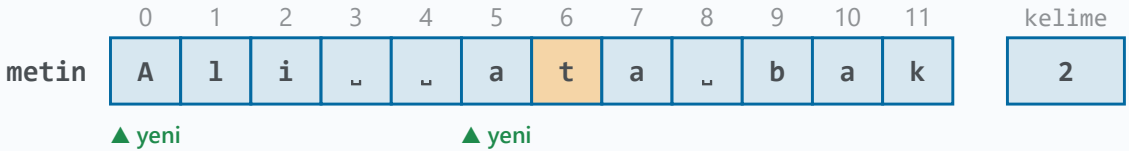
`metin = "Ali ata bak"`

`kelime = 2` `icinde = true`

`i = 5`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



't' aynı kelimenin devamı.

Adım 8 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
▶    if (c != ' ' && !icinde) {
7      kelime++;
8      icinde = true;
9    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

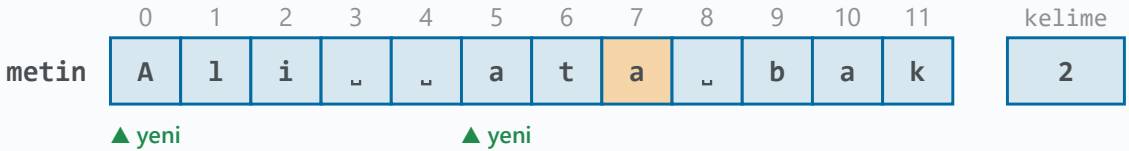
`metin = "Ali ata bak"`

`kelime = 2` `icinde = true`

`i = 6`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



'a' aynı kelimenin devamı.

Adım 9 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
▶    if (c != ' ' && !icinde) {
7      kelime++;
8      icinde = true;
9    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

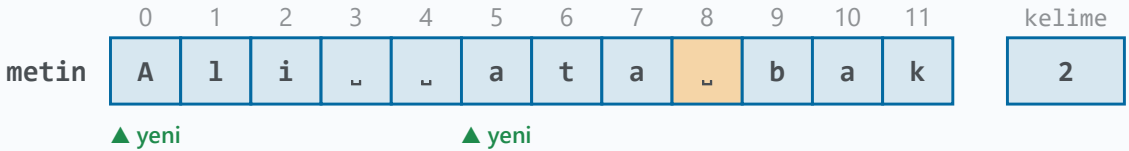
`metin = "Ali ata bak"`

`kelime = 2` `icinde = true`

`i = 7`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



Boşluk → kelime dışına çıkıldı.

Adım 10 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
6    if (c != ' ' && !icinde) {
7      kelime++;
8      icinde = true;
▶    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

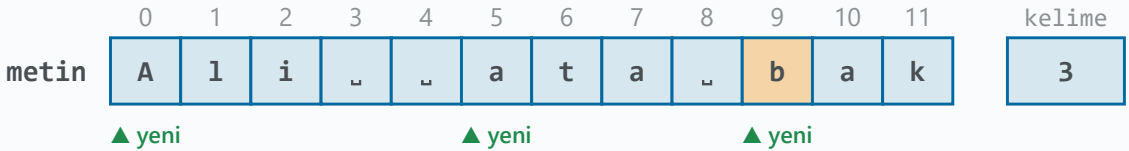
`metin = "Ali ata bak"`

`kelime = 2` `icinde = false`

`i = 8`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



'b' harf ve kelime dışındaydık → **yeni kelime**: kelime = 3.


```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
▶    if (c != ' ' && !icinde) {
▶      kelime++;
▶      icinde = true;
9    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

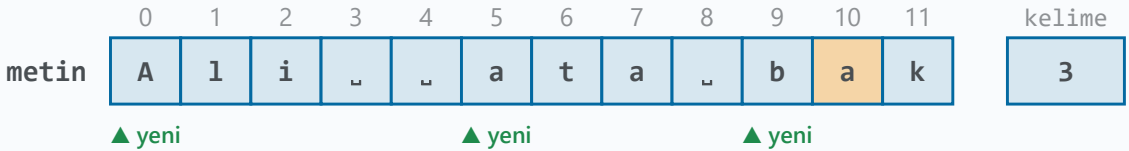
`metin = "Ali ata bak"`

`kelime = 3` `icinde = true`

`i = 9`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



'a' aynı kelimenin devamı.

Adım 12 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
▶    if (c != ' ' && !icinde) {
7      kelime++;
8      icinde = true;
9    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

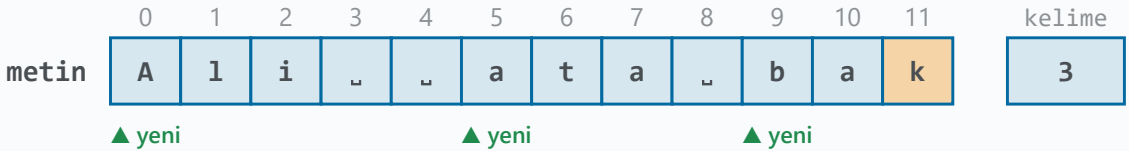
`metin = "Ali ata bak"`

`kelime = 3` `icinde = true`

`i = 10`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



'k' aynı kelimenin devamı.

Adım 13 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
▶    if (c != ' ' && !icinde) {
7      kelime++;
8      icinde = true;
9    } else if (c == ' ') icinde = false;
10 }
11 System.out.println(kelime);
```

DEĞİŞKENLER

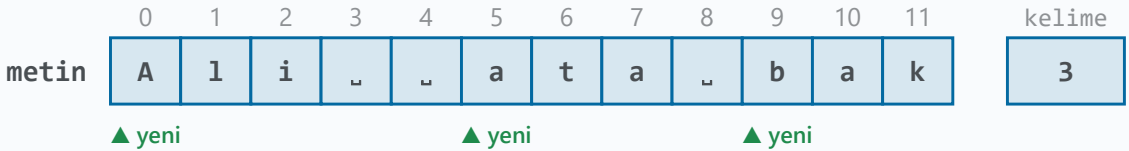
`metin = "Ali ata bak"`

`kelime = 3` `icinde = true`

`i = 11`

ÇIKTI

Kelime Sayma (boşluk geçişleri)



Sonuç **3**. Kısa yol: `metin.trim().split("\\s+").length` (yalnızca boş olmayan metin için; boş metinde sonuç 0 değil 1 çıkar, önce `isBlank()` ile denetlenir).

Adım 14 / 14

```
1  String metin = "Ali  ata bak";
2  int kelime = 0;
3  boolean icinde = false;
4  for (int i = 0; i < metin.length(); i++) {
5    char c = metin.charAt(i);
6    if (c != ' ' && !icinde) {
7      kelime++;
8      icinde = true;
9    } else if (c == ' ') icinde = false;
10 }
▶  System.out.println(kelime);
```

DEĞİŞKENLER

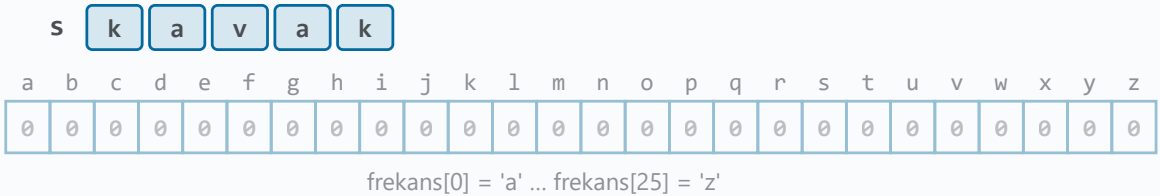
`metin = "Ali ata bak"`

`kelime = 3` `icinde = true`

ÇIKTI

```
3
```

Karakter Frekansı (`int[26]`)



26 hücreli sayaç: **'a' → 0, 'b' → 1, …** Harfin indisi `c - 'a'` ile bulunur (char'lar sayıdır).

Adım 1 / 9

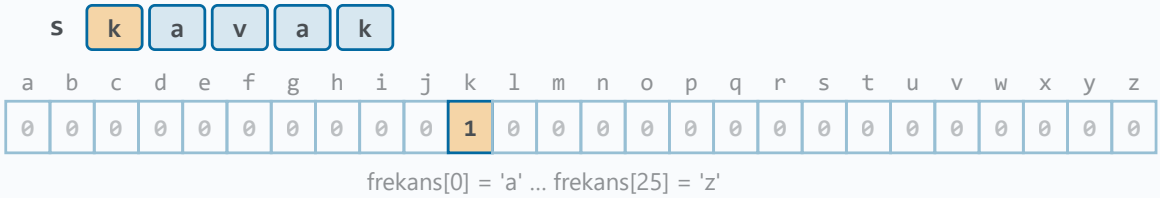
```
▶ String s = "kavak";
▶ int[] frekans = new int[26];
3 for (int i = 0; i < s.length(); i++) {
4   char c = s.charAt(i);
5   frekans[c - 'a']++;
6 }
7 for (int k = 0; k < 26; k++)
8   if (frekans[k] > 0)
9     System.out.println((char) ('a' + k)
10                       + ": " + frekans[k]);
```

DEĞİŞKENLER

`s = "kavak"`

ÇIKTI

Karakter Frekansı (`int[26]`)



'k' - 'a' = 107 − 97 = **10** → frekans[10] = 1.

Adım 2 / 9

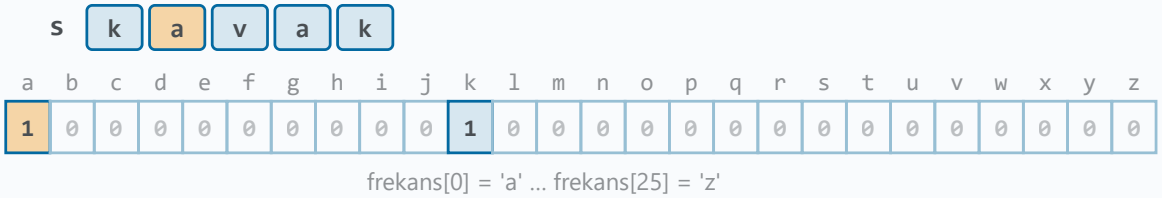
```
1  String s = "kavak";
2  int[] frekans = new int[26];
3  for (int i = 0; i < s.length(); i++) {
▶    char c = s.charAt(i);
▶    frekans[c - 'a']++;
6  }
7  for (int k = 0; k < 26; k++)
8    if (frekans[k] > 0)
9      System.out.println((char) ('a' + k)
10                        + ": " + frekans[k]);
```

DEĞİŞKENLER

`s = "kavak"` `i = 0` `c = 'k'`

ÇIKTI

Karakter Frekansı (`int[26]`)



'a' - 'a' = 97 − 97 = **0** → frekans[0] = 1.

Adım 3 / 9

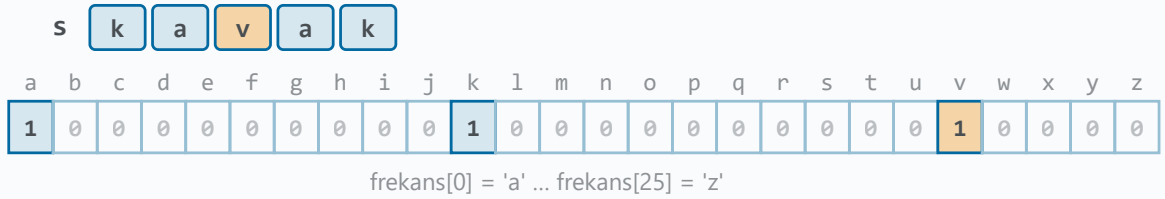
```
1  String s = "kavak";
2  int[] frekans = new int[26];
3  for (int i = 0; i < s.length(); i++) {
▶    char c = s.charAt(i);
▶    frekans[c - 'a']++;
6  }
7  for (int k = 0; k < 26; k++)
8    if (frekans[k] > 0)
9      System.out.println((char) ('a' + k)
10                        + ": " + frekans[k]);
```

DEĞİŞKENLER

`s = "kavak"` `i = 1` `c = 'a'`

ÇIKTI

Karakter Frekansı (`int[26]`)



'v' - 'a' = 118 − 97 = **21** → frekans[21] = 1.

Adım 4 / 9

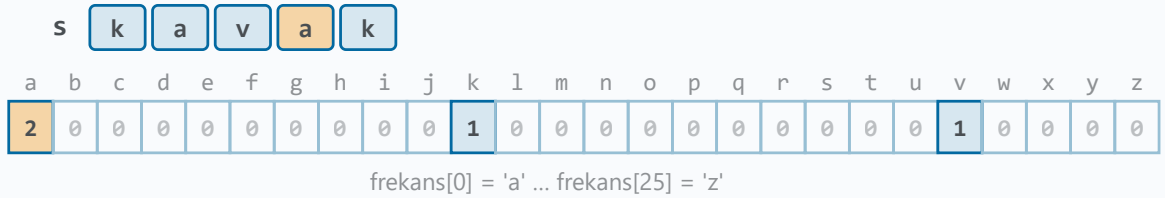
```
1  String s = "kavak";
2  int[] frekans = new int[26];
3  for (int i = 0; i < s.length(); i++) {
▶    char c = s.charAt(i);
▶    frekans[c - 'a']++;
6  }
7  for (int k = 0; k < 26; k++)
8    if (frekans[k] > 0)
9      System.out.println((char) ('a' + k)
10                        + ": " + frekans[k]);
```

DEĞİŞKENLER

`s = "kavak"` `i = 2` `c = 'v'`

ÇIKTI

Karakter Frekansı (`int[26]`)



'a' - 'a' = 97 − 97 = **0** → frekans[0] = 2.

Adım 5 / 9

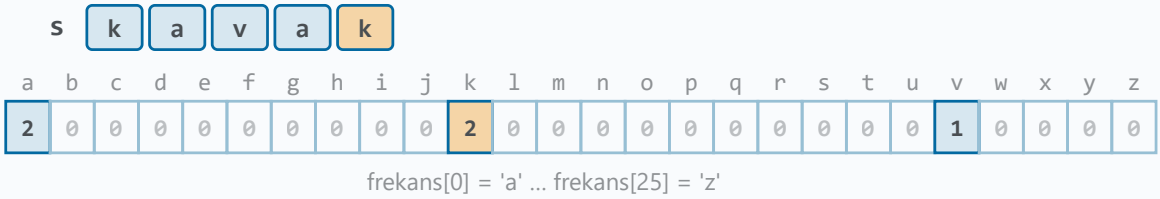
```
1  String s = "kavak";
2  int[] frekans = new int[26];
3  for (int i = 0; i < s.length(); i++) {
▶    char c = s.charAt(i);
▶    frekans[c - 'a']++;
6  }
7  for (int k = 0; k < 26; k++)
8    if (frekans[k] > 0)
9      System.out.println((char) ('a' + k)
10                        + ": " + frekans[k]);
```

DEĞİŞKENLER

`s = "kavak"` `i = 3` `c = 'a'`

ÇIKTI

Karakter Frekansı (`int[26]`)



'k' - 'a' = 107 − 97 = **10** → frekans[10] = 2.

Adım 6 / 9

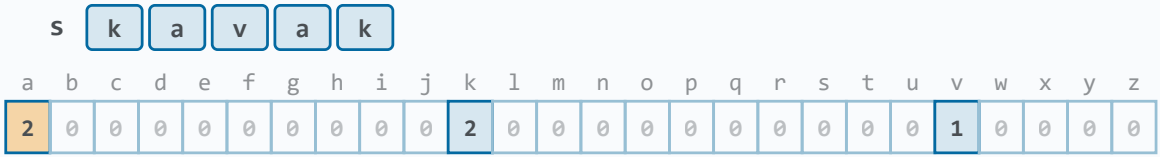
```
1  String s = "kavak";
2  int[] frekans = new int[26];
3  for (int i = 0; i < s.length(); i++) {
▶    char c = s.charAt(i);
▶    frekans[c - 'a']++;
6  }
7  for (int k = 0; k < 26; k++)
8    if (frekans[k] > 0)
9      System.out.println((char) ('a' + k)
10                        + ": " + frekans[k]);
```

DEĞİŞKENLER

`s = "kavak"` `i = 4` `c = 'k'`

ÇIKTI

Karakter Frekansı (`int[26]`)



k = 0: frekans > 0 → `(char) ('a' + 0)` = 'a' yazdırılır.

Adım 7 / 9

```
1 String s = "kavak";
2 int[] frekans = new int[26];
3 for (int i = 0; i < s.length(); i++) {
4   char c = s.charAt(i);
5   frekans[c - 'a']++;
6 }
▶ for (int k = 0; k < 26; k++)
▶   if (frekans[k] > 0)
▶     System.out.println((char) ('a' + k)
▶                        + ": " + frekans[k]);
```

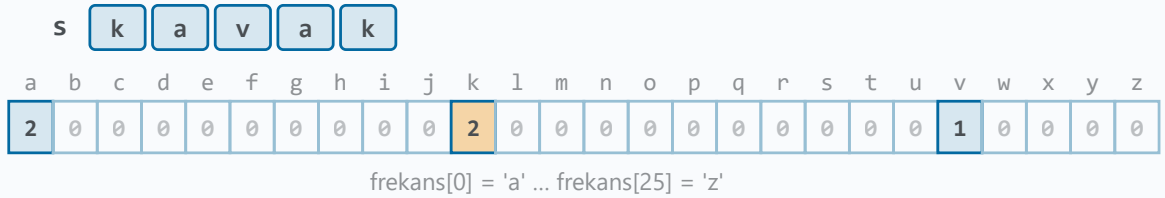
DEĞİŞKENLER

`s = "kavak"` `k = 0`

ÇIKTI

```
a: 2
```

Karakter Frekansı (`int[26]`)



k = 10: frekans > 0 → `(char) ('a' + 10)` = 'k' yazdırılır.

Adım 8 / 9

```
1 String s = "kavak";
2 int[] frekans = new int[26];
3 for (int i = 0; i < s.length(); i++) {
4   char c = s.charAt(i);
5   frekans[c - 'a']++;
6 }
▶ for (int k = 0; k < 26; k++)
▶   if (frekans[k] > 0)
▶     System.out.println((char) ('a' + k)
▶                        + ": " + frekans[k]);
```

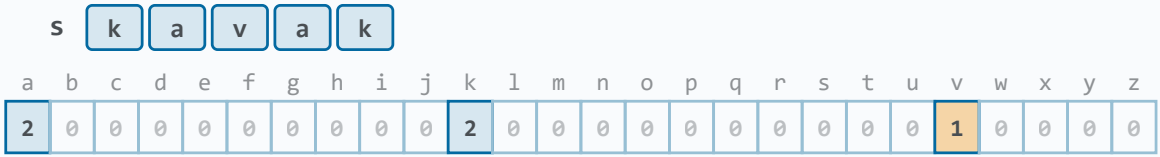
DEĞİŞKENLER

`s = "kavak"` `k = 10`

ÇIKTI

```
a: 2
k: 2
```

Karakter Frekansı (`int[26]`)



k = 21: frekans > 0 → `(char) ('a' + 21)` = 'v' yazdırılır. Tablo yalnızca a–z içindir; ç, ğ, ı, ö, ş, ü için ayrı bir düzen gerekir.

Adım 9 / 9

```
1 String s = "kavak";
2 int[] frekans = new int[26];
3 for (int i = 0; i < s.length(); i++) {
4   char c = s.charAt(i);
5   frekans[c - 'a']++;
6 }
▶ for (int k = 0; k < 26; k++)
▶   if (frekans[k] > 0)
▶     System.out.println((char) ('a' + k)
▶                        + ": " + frekans[k]);
```

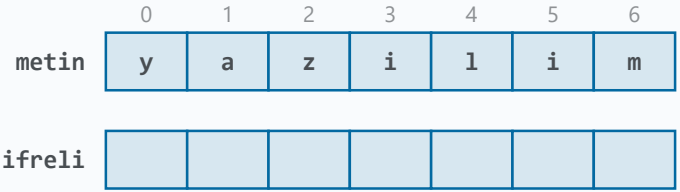
DEĞİŞKENLER

`s = "kavak"` `k = 21`

ÇIKTI

```
a: 2
k: 2
v: 1
```

Sezar Şifreleme ve Çözme



Her harf alfabede **3** ileri kaydırılır. Sonuç StringBuilder'da biriktirilir.

Adım 1 / 9

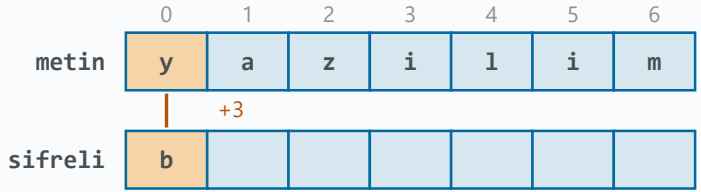
```
String metin = "yazilim";
int kaydir = 3;
StringBuilder sifreli = new StringBuilder();
for (int i = 0; i < metin.length(); i++) {
  char c = metin.charAt(i);
  int yer = (c - 'a' + kaydir) % 26;
  char yeni = (char) ('a' + yer);
  sifreli.append(yeni);
}
System.out.println(sifreli);
```

DEĞİŞKENLER

`metin = "yazilim"` `kaydir = 3` `sifreli = ""`

ÇIKTI

Sezar Şifreleme ve Çözme



a b c … x y z
y + 3 → b (başa sarar)

'y': (24 + 3) % 26 = **1** → 'b' (26'yı aştı, % 26 başa sardı).

Adım 2 / 9

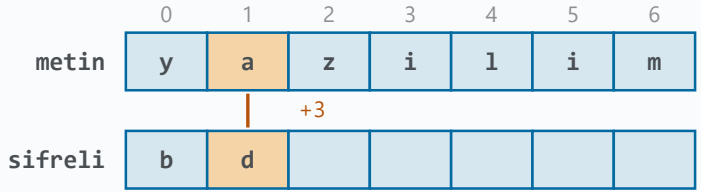
```
1 String metin = "yazilim";
2 int kaydir = 3;
3 StringBuilder sifreli = new StringBuilder();
4 for (int i = 0; i < metin.length(); i++) {
    char c = metin.charAt(i);
    int yer = (c - 'a' + kaydir) % 26;
    char yeni = (char) ('a' + yer);
    sifreli.append(yeni);
9 }
10 System.out.println(sifreli);
```

DEĞİŞKENLER

`metin = "yazilim"` `kaydir = 3` `sifreli = "b"` `c = 'y'` `yer = 1`

ÇIKTI

Sezar Şifreleme ve Çözme



'a': (0 + 3) % 26 = **3** → 'd'.

Adım 3 / 9

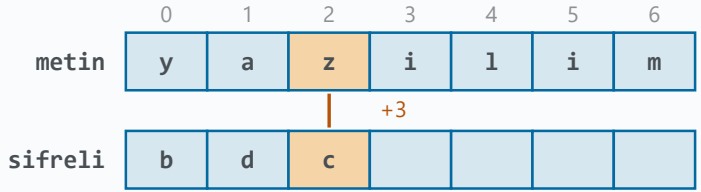
```
1 String metin = "yazilim";
2 int kaydir = 3;
3 StringBuilder sifreli = new StringBuilder();
4 for (int i = 0; i < metin.length(); i++) {
▶   char c = metin.charAt(i);
▶   int yer = (c - 'a' + kaydir) % 26;
▶   char yeni = (char) ('a' + yer);
▶   sifreli.append(yeni);
9 }
10 System.out.println(sifreli);
```

DEĞİŞKENLER

`metin = "yazilim"` `kaydir = 3` `sifreli = "bd"` `c = 'a'` `yer = 3`

ÇIKTI

Sezar Şifreleme ve Çözme



a b c … x y z
y + 3 → b (başa sarar)

'z': (25 + 3) % 26 = **2** → 'c' (26'yı aştı, % 26 başa sardı).

Adım 4 / 9

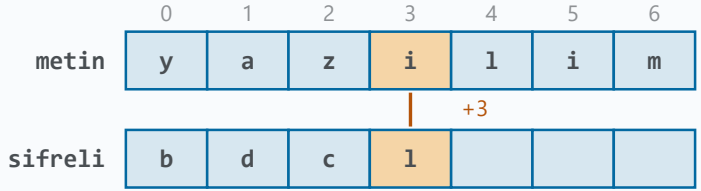
```
1 String metin = "yazilim";
2 int kaydir = 3;
3 StringBuilder sifreli = new StringBuilder();
4 for (int i = 0; i < metin.length(); i++) {
    char c = metin.charAt(i);
    int yer = (c - 'a' + kaydir) % 26;
    char yeni = (char) ('a' + yer);
    sifreli.append(yeni);
9 }
10 System.out.println(sifreli);
```

DEĞİŞKENLER

`metin = "yazilim"` `kaydir = 3` `sifreli = "bdc"` `c = 'z'` `yer = 2`

ÇIKTI

Sezar Şifreleme ve Çözme



a b c … x y z
y + 3 → b (başa sarar)

'i': (8 + 3) % 26 = **11** → 'l'.

Adım 5 / 9

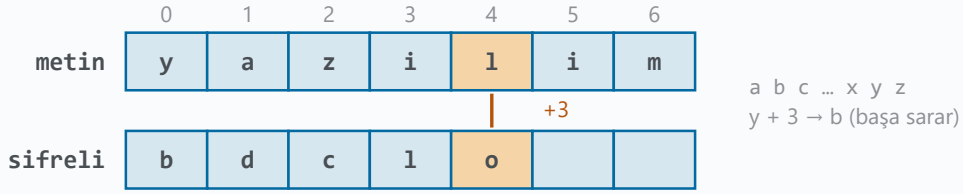
```
1 String metin = "yazilim";
2 int kaydir = 3;
3 StringBuilder sifreli = new StringBuilder();
4 for (int i = 0; i < metin.length(); i++) {
▶   char c = metin.charAt(i);
▶   int yer = (c - 'a' + kaydir) % 26;
▶   char yeni = (char) ('a' + yer);
▶   sifreli.append(yeni);
9 }
10 System.out.println(sifreli);
```

DEĞİŞKENLER

`metin = "yazilim"` `kaydir = 3` `sifreli = "bdcl"` `c = 'i'` `yer = 11`

ÇIKTI

Sezar Şifreleme ve Çözme



a b c … x y z
y + 3 → b (başa sarar)

'l': (11 + 3) % 26 = **14** → 'o'.

Adım 6 / 9

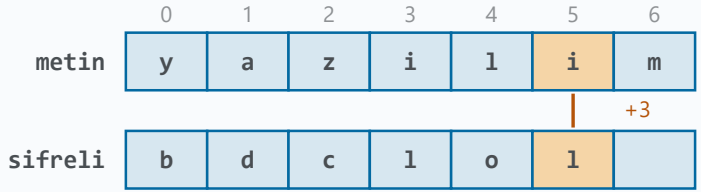
```
1 String metin = "yazilim";
2 int kaydir = 3;
3 StringBuilder sifreli = new StringBuilder();
4 for (int i = 0; i < metin.length(); i++) {
▶   char c = metin.charAt(i);
▶   int yer = (c - 'a' + kaydir) % 26;
▶   char yeni = (char) ('a' + yer);
▶   sifreli.append(yeni);
9 }
10 System.out.println(sifreli);
```

DEĞİŞKENLER

`metin = "yazilim"` `kaydir = 3` `sifreli = "bdclo"` `c = 'l'` `yer = 14`

ÇIKTI

Sezar Şifreleme ve Çözme



a b c … x y z
y + 3 → b (başa sarar)

'i': (8 + 3) % 26 = **11** → 'l'.

Adım 7 / 9

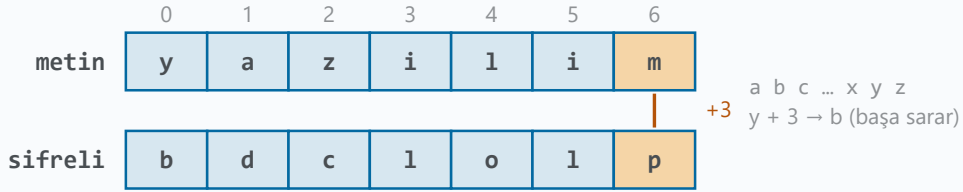
```
1 String metin = "yazilim";
2 int kaydir = 3;
3 StringBuilder sifreli = new StringBuilder();
4 for (int i = 0; i < metin.length(); i++) {
    char c = metin.charAt(i);
    int yer = (c - 'a' + kaydir) % 26;
    char yeni = (char) ('a' + yer);
    sifreli.append(yeni);
9 }
10 System.out.println(sifreli);
```

DEĞİŞKENLER

`metin = "yazilim"` `kaydir = 3` `sifreli = "bdclol"` `c = 'i'` `yer = 11`

ÇIKTI

Sezar Şifreleme ve Çözme



'm': (12 + 3) % 26 = **15** → 'p'.

Adım 8 / 9

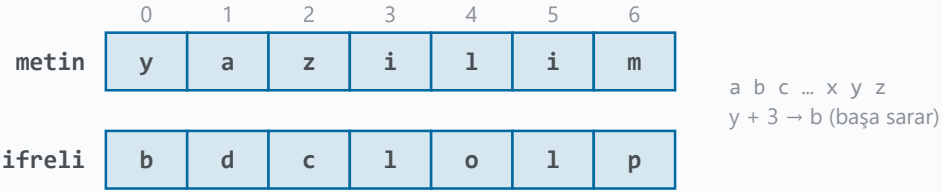
```
1 String metin = "yazilim";
2 int kaydir = 3;
3 StringBuilder sifreli = new StringBuilder();
4 for (int i = 0; i < metin.length(); i++) {
▶   char c = metin.charAt(i);
▶   int yer = (c - 'a' + kaydir) % 26;
▶   char yeni = (char) ('a' + yer);
▶   sifreli.append(yeni);
9 }
10 System.out.println(sifreli);
```

DEĞİŞKENLER

`metin = "yazilim"` `kaydir = 3` `sifreli = "bdclolp"` `c = 'm'` `yer = 15`

ÇIKTI

Sezar Şifreleme ve Çözme



Şifreli metin **bdclolp**. **Çözme:** aynı döngüde `kaydir` yerine `26 - kaydir` kullanılır → "yazilim".

Adım 9 / 9

```
1 String metin = "yazilim";
2 int kaydir = 3;
3 StringBuilder sifreli = new StringBuilder();
4 for (int i = 0; i < metin.length(); i++) {
5   char c = metin.charAt(i);
6   int yer = (c - 'a' + kaydir) % 26;
7   char yeni = (char) ('a' + yer);
8   sifreli.append(yeni);
9 }
▶ System.out.println(sifreli);
```

DEĞİŞKENLER

`metin = "yazilim"` `kaydir = 3` `sifreli = "bdclolp"`

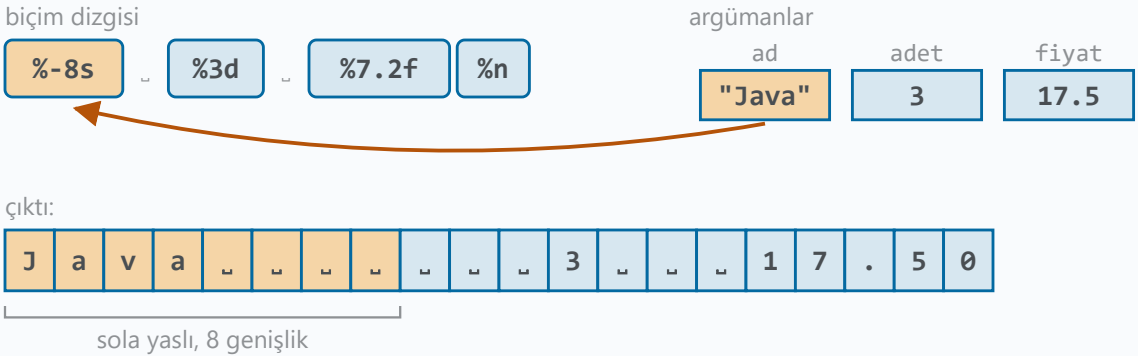
ÇIKTI

```
bdclolp
```

BİÇİMLEME

printf ile Biçimleme

```
String ad = "Java"; int adet = 3; double fiyat = 17.5;
System.out.printf("%-8s %3d %7.2f%n", ad, adet, fiyat);
```



`%-8s` ← `ad` : "Java" 8 hücreye **sola** yaslanır, kalan 4 hücre boşluk.

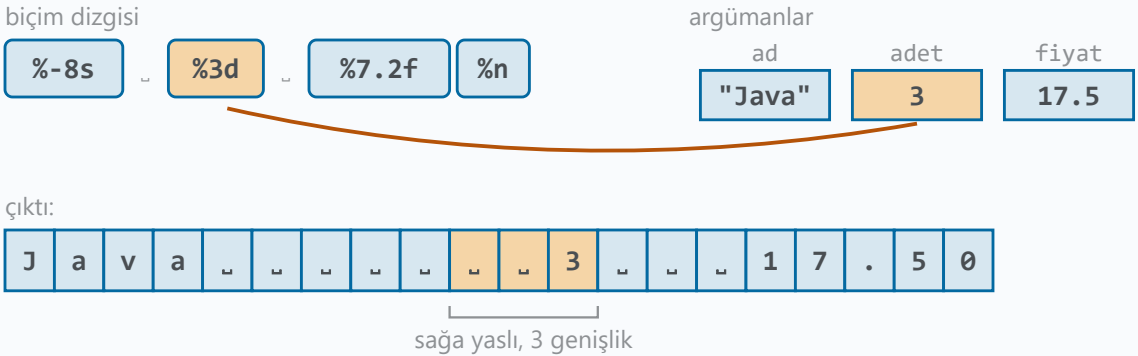
Adım 1 / 4

%s metin, %d tam sayı, %f kayan noktalı sayı. Türkçe yerel ayarlı bilgisayarda `%.2f` ondalık ayırıcı olarak virgül basabilir (17,50).

BİÇİMLEME

printf ile Biçimleme

```
String ad = "Java"; int adet = 3; double fiyat = 17.5;
System.out.printf("%-8s %3d %7.2f%n", ad, adet, fiyat);
```



`%3d` ← `adet` : 3 tam sayısı 3 hücreye **sağa** yaslanır.

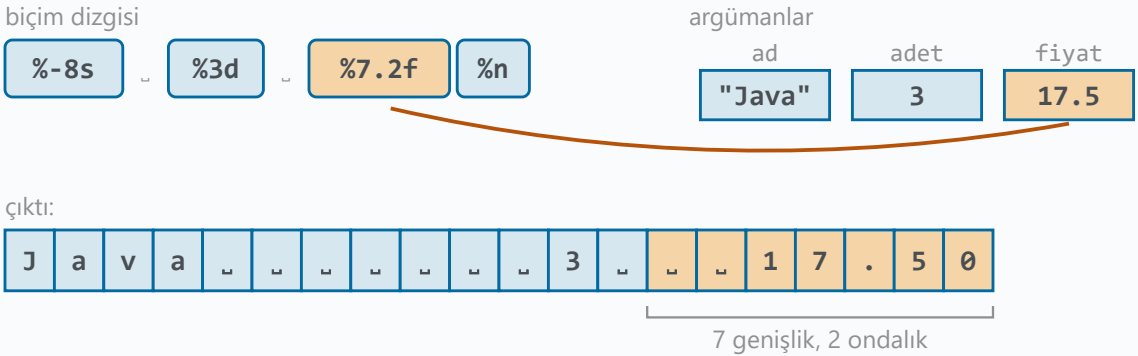
Adım 2 / 4

%s metin, %d tam sayı, %f kayan noktalı sayı. Türkçe yerel ayarlı bilgisayarda `%.2f` ondalık ayırıcı olarak virgül basabilir (17,50).

BİÇİMLEME

printf ile Biçimleme

```
String ad = "Java"; int adet = 3; double fiyat = 17.5;
System.out.printf("%-8s %3d %7.2f%n", ad, adet, fiyat);
```



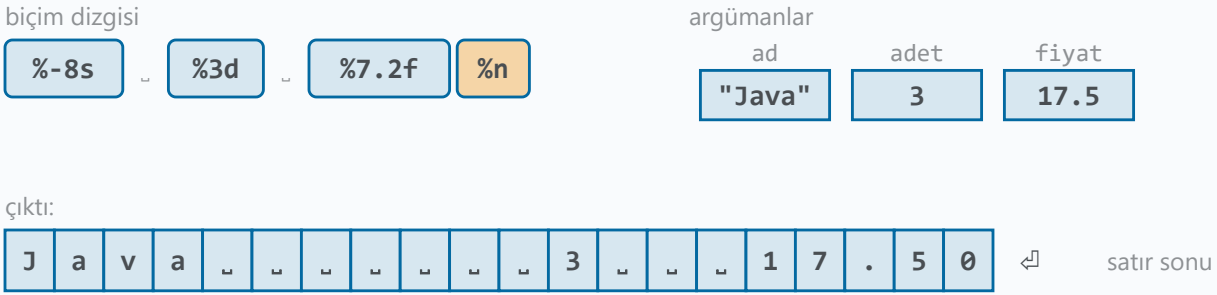
`%7.2f` ← `fiyat` : 2 ondalıkla "17.50" (5 karakter) 7 hücreye sağa yaslanır.

%s metin, %d tam sayı, %f kayan noktalı sayı. Türkçe yerel ayarlı bilgisayarda `%.2f` ondalık ayırıcı olarak virgül basabilir (17,50).

BİÇİMLEME

printf ile Biçimleme

```
String ad = "Java"; int adet = 3; double fiyat = 17.5;
System.out.printf("%-8s %3d %7.2f%n", ad, adet, fiyat);
```



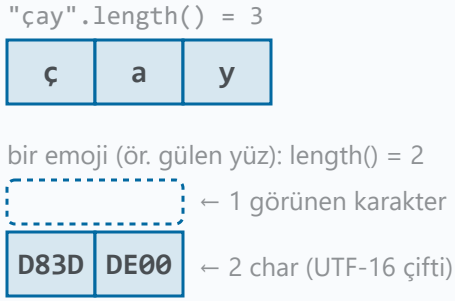
`%n` satır sonu ekler. Biçim belirteçleri argümanlarla **sırayla** eşleşir; tip uyuşmazsa hata oluşur.

Adım 4 / 4

%s metin, %d tam sayı, %f kayan noktalı sayı. Türkçe yerel ayarlı bilgisayarda `%.2f` ondalık ayırıcı olarak virgül basabilir (17,50).

Unicode ve Metin

- String, UTF-16 **kod birimlerinden** (char) oluşur; Türkçe harfler tek char'a sığar.
- Bazı görünür karakterler (emoji, bazı simgeler) iki char ya da birden fazla Unicode kod noktası içerir.
- Bu yüzden kullanıcıya görünen karakter sayısı için `length()` her durumda yeterli değildir.



ÖZET

Bölüm Özeti

Değiştirilemezlik

String nesnesi değiştirilemez; `toUpperCase`, `+`, `substring` yeni nesne döndürür.

İndis

0 … length() − 1; `substring(a, b)`: a dahil b hariç, uzunluk b − a.

Karşılaştırma

İçerik için `equals`, sıra için `compareTo`; `==` yalnızca referansı karşılaştırır.

Girdi / çıktı

`next` kelime, `nextLine` satır okur; yoğun birleştirmede StringBuilder, biçim için printf.

ÖZET

Kontrol Soruları

1. String neden değiştirilemez olarak adlandırılır?
2. substring başlangıç ve bitiş sınırlarını nasıl kullanır?
3. `==` ile `equals` arasındaki fark nedir?
4. next ve nextLine nasıl ayrılır?
5. Döngü içinde neden StringBuilder tercih edilebilir?

▶ **Cevaplar**

ÖZET

Alıştırmalar

1. Bir cümledeki büyük harf, küçük harf, rakam ve boşluk sayılarını ayrı ayrı bulun.
2. Bir kelimenin anagram olup olmadığını (ör. "elma" – "alem") karakter frekanslarıyla kontrol edin.
3. Kullanıcının girdiği ad-soyadı "Soyad, Ad" biçiminde ve soyadı büyük harfle yazdırın.
4. Bir cümledeki en uzun kelimeyi bulun (`split` ile).
5. Sezar şifresini büyük harfleri de (A–Z) destekleyecek şekilde genişletin; diğer karakterler aynen kalsın.
6. `replace` kullanmadan bir dizgideki tüm boşlukları silen kodu StringBuilder ile yazın.

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: **Strings**, **The StringBuilder Class**.
- Oracle, Java SE 21 API: **String**, **StringBuilder**.
- Sedgewick ve Wayne, Computer Science: An Interdisciplinary Approach, Addison-Wesley, 2016: **3.1 Using Data Types**.
- Unicode Consortium, The Unicode Standard.

Son

Dizgiler

Sonraki sunum: Özyineleme · Çalışma soruları · Simülatör: Dizgi Laboratuvarı · Terim sözlüğü