

Bölüm 5: Dizgi (String)

Java ile Nesne Yönelimli Programlama

String nesnesi

İndisleme

Temel metotlar

Değiştirilemezlik

StringBuilder

Scanner ile girdi

== ve equals()

Dizgi havuzu

compareTo

printf ve biçimlendirme

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

Öğrenme Çıktıları

Dizgi oluşturmak

`String` nesnesini dizgi sabitiyle (literal) ve `new` ile oluşturmak; `+` birleştirmesinin sırasını açıklamak.

Değiştirilemezlik

Metotların yeni nesne döndürdüğünü bellekte göstermek; döngüde `StringBuilder` kullanmak.

Karşılaştırma

`==` ile `equals()` farkını dizgi havuzuyla açıklamak; `compareTo` ve benzerlerini kullanmak.

İndis ve metotlar

`charAt`, `length`, `indexOf`, `substring` ile parçalara erişmek; dizgiyi döngüyle dolaşmak.

Kullanıcı girişi

`next()`, `nextLine()`, `nextInt()` farkını açıklamak; satır sonu tuzağından kaçınmak.

Biçimli çıktı

`printf` ve `String.format` ile genişlik, hizalama ve hassasiyeti ayarlamak.

Dizgi (String) Nedir?

- **Dizgi** (string), karakterlerden oluşan bir metni saklayan bir **nesnedir**; tipi `java.lang.String` sınıfıdır.
- `String` bir **sınıftır**, ilkel tip değildir; adı büyük harfle başlar. `import` gerekmez.
- Çift tırnak içindeki `"Tarkan"` gibi sabitler **dizgi sabitidir** (string literal); nesne `new` kullanılmadan oluşur.
- Karakterler **Unicode** ile saklanır: `ç ğ ı İ ö ş ü` gibi Türkçe harfler sorunsuzdur.
- Tek karakter `char` ile tek tırnakta (`'T'`), metin `String` ile çift tırnakta (`"T"`) yazılır.

```
String ad = "Tarkan";  
String adSoyad = "Mustafa Sandal";  
String nokta = "(" + 3 + ", " + 5 + ")";  
System.out.println(ad);  
System.out.println(adSoyad);  
System.out.println(nokta);
```

ÇIKTI

```
Tarkan  
Mustafa Sandal  
(3, 5)
```

T	a	r	k	a	n
0	1	2	3	4	5

Dizgi, sıralı karakter kutuları gibidir.

Dizgi Oluşturma ve Birleştirme (+)

```
String s0 = "Giresun";           // deyim
char[] harfler = {'J', 'a', 'v', 'a'};
String s1 = new String(harfler); // char[]
String s2 = String.valueOf(42);  // sayıdan
String s3 = "Giresun".concat(" Üni.");
System.out.println(s1 + " " + s2 + " " + s3);
```

```
System.out.println("Toplam: " + 3 + 5);
System.out.println("Toplam: " + (3 + 5));
System.out.println(3 + 5 + " elma");
System.out.println('a' + 'b' + "c");
System.out.println("" + 'a' + 'b');
```

ÇIKTI

```
Java 42 Giresun Üni.
Toplam: 35
Toplam: 8
8 elma
195c
ab
```

- `+` operatörünün bir yanı `String` ise diğer yan da dizgiye çevrilir ve **birleştirilir** (concatenation).
- `+` **soldan sağa** işlenir: `"Toplam: " + 3` önce `"Toplam: 3"` olur, sonra `5` eklenir → `Toplam: 35`.
- Toplamı görmek için parantez kullanın: `"Toplam: " + (3 + 5)`.
- `3 + 5 + " elma"`: Önce sayılar toplanır (8), sonra dizgi eklenir.
- `'a' + 'b'` iki `char` 'ın **sayısal** toplamıdır ($97 + 98 = 195$). Başa `""` koymak dizgi birleştirmeyi zorlar.
- Her birleştirme **yeni** bir `String` nesnesi üretir.

+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
► int a = 4, b = 2;  
2 String s1 = "Sonuç: " + a + b;  
3 String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
5 String s4 = 'x' + 'y' + "z";  
6 String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```

a = 4 b = 2

int int

İki `int` değişken. Her satırda `+` soldan sağa işlenir; her adımda tip kontrol edilir.

Adım 1 / 13

DEĞİŞKENLER

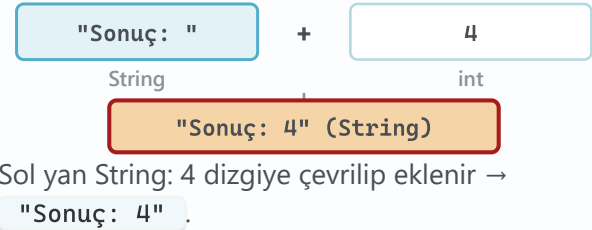
a = 4

b = 2

ÇIKTI

+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 ▶ String s1 = "Sonuç: " + a + b;  
3 String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
5 String s4 = 'x' + 'y' + "z";  
6 String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```



Adım 2 / 13

DEĞİŞKENLER

a = 4

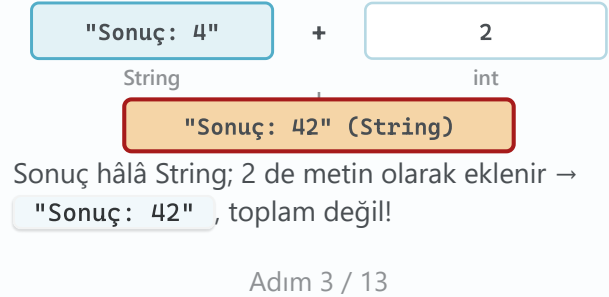
b = 2

ÇIKTI



+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 ▶ String s1 = "Sonuç: " + a + b;  
3 String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
5 String s4 = 'x' + 'y' + "z";  
6 String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```



DEĞİŞKENLER

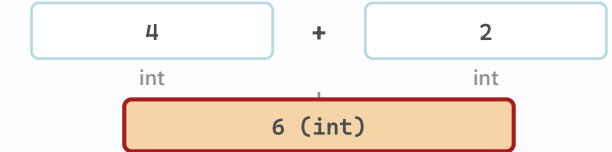
s1 = Sonuç: 42

ÇIKTI



+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 String s1 = "Sonuç: " + a + b;  
3 ► String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
5 String s4 = 'x' + 'y' + "z";  
6 String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```



Parantez önce: 4 + 2 iki int, sayısal toplam 6.

Adım 4 / 13

DEĞİŞKENLER

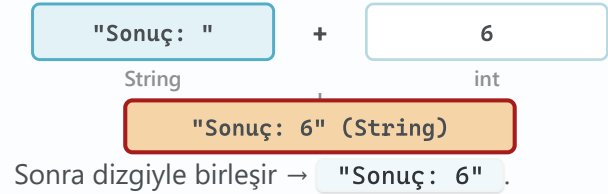
s1 = Sonuç: 42

ÇIKTI



+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 String s1 = "Sonuç: " + a + b;  
3 ► String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
5 String s4 = 'x' + 'y' + "z";  
6 String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```



Adım 5 / 13

DEĞİŞKENLER

s1 = Sonuç: 42

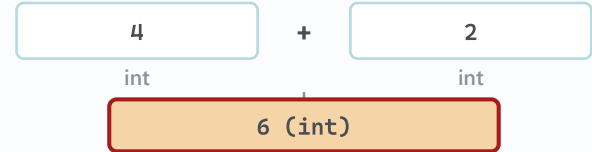
s2 = Sonuç: 6

ÇIKTI



+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;
2 String s1 = "Sonuç: " + a + b;
3 String s2 = "Sonuç: " + (a + b);
4 ► String s3 = a + b + " kalem";
5 String s4 = 'x' + 'y' + "z";
6 String s5 = "" + 'x' + 'y';
7 System.out.println(s1 + " | " + s2);
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```



Soldaki iki yan da int: önce toplanır → 6.

Adım 6 / 13

DEĞİŞKENLER

s1 = Sonuç: 42

s2 = Sonuç: 6

ÇIKTI

+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 String s1 = "Sonuç: " + a + b;  
3 String s2 = "Sonuç: " + (a + b);  
4 ► String s3 = a + b + " kalem";  
5 String s4 = 'x' + 'y' + "z";  
6 String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```



Adım 7 / 13

DEĞİŞKENLER

s1 = Sonuç: 42

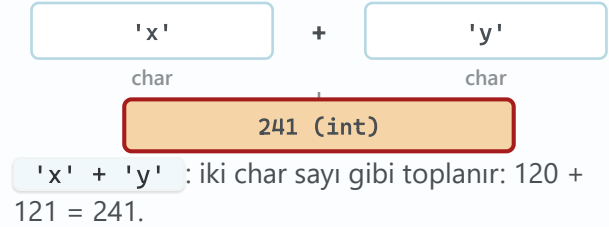
s2 = Sonuç: 6

s3 = 6 kalem

ÇIKTI

+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 String s1 = "Sonuç: " + a + b;  
3 String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
► String s4 = 'x' + 'y' + "z";  
6 String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```



Adım 8 / 13

DEĞİŞKENLER

s1 = Sonuç: 42

s2 = Sonuç: 6

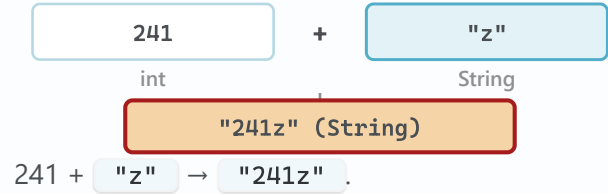
s3 = 6 kalem

ÇIKTI



+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 String s1 = "Sonuç: " + a + b;  
3 String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
► String s4 = 'x' + 'y' + "z";  
6 String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```



Adım 9 / 13

DEĞİŞKENLER

s1 = Sonuç: 42

s2 = Sonuç: 6

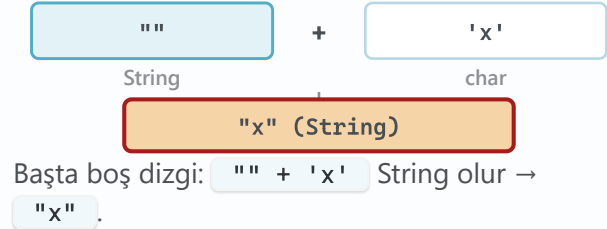
s3 = 6 kalem

s4 = 241z

ÇIKTI

+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 String s1 = "Sonuç: " + a + b;  
3 String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
5 String s4 = 'x' + 'y' + "z";  
▶ String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```



Adım 10 / 13

DEĞİŞKENLER

s1 = Sonuç: 42

s2 = Sonuç: 6

s3 = 6 kalem

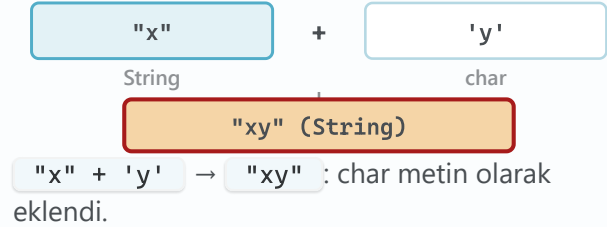
s4 = 241z

ÇIKTI

```
Sonuç: 42 | Sonuç: 6  
6 kalem | 241z | 
```

+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 String s1 = "Sonuç: " + a + b;  
3 String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
5 String s4 = 'x' + 'y' + "z";  
▶ String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```



Adım 11 / 13

DEĞİŞKENLER

s1 = Sonuç: 42

s2 = Sonuç: 6

s3 = 6 kalem

s4 = 241z

s5 = xy

ÇIKTI

+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 String s1 = "Sonuç: " + a + b;  
3 String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
5 String s4 = 'x' + 'y' + "z";  
6 String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
8 System.out.println(s3 + " | " + s4 + " | " + s5);
```

println

birleştirilmiş dizgiler yazılır

İlk satır yazılır.

Adım 12 / 13

DEĞİŞKENLER

s1 = Sonuç: 42

s2 = Sonuç: 6

s3 = 6 kalem

s4 = 241z

s5 = xy

ÇIKTI

Sonuç: 42 | Sonuç: 6

+ Soldan Sağa: Toplama mı, Birleştirme mi?

```
1 int a = 4, b = 2;  
2 String s1 = "Sonuç: " + a + b;  
3 String s2 = "Sonuç: " + (a + b);  
4 String s3 = a + b + " kalem";  
5 String s4 = 'x' + 'y' + "z";  
6 String s5 = "" + 'x' + 'y';  
7 System.out.println(s1 + " | " + s2);  
▶ System.out.println(s3 + " | " + s4 + " | " + s5);
```

println

birleştirilmiş dizgiler yazılır

İkinci satır yazılır. Kural: bir yan String olduğu anda birleştirme; o ana kadar sayılar toplanır.

Adım 13 / 13

DEĞİŞKENLER

s1 = Sonuç: 42

s2 = Sonuç: 6

s3 = 6 kalem

s4 = 241z

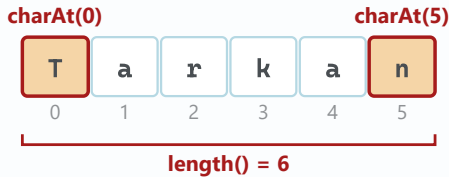
s5 = xy

ÇIKTI

```
Sonuç: 42 | Sonuç: 6  
6 kalem | 241z | xy
```

İndisleme (Indexing)

- İlk karakterin indisi **0**, sonuncunun `length() - 1`.
- `charAt(i)` : `i` indisli karakter (`char`).
- `length()` bir **metottur**, parantez gerekir (dizilerdeki `length` ise alandır).
- Uzunluğu aşan ya da negatif indis: çalışma zamanında `StringIndexOutOfBoundsException`.
- `char` bir sayıdır (Unicode kodu): `c + 1` sayı, `(char) (c + 1)` sonraki harf.



```
String isim = "Tarkan";
System.out.println(isim.length());
System.out.println(isim.charAt(0));
System.out.println(isim.charAt(isim.length() - 1));
char c = isim.charAt(2);
System.out.println(c + 1);
System.out.println((char) (c + 1));
```

ÇIKTI

```
6
T
n
115
s
```

```
String s = "Java";
System.out.println(s.charAt(4));
```

```
Exception in thread "main"
java.lang.StringIndexOutOfBoundsException:
    Index 4 out of bounds for length 4
```

Karakterleri Dolaşmak: Adım Adım

```
1 Scanner scanner = new Scanner(System.in);
2 ▶ System.out.print("Bir kelime giriniz: ");
3 String kelime = scanner.nextLine();
4 for (int i = 0; i < kelime.length(); i += 2) {
5     char harf = kelime.charAt(i);
6     System.out.println((i + 1) + ". harf: " + harf);
7 }
8 scanner.close();
```

kelime henüz okunmadı

Ekrana istem yazılır; `print` alt satıra geçmez.

Adım 1 / 8

DEĞİŞKENLER

—

ÇIKTI

Bir kelime giriniz:

Karakterleri Dolaşmak: Adım Adım

```
1 Scanner scanner = new Scanner(System.in);
2 System.out.print("Bir kelime giriniz: ");
3 ▶ String kelime = scanner.nextLine();
4 for (int i = 0; i < kelime.length(); i += 2) {
5     char harf = kelime.charAt(i);
6     System.out.println((i + 1) + ". harf: " + harf);
7 }
8 scanner.close();
```

G	i	r	e	s	u	n
0	1	2	3	4	5	6

Kullanıcı **Giresun** yazıp Enter'a basar;
`nextLine()` satırın tamamını okur.

Adım 2 / 8

DEĞİŞKENLER

`kelime = "Giresun"`

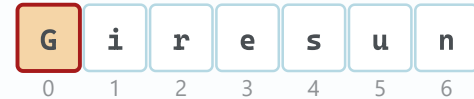
ÇIKTI

Bir kelime giriniz: Giresun

Karakterleri Dolaşmak: Adım Adım

```
1 Scanner scanner = new Scanner(System.in);
2 System.out.print("Bir kelime giriniz: ");
3 String kelime = scanner.nextLine();
4 for (int i = 0; i < kelime.length(); i += 2) {
5     ▶ char harf = kelime.charAt(i);
6     ▶ System.out.println((i + 1) + ". harf: " + harf);
7 }
8 scanner.close();
```

i = 0



i = 0 : `charAt(0)` = 'G'; satır yazdırılır,
sonra `i += 2` .

Adım 3 / 8

DEĞİŞKENLER

`kelime = "Giresun"`

`i = 0`

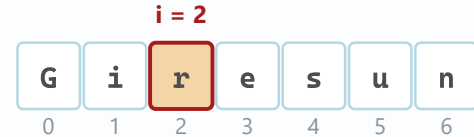
`harf = 'G'`

ÇIKTI

```
Bir kelime giriniz: Giresun
1. harf: G
```

Karakterleri Dolaşmak: Adım Adım

```
1 Scanner scanner = new Scanner(System.in);
2 System.out.print("Bir kelime giriniz: ");
3 String kelime = scanner.nextLine();
4 for (int i = 0; i < kelime.length(); i += 2) {
5     ▶ char harf = kelime.charAt(i);
6     ▶ System.out.println((i + 1) + ". harf: " + harf);
7 }
8 scanner.close();
```



i = 2 : `charAt(2)` = 'r'; satır yazdırılır,
sonra **i += 2** .

Adım 4 / 8

DEĞİŞKENLER

kelime = "Giresun"

i = 2

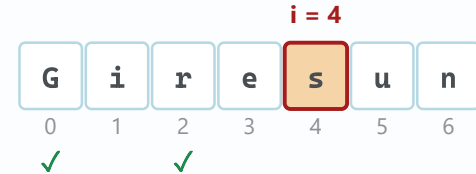
harf = 'r'

ÇIKTI

```
Bir kelime giriniz: Giresun
1. harf: G
3. harf: r
```

Karakterleri Dolaşmak: Adım Adım

```
1 Scanner scanner = new Scanner(System.in);
2 System.out.print("Bir kelime giriniz: ");
3 String kelime = scanner.nextLine();
4 for (int i = 0; i < kelime.length(); i += 2) {
5     ▶ char harf = kelime.charAt(i);
6     ▶ System.out.println((i + 1) + ". harf: " + harf);
7 }
8 scanner.close();
```



`i = 4` : `charAt(4)` = 's'; satır yazdırılır,
sonra `i += 2` .

Adım 5 / 8

DEĞİŞKENLER

`kelime = "Giresun"`

`i = 4`

`harf = 's'`

ÇIKTI

Bir kelime giriniz: Giresun

1. harf: G

3. harf: r

5. harf: s

Karakterleri Dolaşmak: Adım Adım

```
1 Scanner scanner = new Scanner(System.in);
2 System.out.print("Bir kelime giriniz: ");
3 String kelime = scanner.nextLine();
4 for (int i = 0; i < kelime.length(); i += 2) {
5     ▶ char harf = kelime.charAt(i);
6     ▶ System.out.println((i + 1) + ". harf: " + harf);
7 }
8 scanner.close();
```



`i = 6` : `charAt(6)` = 'n'; satır yazdırılır,
sonra `i += 2` .

Adım 6 / 8

DEĞİŞKENLER

`kelime = "Giresun"`

`i = 6`

`harf = 'n'`

ÇIKTI

Bir kelime giriniz: Giresun

1. harf: G

3. harf: r

5. harf: s

7. harf: n

Karakterleri Dolaşmak: Adım Adım

```
1 Scanner scanner = new Scanner(System.in);
2 System.out.print("Bir kelime giriniz: ");
3 String kelime = scanner.nextLine();
4 ► for (int i = 0; i < kelime.length(); i += 2) {
5     char harf = kelime.charAt(i);
6     System.out.println((i + 1) + ". harf: " + harf);
7 }
8 scanner.close();
```

$i = 8 \rightarrow 8 < 7$ yanlış

G	i	r	e	s	u	n
0	1	2	3	4	5	6
✓		✓		✓		✓

$i = 8$ için **$8 < 7$** yanlış: döngü biter. Son indis 6'dır, 7 kullanılmaz.

Adım 7 / 8

DEĞİŞKENLER

kelime = "Giresun"

$i = 8$

ÇIKTI

```
Bir kelime giriniz: Giresun
1. harf: G
3. harf: r
5. harf: s
7. harf: n
```

Karakterleri Dolaşmak: Adım Adım

```
1 Scanner scanner = new Scanner(System.in);
2 System.out.print("Bir kelime giriniz: ");
3 String kelime = scanner.nextLine();
4 for (int i = 0; i < kelime.length(); i += 2) {
5     char harf = kelime.charAt(i);
6     System.out.println((i + 1) + ". harf: " + harf);
7 }
▶ scanner.close();
```

$i = 8 \rightarrow 8 < 7$ yanlış

G	i	r	e	s	u	n
0	1	2	3	4	5	6
✓		✓		✓		✓

Scanner kapatılır, program sona erer.

Adım 8 / 8

DEĞİŞKENLER

kelime = "Giresun"

ÇIKTI

```
Bir kelime giriniz: Giresun
1. harf: G
3. harf: r
5. harf: s
7. harf: n
```

DİZGİ METOTLARI

Temel Dizgi Metotları

Metot	Ne yapar?	s = "Java Programlama" için
length()	Karakter sayısını döndürür.	16
charAt(i)	i indisli karakter (char).	s.charAt(0) → 'J'
indexOf(str)	Alt dizginin ilk başlangıç indisi; bulunamazsa -1.	s.indexOf("Programlama") → 5 s.indexOf("C++") → -1
indexOf(str, bas)	Aramaya bas indisinden başlar.	s.indexOf("a", 4) → 10
lastIndexOf(str)	Son geçtiği yerin indisi.	s.lastIndexOf('a') → 15
substring(i1, i2)	i1 dahil, i2 hariç arasındaki alt dizgi.	s.substring(5, 12) → "Program"
substring(i1)	i1 'den sona kadar.	s.substring(5) → "Programlama"
toLowerCase()	Küçük harfli yeni dizgi.	"java programlama"
toUpperCase()	Büyük harfli yeni dizgi.	"JAVA PROGRAMLAMA"

Hiçbir metot s 'yi değiştirmez; sonuç her zaman yeni bir değer olarak döner.

indexOf ve substring: Adım Adım



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

Adım 1 / 7

```
► String str = "Java Programlama";  
2 int indeks = str.indexOf("Programlama");  
3 int uzunluk = str.length();  
4 String alt = str.substring(5, 12);  
5 String son = str.substring(5);  
6 String kucuk = str.toLowerCase();  
7 String buyuk = str.toUpperCase();
```

DEĞİŞKENLER

str = "Java Programlama"

indexOf ve substring: Adım Adım



`indexOf("Programlama")` eşleşmenin **başladığı** indisi verir: 5.

Adım 2 / 7

```
1 String str = "Java Programlama";  
▶ 2 int indeks = str.indexOf("Programlama");  
3 int uzunluk = str.length();  
4 String alt = str.substring(5, 12);  
5 String son = str.substring(5);  
6 String kucuk = str.toLowerCase();  
7 String buyuk = str.toUpperCase();
```

DEĞİŞKENLER

`str = "Java Programlama"`

`indeks = 5`

indexOf ve substring: Adım Adım



16 karakter

`length()` = 16; indisler 0 ... 15.

Adım 3 / 7

```
1 String str = "Java Programlama";
2 int indeks = str.indexOf("Programlama");
3 ► int uzunluk = str.length();
4 String alt = str.substring(5, 12);
5 String son = str.substring(5);
6 String kucuk = str.toLowerCase();
7 String buyuk = str.toUpperCase();
```

DEĞİŞKENLER

`str = "Java Programlama"`

`indeks = 5`

`uzunluk = 16`

indexOf ve substring: Adım Adım



`substring(5, 12)` : 5, 6, ..., 11 → **"Program"** (12 - 5 = 7 karakter).

Adım 4 / 7

```
1 String str = "Java Programlama";
2 int indeks = str.indexOf("Programlama");
3 int uzunluk = str.length();
4 ► String alt = str.substring(5, 12);
5 String son = str.substring(5);
6 String kucuk = str.toLowerCase();
7 String buyuk = str.toUpperCase();
```

DEĞİŞKENLER

`str = "Java Programlama"` `indeks = 5`

`uzunluk = 16` `alt = "Program"`

indexOf ve substring: Adım Adım

5'ten sona kadar



`substring(5)` → "Programlama".

Adım 5 / 7

```
1 String str = "Java Programlama";  
2 int indeks = str.indexOf("Programlama");  
3 int uzunluk = str.length();  
4 String alt = str.substring(5, 12);  
5 ► String son = str.substring(5);  
6 String kucuk = str.toLowerCase();  
7 String buyuk = str.toUpperCase();
```

DEĞİŞKENLER

`str = "Java Programlama"`

`uzunluk = 16`

`alt = "Program"`

`son = "Programlama"`

indexOf ve substring: Adım Adım

yeni nesne



`toLowerCase()` **yeni** bir dizgi döndürür; `str` hâlâ "Java Programlama".

Adım 6 / 7

```
1 String str = "Java Programlama";  
2 int indeks = str.indexOf("Programlama");  
3 int uzunluk = str.length();  
4 String alt = str.substring(5, 12);  
5 String son = str.substring(5);  
6 String kucuk = str.toLowerCase();  
7 String buyuk = str.toUpperCase();
```

DEĞİŞKENLER

`str = "Java Programlama"`

`alt = "Program"`

`son = "Programlama"`

`kucuk = "java programlama"`

indexOf ve substring: Adım Adım

yeni nesne



`toUpperCase()` da yeni dizgi döndürür. Orijinal dizgi hiç değişmedi.

Adım 7 / 7

```
1 String str = "Java Programlama";  
2 int indeks = str.indexOf("Programlama");  
3 int uzunluk = str.length();  
4 String alt = str.substring(5, 12);  
5 String son = str.substring(5);  
6 String kucuk = str.toLowerCase();  
► String buyuk = str.toUpperCase();
```

DEĞİŞKENLER

`str = "Java Programlama"`

`kucuk = "java programlama"`

`buyuk = "JAVA PROGRAMLAMA"`

Diğer Yararlı Metotlar

```
String s = " Merhaba Dünya ";
System.out.println("[ " + s.trim() + " ]");
System.out.println("[ " + s.replace('a', 'e') + " ]");
String[] meyveler = "elma,armut,kiraz".split(",");
System.out.println(meyveler.length);
System.out.println(meyveler[1]);
System.out.println(String.join(" - ", meyveler));
System.out.println("ab".repeat(3));           // Java 11+
System.out.println("").isEmpty());
System.out.println("   ".isBlank());           // Java 11+
char[] dizi = "Java".toCharArray();
System.out.println(dizi.length + " " + dizi[0]);
System.out.println("Giresun".replace("sun", "SUN"));
System.out.println(String.valueOf(3.5) + 1);
System.out.println(Integer.parseInt("42") + 1);
```

ÇIKTI

```
[Merhaba Dünya]
[ Merhebe Dünya ]
3
armut
elma - armut - kiraz
ababab
true
true
4 J
GireSUN
3.51
43
```

- `trim()` : baştaki/sondaki boşlukları atar.
`replace` : tüm eşleşmeleri değiştirir.
- `split(ayrac)` : parçaları `String[]` dizisi olarak verir; `String.join` tersini yapar.
- `String.valueOf(x)` : sayı → dizgi;
`Integer.parseInt(s)` ,
`Double.parseDouble(s)` : dizgi → sayı.

Değiştirilemez (Immutable) Nesne

- `String` nesneleri oluşturulduktan sonra **değiştirilemez** (immutable).
- `toUpperCase`, `replace`, `substring`, `+` orijinali değiştirmez; **yeni bir nesne** döndürür.
- Sonucu kullanmak için **atamak** gerekir: `s = s.toUpperCase();`
- Faydaları: güvenli ve tutarlı kod (dizgiyi paylaşan kodlar birbirini bozamaz), **dizgi havuzu** (string pooling) ile bellek tasarrufu, iş parçacıkları arasında güvenli paylaşım.

Benzetme: dizgi bir **taş tablettir**. Kazıyıp düzeltemezsiniz; yeni bir tablet yazıp eskisinin yerine koyarsınız.



```
String s = "Giresun Üniversitesi";
s.toUpperCase(); // sonuç kaybolur
System.out.println(s);
s = s.toUpperCase(); // sonuç s'ye atanır
System.out.println(s);
```

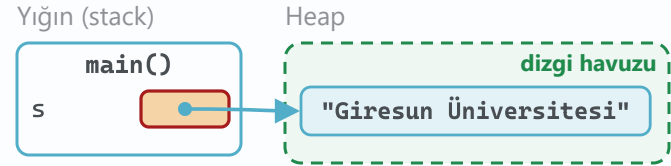
ÇIKTI

```
Giresun Üniversitesi
GİRESUN ÜNİVERSİTESİ
```

Türkçe yerel ayarlı (tr-TR) bilgisayarda çıktı **GİRESUN ÜNİVERSİTESİ**, İngilizce ayarda **GİRESUN ÜNİVERSİTESI** olur.

Değiştirilemezlik: Bellekte Adım Adım

```
► String s = "Giresun Üniversitesi";  
2 s.toUpperCase();  
3 System.out.println(s);  
4 s = s.toUpperCase();  
5 System.out.println(s);
```



Dizgi sabiti havuzunda bir nesnedir; `s` onu gösterir.

Adım 1 / 5

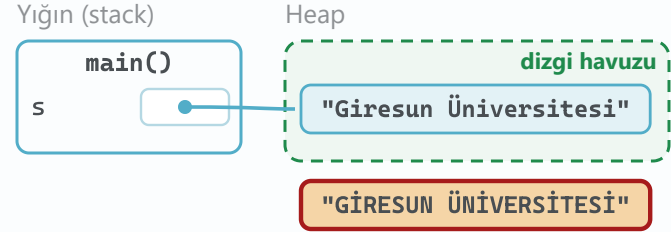
DEĞİŞKENLER

`s = →#1 "Giresun Üniversitesi"`

ÇIKTI

Değiştirilemezlik: Bellekte Adım Adım

```
1 String s = "Giresun Üniversitesi";  
▶ s.toUpperCase();  
3 System.out.println(s);  
4 s = s.toUpperCase();  
5 System.out.println(s);
```



`toUpperCase()` **yeni** bir nesne üretir. Ama sonuç hiçbir değişkene atanmadı!

Adım 2 / 5

DEĞİŞKENLER

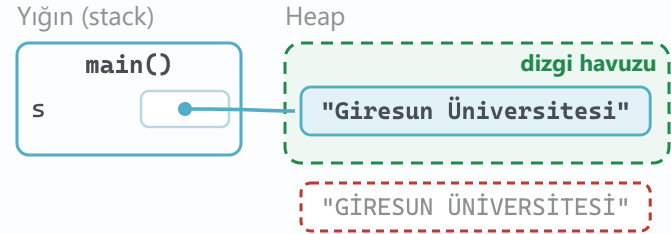
s = →#1 "Giresun Üniversitesi"

ÇIKTI



Değiştirilemezlik: Bellekte Adım Adım

```
1 String s = "Giresun Üniversitesi";  
2 s.toUpperCase();  
▶ System.out.println(s);  
4 s = s.toUpperCase();  
5 System.out.println(s);
```



s hâlâ eski nesneyi gösterir: orijinal metin yazılır. Sahipsiz nesne çöp toplayıcıya kalır.

Adım 3 / 5

DEĞİŞKENLER

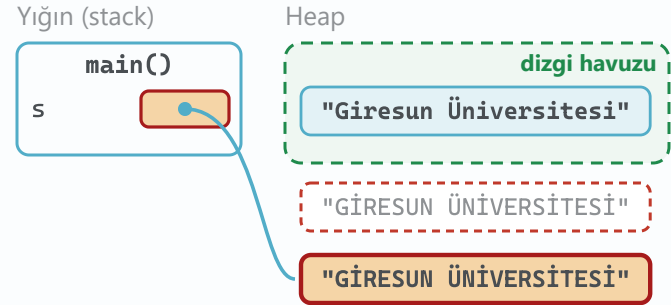
s = →#1 "Giresun Üniversitesi"

ÇIKTI

Giresun Üniversitesi

Değiştirilemezlik: Bellekte Adım Adım

```
1 String s = "Giresun Üniversitesi";  
2 s.toUpperCase();  
3 System.out.println(s);  
▶ s = s.toUpperCase();  
5 System.out.println(s);
```



Yine yeni bir nesne oluşur ve bu kez **referansı s 'ye atanır**. Eski nesne değişmedi, yalnızca ok yer değiştirdi.

Adım 4 / 5

DEĞİŞKENLER

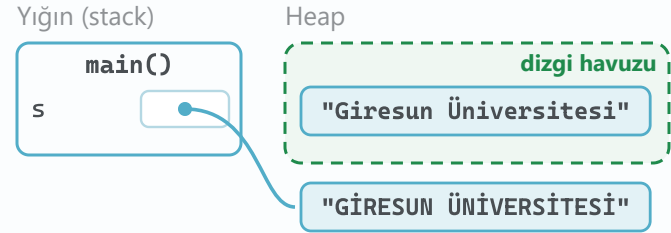
s = →#3 "GİRESUN ÜNİVERSİTESİ"

ÇIKTI

Giresun Üniversitesi

Değiştirilemezlik: Bellekte Adım Adım

```
1 String s = "Giresun Üniversitesi";  
2 s.toUpperCase();  
3 System.out.println(s);  
4 s = s.toUpperCase();  
▶ System.out.println(s);
```



s artık büyük harfli nesneyi gösterir.

Adım 5 / 5

DEĞİŞKENLER

s = →#3 "GİRESUN ÜNİVERSİTESİ"

ÇIKTI

```
Giresun Üniversitesi  
GİRESUN ÜNİVERSİTESİ
```

Türkçe Harfler ve Yerel Ayar (Locale)

```
import java.util.Locale;

String s = "Giresun Üniversitesi";
System.out.println(s.toUpperCase());
System.out.println(s.toUpperCase(Locale.ROOT));
System.out.println("TITLE".toLowerCase());
System.out.println("TITLE".toLowerCase().equals("title"));
System.out.println("TITLE".toLowerCase(Locale.ROOT));
```

ÇIKTI (TR-TR)

```
GİRESUN ÜNİVERSİTESİ
GİRESUN ÜNİVERSİTESİ
title
false
title
```

Dönüşüm	Türkçe (tr)	ROOT / İngilizce
İ → büyük	İ	I
I → küçük	ı	i
ı → büyük	İ	I

- Parametresiz
`toUpperCase()` / `toLowerCase()`
bilgisayarın **varsayılan yerel ayarını** kullanır: aynı program farklı makinede farklı sonuç verebilir.
- Türkçe ayarda `"TITLE".toLowerCase()`
→ `"title"` : karşılaştırma `false` !
- Komut, dosya adı gibi metinlerde
`Locale.ROOT` ; Türkçe metinde
`Locale.forLanguageTag("tr-TR")`
verin.

Döngüde Birleştirme ve StringBuilder

```
String sonuc = "";
for (int i = 1; i <= 5; i++) {
    sonuc += i + " ";    // her turda yeni String
}
System.out.println(sonuc);

StringBuilder sb = new StringBuilder();
for (int i = 1; i <= 5; i++) {
    sb.append(i).append(' ');    // aynı nesne büyür
}
String metin = sb.toString();
System.out.println(metin);

StringBuilder ters = new StringBuilder("Giresun");
ters.reverse();
System.out.println(ters);
StringBuilder k = new StringBuilder("Java");
k.insert(0, ">> ").append('!');
k.setCharAt(3, 'j');
System.out.println(k + " " + k.length());
```

ÇIKTI

```
1 2 3 4 5
1 2 3 4 5
nuseriG
>> java! 8
```

- `sonuc += ...` her turda eski metni kopyalayıp **yeni** bir nesne oluşturur; n turda yaklaşık $n^2/2$ karakter kopyalanır.
- **StringBuilder** **değiştirilebilir** (mutable) bir karakter tamponudur: `append`, `insert`, `reverse`, `setCharAt`, `deleteCharAt` aynı nesne üzerinde çalışır.
- İş bitince `toString()` ile `String` 'e çevrilir.
- Birkaç parçalık birleştirmede `+` yeterli ve okunaklıdır; **döngü içinde** çok birleştirmede **StringBuilder** tercih edilir.

StringBuilder: Tek Tampon, Yerinde Değişiklik

```
► StringBuilder sb = new StringBuilder("Java");  
2 sb.append(' ').append(8);  
3 sb.insert(0, ">> ");  
4 sb.setCharAt(3, 'j');  
5 sb.deleteCharAt(sb.length() - 1);  
6 sb.append("programlama dili");  
7 String s = sb.toString();  
8 System.out.println(s + " " + sb.length() + " " +  
  sb.capacity());
```

length = 4

capacity = 20



Tek bir değiştirilebilir tampon: 4 karakter dolu, kapasite $16 + 4 = 20$.

Adım 1 / 9

DEĞİŞKENLER

length = 4

capacity = 20

ÇIKTI

```
>>> Java programlama dili 20
```

StringBuilder: Tek Tampon, Yerinde Değişiklik

```
1 StringBuilder sb = new StringBuilder("Java");
2 ▶ sb.append(' ').append(8);
3 sb.insert(0, ">> ");
4 sb.setCharAt(3, 'j');
5 sb.deleteCharAt(sb.length() - 1);
6 sb.append("programlama dili");
7 String s = sb.toString();
8 System.out.println(s + " " + sb.length() + " " +
  sb.capacity());
```

length = 5

capacity = 20



`append(' ')` : boşluk sona eklenir; **aynı nesne**, yeni nesne yok.

Adım 2 / 9

DEĞİŞKENLER**length = 5**

capacity = 20

ÇIKTI

```
>>> Java8 programlama dili 20
```

StringBuilder: Tek Tampon, Yerde Değişiklik

```
1 StringBuilder sb = new StringBuilder("Java");
2 ▶ sb.append(' ').append(8);
3 sb.insert(0, ">> ");
4 sb.setCharAt(3, 'j');
5 sb.deleteCharAt(sb.length() - 1);
6 sb.append("programlama dili");
7 String s = sb.toString();
8 System.out.println(s + " " + sb.length() + " " +
  sb.capacity());
```

length = 6

capacity = 20



Zincirdeki ikinci `append(8)` : sayı karakter olarak eklenir. append nesnenin kendisini döndürdüğü için zincir kurulabilir.

Adım 3 / 9

DEĞİŞKENLER

length = 6

capacity = 20

ÇIKTI

```
>>> Java 8 programlama dili 20
```

StringBuilder: Tek Tampon, Yerde Değişiklik

```
1 StringBuilder sb = new StringBuilder("Java");
2 sb.append(' ').append(8);
3 ▶ sb.insert(0, ">> ");
4 sb.setCharAt(3, 'j');
5 sb.deleteCharAt(sb.length() - 1);
6 sb.append("programlama dili");
7 String s = sb.toString();
8 System.out.println(s + " " + sb.length() + " " +
  sb.capacity());
```

length = 9

capacity = 20



`insert(0, ">> ")` : baştaki 3 hücreye yer açmak için diğer karakterler sağa kayar.

Adım 4 / 9

DEĞİŞKENLER**length = 9**

capacity = 20

ÇIKTI

```
>> Java8 programlama dili 20
```

StringBuilder: Tek Tampon, Yerinde Değişiklik

```
1 StringBuilder sb = new StringBuilder("Java");
2 sb.append(' ').append(8);
3 sb.insert(0, ">> ");
4 ► sb.setCharAt(3, 'j');
5 sb.deleteCharAt(sb.length() - 1);
6 sb.append("programlama dili");
7 String s = sb.toString();
8 System.out.println(s + " " + sb.length() + " " +
  sb.capacity());
```

length = 9

capacity = 20



`setCharAt(3, 'j')` : 3. indis yerinde değişti. String'de bu mümkün değildi.

Adım 5 / 9

DEĞİŞKENLER

length = 9

capacity = 20

ÇIKTI

```
>>> j a v a 8  20
```

StringBuilder: Tek Tampon, Yerde Değişiklik

```
1 StringBuilder sb = new StringBuilder("Java");
2 sb.append(' ').append(8);
3 sb.insert(0, ">> ");
4 sb.setCharAt(3, 'j');
5 ► sb.deleteCharAt(sb.length() - 1);
6 sb.append("programlama dili");
7 String s = sb.toString();
8 System.out.println(s + " " + sb.length() + " " +
  sb.capacity());
```

length = 8

capacity = 20



▲ silindi

`deleteCharAt(7)` : sondaki '8' silindi,
uzunluk 8.

Adım 6 / 9

DEĞİŞKENLER**length = 8**

capacity = 20

ÇIKTI

```
>>> j a v a programlama dili 8 20
```

StringBuilder: Tek Tampon, Yerde Değişiklik

```
1 StringBuilder sb = new StringBuilder("Java");
2 sb.append(' ').append(8);
3 sb.insert(0, ">> ");
4 sb.setCharAt(3, 'j');
5 sb.deleteCharAt(sb.length() - 1);
▶ sb.append("programlama dili");
7 String s = sb.toString();
8 System.out.println(s + " " + sb.length() + " " +
  sb.capacity());
```

length = 24**capacity = 42****20 → 42: yeni dizi, içerik kopyalandı**

16 karakter daha: $24 > 20$, sığmaz. Tampon büyür: yeni kapasite $(20 + 1) \times 2 = 42$, eski içerik kopyalanır.

Adım 7 / 9

DEĞİŞKENLER

length = 24**capacity = 42**

ÇIKTI



StringBuilder: Tek Tampon, Yerinde Değişiklik

```
1 StringBuilder sb = new StringBuilder("Java");
2 sb.append(' ').append(8);
3 sb.insert(0, ">> ");
4 sb.setCharAt(3, 'j');
5 sb.deleteCharAt(sb.length() - 1);
6 sb.append("programlama dili");
▶ String s = sb.toString();
8 System.out.println(s + " " + sb.length() + " " +
  sb.capacity());
```

length = 24

capacity = 42

>	>	.	j	a	v	a	.	p	r	o	g	r	a	m	l	a	m	a	.	d	i	l	i	.	.	.
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

s: yeni String (değiştirilemez)

toString() : tamponun o anki içeriğiyle
yeni bir String oluşur.

Adım 8 / 9

DEĞİŞKENLER

length = 24

capacity = 42

s = ">> java programlama dili"

ÇIKTI

```
>> java programlama dili 24 42
```

StringBuilder: Tek Tampon, Yerinde Değişiklik

```
1 StringBuilder sb = new StringBuilder("Java");
2 sb.append(' ').append(8);
3 sb.insert(0, ">> ");
4 sb.setCharAt(3, 'j');
5 sb.deleteCharAt(sb.length() - 1);
6 sb.append("programlama dili");
7 String s = sb.toString();
▶ System.out.println(s + " " + sb.length() + " " + sb.capacity());
```

length = 24

capacity = 42

>	>	.	j	a	v	a	.	p	r	o	g	r	a	m	l	a	m	a	.	d	i	l	i	.	.	.
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

s: yeni String (değiştirilemez)

Uzunluk 24, kapasite 42 yazılır.

Adım 9 / 9

DEĞİŞKENLER

length = 24

capacity = 42

s = ">> java programlama dili"

ÇIKTI

>> java programlama dili 24 42

Kullanıcı Girişi: Scanner

- `java.util.Scanner` sınıfı klavyeden (`System.in`) girdi okur: `import java.util.Scanner;`
- `next()` : boşlukları atlar, bir sonraki **kelimeyi** (boşluğa kadar) okur.
- `nextLine()` : bulunduğu yerden **satır sonuna kadar** her şeyi okur; satır sonu karakterini tüketir ama sonuca katmaz.
- `nextInt()` , `nextDouble()` : bir sayı okur; arkasındaki satır sonunu **tamponda bırakır**.
- İş bitince `klavye.close()` .

```
Scanner klavye = new Scanner(System.in);
System.out.println("bir metin giriniz");
String girdi = klavye.next();
System.out.println(girdi);    // bilgisayar
girdi = klavye.next();
System.out.println(girdi);    // ve
girdi = klavye.nextLine();
System.out.println(girdi);    // programlamaya giriş
```

GİRDİ (KLAVYE)

bilgisayar ve programlamaya giriş

ÇIKTI

bir metin giriniz
bilgisayar
ve
programlamaya giriş

`nextLine()` sonucunun başında bir **boşluk** vardır: "ve" kelimesinden sonraki boşluk da satırın parçasıdır.

next() ve nextLine(): Adım Adım

Girdi tamponu (boş)

(kullanıcı henüz yazmadı)

İstem yazılır; program kullanıcının satırı girmesini bekler.

Adım 1 / 7

```
1 Scanner klavye = new Scanner(System.in);  
▶ System.out.println("bir metin giriniz");  
3 String girdi = klavye.next();  
4 System.out.println(girdi);  
5 girdi = klavye.next();  
6 System.out.println(girdi);  
7 girdi = klavye.nextLine();  
8 System.out.println(girdi);
```

DEĞİŞKENLER

—

ÇIKTI

bir metin giriniz

next() ve nextLine(): Adım Adım

Girdi tamponu

b i l g i s a y a r _ v e _ p r o g r a m l a m a y a _ g i r i ş ↵

▲
imleç

Kullanıcı satırı yazıp Enter'a basar (tamponda ↵ de vardır). `next()` ilk boşluğa kadar okur: **"bilgisayar"**.

Adım 2 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.println("bir metin giriniz");
▶ 3 String girdi = klavye.next();
4 System.out.println(girdi);
5 girdi = klavye.next();
6 System.out.println(girdi);
7 girdi = klavye.nextLine();
8 System.out.println(girdi);
```

DEĞİŞKENLER

`girdi = "bilgisayar"`

ÇIKTI

```
bir metin giriniz
bilgisayar ve programlamaya giriş
```

next() ve nextLine(): Adım Adım

Girdi tamponu

b i l g i s a y a r _ v e _ p r o g r a m l a m a y a _ g i r i ş ↵

▲
imleç

Okunan kelime yazdırılır. İmleç boşluğun önünde bekler.

Adım 3 / 7

```
1 Scanner klavye = new Scanner(System.in);  
2 System.out.println("bir metin giriniz");  
3 String girdi = klavye.next();  
4 ► System.out.println(girdi);  
5 girdi = klavye.next();  
6 System.out.println(girdi);  
7 girdi = klavye.nextLine();  
8 System.out.println(girdi);
```

DEĞİŞKENLER

girdi = "bilgisayar"

ÇIKTI

```
bir metin giriniz  
bilgisayar ve programlamaya giriş  
bilgisayar
```

next() ve nextLine(): Adım Adım

Girdi tamponu

b i l g i s a y a r _ v e _ p r o g r a m l a m a y a _ g i r i ş ↵

▲
imleç

next() önce boşluğu **atlar**, sonra **"ve"** kelimesini okur.

Adım 4 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.println("bir metin giriniz");
3 String girdi = klavye.next();
4 System.out.println(girdi);
5 ► girdi = klavye.next();
6 System.out.println(girdi);
7 girdi = klavye.nextLine();
8 System.out.println(girdi);
```

DEĞİŞKENLER

girdi = "ve"

ÇIKTI

```
bir metin giriniz
bilgisayar ve programlamaya giriş
bilgisayar
```

next() ve nextLine(): Adım Adım

Girdi tamponu

b i l g i s a y a r _ v e _ p r o g r a m l a m a y a _ g i r i ş ↵

▲
imleç

Yazdırılır. İmleç yine bir boşluğun önünde.

Adım 5 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.println("bir metin giriniz");
3 String girdi = klavye.next();
4 System.out.println(girdi);
5 girdi = klavye.next();
6 ▶ System.out.println(girdi);
7 girdi = klavye.nextLine();
8 System.out.println(girdi);
```

DEĞİŞKENLER

girdi = "ve"

ÇIKTI

```
bir metin giriniz
bilgisayar ve programlamaya giriş
bilgisayar
ve
```

next() ve nextLine(): Adım Adım

Girdi tamponu

b i l g i s a y a r _ v e _ p r o g r a m l a m a y a _ g i r i ş ↵

▲
imleç

`nextLine()` boşluğu **atlamaz**: satır sonuna kadar okur, ↵'yi tüketir: " **programlamaya giriş**".

Adım 6 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.println("bir metin giriniz");
3 String girdi = klavye.next();
4 System.out.println(girdi);
5 girdi = klavye.next();
6 System.out.println(girdi);
7 ► girdi = klavye.nextLine();
8 System.out.println(girdi);
```

DEĞİŞKENLER

`girdi = " programlamaya giriş"`

ÇIKTI

```
bir metin giriniz
bilgisayar ve programlamaya giriş
bilgisayar
ve
```

next() ve nextLine(): Adım Adım

Girdi tamponu

b i l g i s a y a r _ v e _ p r o g r a m l a m a y a _ g i r i ş ↵

▲
imleç

Baştaki boşlukla birlikte yazdırılır. Tampon boşaldı.

Adım 7 / 7

```
1 Scanner klavye = new Scanner(System.in);
2 System.out.println("bir metin giriniz");
3 String girdi = klavye.next();
4 System.out.println(girdi);
5 girdi = klavye.next();
6 System.out.println(girdi);
7 girdi = klavye.nextLine();
▶ System.out.println(girdi);
```

DEĞİŞKENLER

girdi = " programlamaya giriş"

ÇIKTI

```
bir metin giriniz
bilgisayar ve programlamaya giriş
bilgisayar
ve
programlamaya giriş
```

KULLANICI GİRİŞİ

nextInt() Sonrası nextLine() Tuzağı

Hatalı: ad hiç sorulmadan geçilir.

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

KONSOL (GİRDİ: 19 ↵ AYŞE ↵)

```
Yaş: 19
Ad: [] 19
```

Doğru: artakalan satır sonunu tüket.

```
int yas = klavye.nextInt();
klavye.nextLine();           // ↵'yi at
System.out.print("Ad: ");
String ad = klavye.nextLine();
```

KONSOL

```
Yaş: 19
Ad: Ayşe
[Ayşe] 19
```

nextInt() sonrası tampon

1	9	↵	A	y	ş	e	↵
---	---	---	---	---	---	---	---

imleç ↵'nin önünde

`nextInt()` yalnızca `19` 'u alır. Hemen ardından gelen `nextLine()` , `↵` 'ye kadar olan **boş** metni okur. Alternatif: her şeyi `nextLine()` ile okuyup `Integer.parseInt` ile çevirmek.

KARŞILAŞTIRMA

== ile Karşılaştırma

- `==` operatörü nesnelerin **referanslarını** (bellekteki adreslerini) karşılaştırır, **içeriklerini değil**.
- İki değişken **aynı nesneyi** gösteriyorsa `true`, içerik aynı olsa bile farklı nesneler ise `false` döner.
- `str1 == str2` 'nin `true` çıkması bir şans değil, **dizgi havuzunun** sonucudur.

Kural: dizgilerin içeriğini karşılaştırırken **asla** `==` kullanmayın; `equals()` kullanın.

```
String str1 = "Giresun";
String str2 = "Giresun";
String str3 = new String("Giresun");
System.out.println(str1 == str2);           // true
System.out.println(str1 == "Giresun");      // true
System.out.println(str1 == str3);           // false
System.out.println(str3 == "Giresun");      // false
```

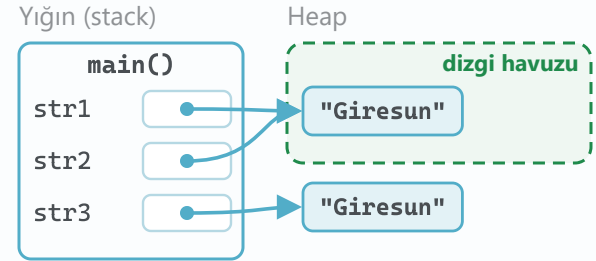
ÇIKTI

```
true
true
false
false
```

KARŞILAŞTIRMA

Dizgi Havuzu (String Pool)

- Kaynak koddaki `"giresun"` gibi sabit ifadeler **dizgi sabitleridir** (literal).
- Dizgi sabitleri heap'teki özel bir bölgede, **dizgi havuzunda** saklanır: içeriği aynı sabitler **aynı nesneyi paylaşır**. Bellek kullanımı azalır, performans artar (dizgiler değiştirilemediği için güvenlidir).
- `new String(...)` havuza tabi değildir: her `new` **yeni bir nesne** demektir.
- Bu yüzden `==`, içeriği aynı **sabitlerde** `true`, içeriği aynı `new` nesnelerinde `false` döner.
- Çalışma zamanında `+` ile üretilen dizgi de yeni nesnedir; derleme zamanında hesaplanan `"Gire" + "sun"` ise havuza gider. `s.intern()` havuzdaki eşini verir.

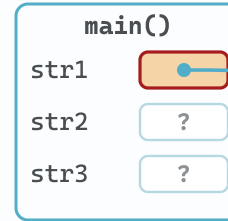


İki dizgi sabiti tek nesneyi paylaşır;
`new` ayrı bir nesne üretir.

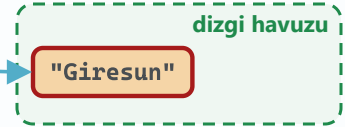
Dizgi Havuzu: Adım Adım

```
► String str1 = "Giresun";  
2 String str2 = "Giresun";  
3 String str3 = new String("Giresun");  
4 System.out.println(str1 == str2);  
5 System.out.println(str1 == "Giresun");  
6 System.out.println(str1 == str3);  
7 System.out.println(str3 == "Giresun");
```

Yığın (stack)



Heap



Havuzda **"Giresun"** sabiti için bir nesne var;
str1 onu gösterir.

Adım 1 / 7

DEĞİŞKENLER

str1 = →#1

str2 = ?

str3 = ?

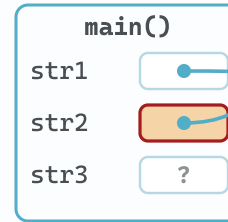
ÇIKTI



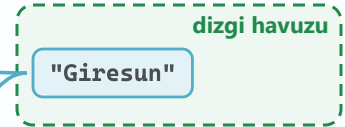
Dizgi Havuzu: Adım Adım

```
1 String str1 = "Giresun";  
▶ String str2 = "Giresun";  
3 String str3 = new String("Giresun");  
4 System.out.println(str1 == str2);  
5 System.out.println(str1 == "Giresun");  
6 System.out.println(str1 == str3);  
7 System.out.println(str3 == "Giresun");
```

Yığın (stack)



Heap



Aynı sabit tekrar kullanılır; yeni nesne **oluşmaz**,
str2 de aynı nesneyi gösterir.

Adım 2 / 7

DEĞİŞKENLER

str1 = →#1

str2 = →#1

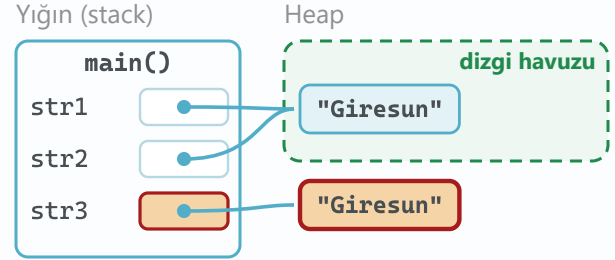
str3 = ?

ÇIKTI



Dizgi Havuzu: Adım Adım

```
1 String str1 = "Giresun";  
2 String str2 = "Giresun";  
▶ String str3 = new String("Giresun");  
4 System.out.println(str1 == str2);  
5 System.out.println(str1 == "Giresun");  
6 System.out.println(str1 == str3);  
7 System.out.println(str3 == "Giresun");
```



new her zaman havuz dışında **yeni** bir nesne oluşturur (içerik aynı).

Adım 3 / 7

DEĞİŞKENLER

str1 = →#1

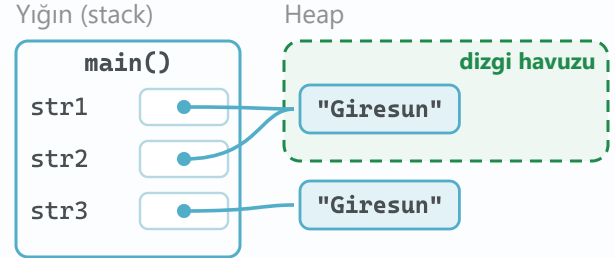
str2 = →#1

str3 = →#2

ÇIKTI

Dizgi Havuzu: Adım Adım

```
1 String str1 = "Giresun";  
2 String str2 = "Giresun";  
3 String str3 = new String("Giresun");  
4 ► System.out.println(str1 == str2);  
5 System.out.println(str1 == "Giresun");  
6 System.out.println(str1 == str3);  
7 System.out.println(str3 == "Giresun");
```



str1 ve str2 aynı nesne (#1) → **true**.

Adım 4 / 7

DEĞİŞKENLER

str1 = →#1

str2 = →#1

str3 = →#2

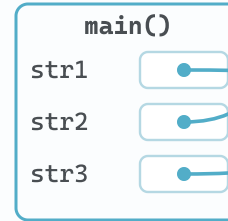
ÇIKTI

true

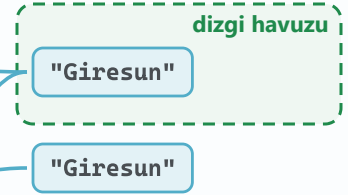
Dizgi Havuzu: Adım Adım

```
1 String str1 = "Giresun";  
2 String str2 = "Giresun";  
3 String str3 = new String("Giresun");  
4 System.out.println(str1 == str2);  
▶ System.out.println(str1 == "Giresun");  
6 System.out.println(str1 == str3);  
7 System.out.println(str3 == "Giresun");
```

Yığın (stack)



Heap



"Giresun" sabiti de havuzdaki #1 → **true**.

Adım 5 / 7

DEĞİŞKENLER

str1 = →#1

str2 = →#1

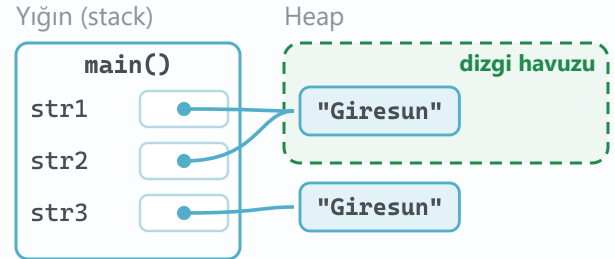
str3 = →#2

ÇIKTI

```
true  
true
```

Dizgi Havuzu: Adım Adım

```
1 String str1 = "Giresun";  
2 String str2 = "Giresun";  
3 String str3 = new String("Giresun");  
4 System.out.println(str1 == str2);  
5 System.out.println(str1 == "Giresun");  
▶ System.out.println(str1 == str3);  
7 System.out.println(str3 == "Giresun");
```



#1 ile #2 farklı nesneler (içerik aynı olsa da) → **false**.

Adım 6 / 7

DEĞİŞKENLER

str1 = →#1

str2 = →#1

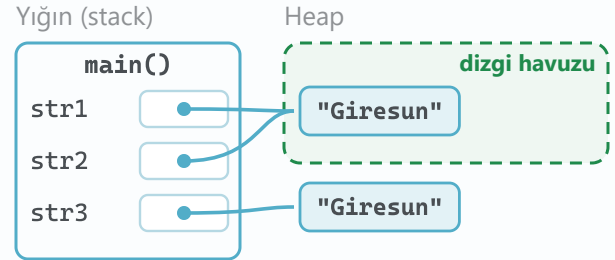
str3 = →#2

ÇIKTI

```
true  
true  
false
```

Dizgi Havuzu: Adım Adım

```
1 String str1 = "Giresun";  
2 String str2 = "Giresun";  
3 String str3 = new String("Giresun");  
4 System.out.println(str1 == str2);  
5 System.out.println(str1 == "Giresun");  
6 System.out.println(str1 == str3);  
▶ System.out.println(str3 == "Giresun");
```



#2 ile havuzdaki #1 farklı → **false**.

Adım 7 / 7

DEĞİŞKENLER

str1 = →#1

str2 = →#1

str3 = →#2

ÇIKTI

```
true  
true  
false  
false
```

KARŞILAŞTIRMA

equals() ile İçerik Karşılaştırma

```
String str1 = "Giresun";
String str2 = "Giresun";
String str3 = new String("Giresun");
System.out.println(str1.equals(str2));
System.out.println(str1.equals("Giresun"));
System.out.println(str1.equals(str3));
System.out.println(str3.equals("Giresun"));
System.out.println(str3.intern() == str1);
String a = "Gire";
String b = a + "sun"; // yeni nesne
System.out.println(b == str1);
System.out.println(b.equals(str1));
```

ÇIKTI

```
true
true
true
true
true
false
true
```

- `equals()` içeriği karakter karakter karşılaştırır: ilk dördü `true`.
- Nesnenin nasıl oluştuğu (dizgi sabiti, `new`, birleştirme, `Scanner`) sonucu etkilemez.
- Kullanıcı girdisi çalışma zamanında oluşur: `girdi == "evet"` neredeyse hep `false`.

```
String cevap = null;
System.out.println("evet".equals(cevap)); // false
System.out.println(cevap.equals("evet")); // hata!
```

```
Exception in thread "main"
java.lang.NullPointerException:
    Cannot invoke "String.equals(Object)" because
    "cevap" is null
```

Sabiti öne yazmak

(`"evet".equals(cevap)`) `null` değerine karşı güvenlidir.

KARŞILAŞTIRMA

Karşılaştırma Metotları

Metot	Döndürdüğü	Açıklama
<code>equals(str)</code>	<code>boolean</code>	İçerik birebir aynı mı?
<code>equalsIgnoreCase(str)</code>	<code>boolean</code>	Büyük/küçük harf farkını göz ardı ederek aynı mı?
<code>startsWith(str)</code>	<code>boolean</code>	Dizgi verilen dizgi ile başlıyor mu?
<code>endsWith(str)</code>	<code>boolean</code>	Dizgi verilen dizgi ile bitiyor mu?
<code>contains(str)</code>	<code>boolean</code>	Verilen dizgi, dizginin içinde geçiyor mu?
<code>isEmpty()</code> / <code>isBlank()</code>	<code>boolean</code>	Uzunluk 0 mı? / Yalnızca boşluk mu? (<code>isBlank</code> : Java 11+)
<code>compareTo(str)</code>	<code>int</code>	Sözlük sırası: negatif (önce), 0 (eşit), pozitif (sonra).
<code>compareToIgnoreCase(str)</code>	<code>int</code>	Harf büyüklüğünü göz ardı eden <code>compareTo</code> .

Hepsi büyük/küçük harfe **duyarlıdır** (adında *IgnoreCase* olanlar hariç) ve `null` parametreyle `equals` / `equalsIgnoreCase` yalnızca `false` döndürür.

Karşılaştırma Metotları: Örnek

```
String s = "Giresun Üni.";
System.out.println(s.startsWith("Gir"));           // true
System.out.println(s.startsWith("Gır"));           // false
System.out.println(s.endsWith("Üni."));           // true
System.out.println(s.endsWith("Üni"));            // false
System.out.println(s.equalsIgnoreCase("Giresun Üni.")); // true
System.out.println(s.equalsIgnoreCase("Gİresun üni.")); // true
System.out.println(s.equalsIgnoreCase("Gİresun Uni.")); // false
System.out.println(s.contains("Üni"));            // true
```

- Yorumlardaki sonuçlar programın gerçek çıktısıdır.
- "Gır" : **ı** ile **i** farklı karakterlerdir → **false**.
- "Üni" : dizgi noktayla bitiyor → **false**.
- **equalsIgnoreCase** karakterleri büyük harfe çevirerek karşılaştırır: **ı** ve **i** 'nin ikisi de **I** olur → "Gİresun üni." eşit sayılır.
- "Gİresun Uni." : **U** ile **Ü** farklı harflerdir → **false**.



compareTo: Sözlük Sırası

```
System.out.println("elma".compareTo("armut"));
System.out.println("armut".compareTo("elma"));
System.out.println("elma".compareTo("elma"));
System.out.println("Elma".compareTo("elma"));
System.out.println("Elma".compareToIgnoreCase("elma"));
System.out.println("ev".compareTo("evet"));
System.out.println("çay".compareTo("dut"));
```

ÇIKTI

```
4
-4
0
-32
0
-2
131
```

- `a.compareTo(b)` : `a` önce gelirse **negatif**, eşitse **0**, sonra gelirse **pozitif**.
- İlk farklı karakterin Unicode kodları çıkarılır:
`'e' - 'a' = 101 - 97 = 4`.
- `'E' - 'e' = 69 - 101 = -32`: büyük harfler küçüklerden önce gelir.
- Biri diğerinin öneki ise uzunluk farkı döner:
`"ev" vs "evet" → 2 - 4 = -2`.
- `'ç' (231) > 'd' (100)`: `compareTo` Türkçe alfabe sırasını **bilmez**, "çay" "dut"tan sonra gelir. Doğru Türkçe sıralama için `java.text.Collator` kullanılır.
- Sıralama (ör. `Arrays.sort`) ve `if (a.compareTo(b) < 0)` gibi kıyaslamalarda kullanılır. `<`, `>` operatörleri dizgilerle çalışmaz.

compareTo: İlk Farklı Karaktere Kadar

```
► String a = "masa", b = "mavi";  
2 int r1 = a.compareTo(b);  
3 int r2 = "kar".compareTo("karne");  
4 int r3 = "Zeki".compareTo("ali");  
5 System.out.println(r1 + " " + r2 + " " + r3);
```

masa · mavi · kar · karne · Zeki · ali

`compareTo` iki dizgiyi baştan başlayarak karakter karakter karşılaştırır.

Adım 1 / 9

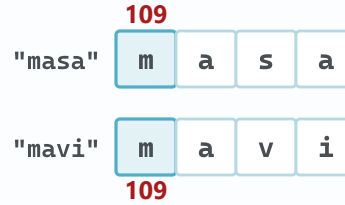
DEĞİŞKENLER

—

ÇIKTI

compareTo: İlk Farklı Karaktere Kadar

```
1 String a = "masa", b = "mavi";  
▶ 2 int r1 = a.compareTo(b);  
3 int r2 = "kar".compareTo("karne");  
4 int r3 = "Zeki".compareTo("ali");  
5 System.out.println(r1 + " " + r2 + " " + r3);
```



k = 0: 'm' (109) ile 'm' (109) eşit, devam.

Adım 2 / 9

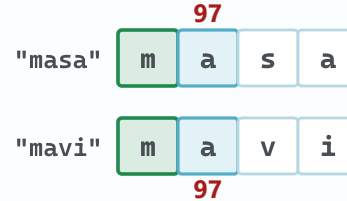
DEĞİŞKENLER

k = 0

ÇIKTI

compareTo: İlk Farklı Karaktere Kadar

```
1 String a = "masa", b = "mavi";  
▶ 2 int r1 = a.compareTo(b);  
3 int r2 = "kar".compareTo("karne");  
4 int r3 = "Zeki".compareTo("ali");  
5 System.out.println(r1 + " " + r2 + " " + r3);
```



k = 1: 'a' ile 'a' eşit.

Adım 3 / 9

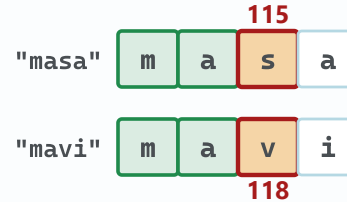
DEĞİŞKENLER

k = 1

ÇIKTI

compareTo: İlk Farklı Karaktere Kadar

```
1 String a = "masa", b = "mavi";  
▶ 2 int r1 = a.compareTo(b);  
3 int r2 = "kar".compareTo("karne");  
4 int r3 = "Zeki".compareTo("ali");  
5 System.out.println(r1 + " " + r2 + " " + r3);
```



k = 2: 's' (115) ile 'v' (118) farklı: karşılaştırma burada biter.

Adım 4 / 9

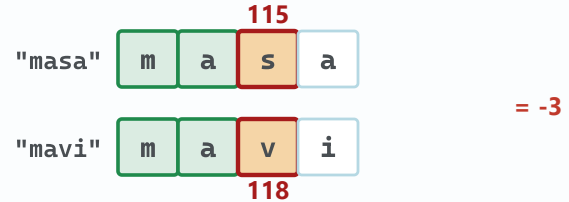
DEĞİŞKENLER

k = 2

ÇIKTI

compareTo: İlk Farklı Karaktere Kadar

```
1 String a = "masa", b = "mavi";  
▶ 2 int r1 = a.compareTo(b);  
3 int r2 = "kar".compareTo("karne");  
4 int r3 = "Zeki".compareTo("ali");  
5 System.out.println(r1 + " " + r2 + " " + r3);
```



Sonuç kodların farkı: $115 - 118 = -3$. Negatif: "masa" önce gelir.

Adım 5 / 9

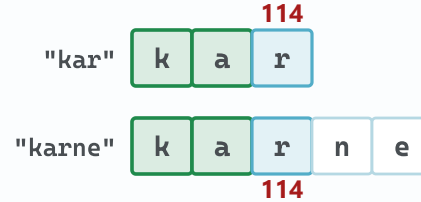
DEĞİŞKENLER

`r1 = -3`

ÇIKTI

compareTo: İlk Farklı Karaktere Kadar

```
1 String a = "masa", b = "mavi";  
2 int r1 = a.compareTo(b);  
▶ 3 int r2 = "kar".compareTo("karne");  
4 int r3 = "Zeki".compareTo("ali");  
5 System.out.println(r1 + " " + r2 + " " + r3);
```



"kar" ile "karne": ilk 3 karakter eşit, kısa olanın karakterleri bitti.

Adım 6 / 9

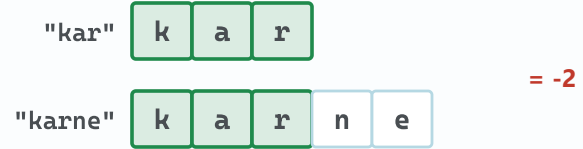
DEĞİŞKENLER

r1 = -3

ÇIKTI

compareTo: İlk Farklı Karaktere Kadar

```
1 String a = "masa", b = "mavi";  
2 int r1 = a.compareTo(b);  
▶ 3 int r2 = "kar".compareTo("karne");  
4 int r3 = "Zeki".compareTo("ali");  
5 System.out.println(r1 + " " + r2 + " " + r3);
```



Biri diğerinin öneki: uzunluk farkı döner, $3 - 5 = -2$.

Adım 7 / 9

DEĞİŞKENLER

r1 = -3

r2 = -2

ÇIKTI

compareTo: İlk Farklı Karaktere Kadar

```
1 String a = "masa", b = "mavi";  
2 int r1 = a.compareTo(b);  
3 int r2 = "kar".compareTo("karne");  
▶ int r3 = "Zeki".compareTo("ali");  
5 System.out.println(r1 + " " + r2 + " " + r3);
```



İlk karakterler farklı: 'Z' (90) – 'a' (97) = **-7**. Büyük harfler küçüklerden önce gelir: Z bile a'dan önce!

Adım 8 / 9

DEĞİŞKENLER

r1 = -3

r2 = -2

r3 = -7

ÇIKTI

```
-3 -2 -7
```

compareTo: İlk Farklı Karaktere Kadar

```
1 String a = "masa", b = "mavi";  
2 int r1 = a.compareTo(b);  
3 int r2 = "kar".compareTo("karne");  
4 int r3 = "Zeki".compareTo("ali");  
► System.out.println(r1 + " " + r2 + " " + r3);
```

-3 -2 -7

-3 -2 -7 yazılır.

Adım 9 / 9

DEĞİŞKENLER

r1 = -3

r2 = -2

r3 = -7

ÇIKTI

-3 -2 -7

printf Metodu

- Kullanım: `System.out.printf("biçim dizgisi", parametreler);`
- Biçim dizgisi, parametrelerin yerleşeceği **yer tutucuları** (format specifier) içerir; parametreler sırayla yerleşir.
- `printf` alt satıra **geçmez**: sona `\n` ya da platformdan bağımsız `%n` eklenir.
- Genişlik: `%Wd` → W karakter, **sağa** hizalı; `%-Wd` → W karakter, **sola** hizalı.
- Hassasiyet: `%.2f` → virgülden sonra 2 basamak (yuvarlanır); `%8.2f` → toplam 8 karakter.
- Değer genişliğe sığmazsa kesilmez, alan genişler.

Yer tutucu	Tip	Örnek → çıktı
<code>%d</code>	Tam sayı	42 → 42
<code>%f</code>	Ondalık (<code>float</code> , <code>double</code>)	<code>%.1f</code> : 3.14159 → 3,1
<code>%s</code>	Dizgi	<code>%5s</code> : "ab" → ab
<code>%c</code>	Karakter	<code>'Z'</code> → Z
<code>%b</code>	Mantıksal	true → true
<code>%n</code>	Yeni satır	—

Ondalık ayırıcı **yerel ayara** bağlıdır: tr-TR'de

3,1 , `Locale.US` ile 3.1 .

%	-	8	.2	f
---	---	---	----	---

başlangıç sola yasla genişlik hassasiyet tip

BIÇIMLI ÇIKTI

printf: Örnek

```
int i = 3;  
float f = -17;  
String s = "java";
```

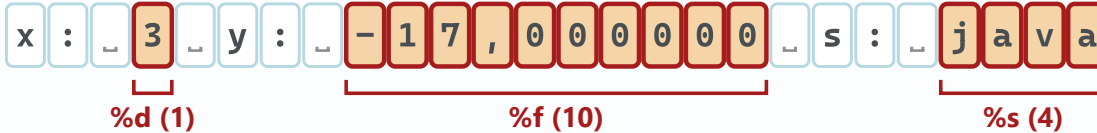
```
System.out.printf("x: %d y: %f s: %s\n", i, f, s);  
System.out.printf("x: %4d y: %5.2f s: %6s\n", i, f, s);  
System.out.printf("x: %-4d y: %-7.2f s: %-6s\n", i, f, s);  
System.out.printf("x: %-4d y: %-6.3f s: %-6s\n", i, f, s);
```

ÇIKTI (TR-TR; SATIR SONUNDAKİ BOŞLUKLAR · İLE GÖSTERİLDİ)

```
x: 3 y: -17,000000 s: java  
x:   3 y: -17,00 s:   java  
x: 3   y: -17,00 s: java..  
x: 3   y: -17,000 s: java..
```

- `%f` varsayılan olarak 6 ondalık basamak yazar: `-17,000000` . Türkçe ayarda ondalık ayırıcı **virgüldür**.
- `%5.2f` : `-17,00` zaten 6 karakter olduğu için 5 genişlik etkisizdir. `%-6.3f` : `-17,000` 7 karakterdir, alan genişler.
- `%6s` → `__java` (sağa hizalı); `%-6s` → `java__` (sola hizalı, sona boşluk eklenir).

Genişlik ve Hassasiyet: Karakter Karakter



Genişlik verilmezse her değer kendi uzunluğu kadar yer kaplar; `%f` 6 ondalık basamak yazar.

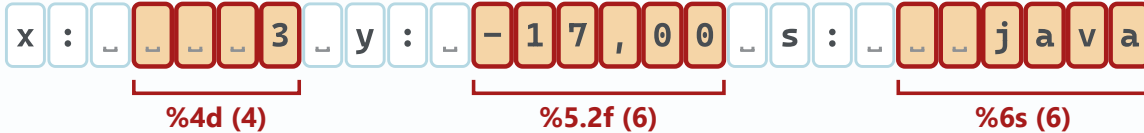
Adım 1 / 4

```
1 int i = 3;
2 float f = -17;
3 String s = "java";
4 System.out.printf("x: %d y: %f s: %s\n", i, f, s);
5 System.out.printf("x: %4d y: %5.2f s: %6s\n", i, f, s);
6 System.out.printf("x: %-4d y: %-7.2f s: %-6s\n", i, f, s);
7 System.out.printf("x: %-4d y: %-6.3f s: %-6s\n", i, f, s);
```

ÇIKTI

```
x: 3 y: -17,000000 s: java
```

Genişlik ve Hassasiyet: Karakter Karakter



%4d ve %6s değeri **sağa** yaslar, solu boşlukla doldurur. `-17,00` 6 karakter: 5 genişliğe sığmaz, alan büyür.

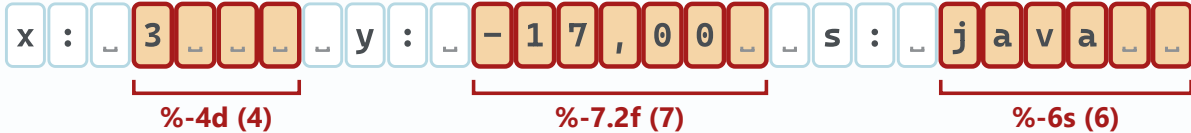
Adım 2 / 4

```
1 int i = 3;
2 float f = -17;
3 String s = "java";
4 System.out.printf("x: %d y: %f s: %s\n", i, f, s);
5 ► System.out.printf("x: %4d y: %5.2f s: %6s\n", i, f, s);
6 System.out.printf("x: %-4d y: %-7.2f s: %-6s\n", i, f, s);
7 System.out.printf("x: %-4d y: %-6.3f s: %-6s\n", i, f, s);
```

ÇIKTI

```
x: 3 y: -17,000000 s: java
x: 3 y: -17,00 s: java
```

Genişlik ve Hassasiyet: Karakter Karakter



Eksi işareti **sola** hizalar: boşluklar değerin sağına eklenir. Sütunlar alt alta düzgün hizalanır.

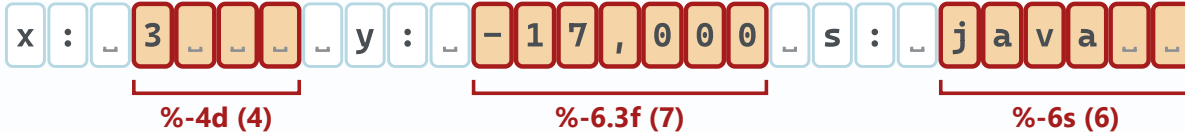
Adım 3 / 4

```
1 int i = 3;
2 float f = -17;
3 String s = "java";
4 System.out.printf("x: %d y: %f s: %s\n", i, f, s);
5 System.out.printf("x: %4d y: %5.2f s: %6s\n", i, f, s);
6 ▶ System.out.printf("x: %-4d y: %-7.2f s: %-6s\n", i, f, s);
7 System.out.printf("x: %-4d y: %-6.3f s: %-6s\n", i, f, s);
```

ÇIKTI

```
x: 3 y: -17,000000 s: java
x:   3 y: -17,00 s:   java
x: 3   y: -17,00 s: java
```

Genişlik ve Hassasiyet: Karakter Karakter



%-6.3f : 3 ondalık → **-17,000** 7 karakter; genişlik 6'dan büyük olduğu için dolgu yapılmaz.

Adım 4 / 4

```
1 int i = 3;
2 float f = -17;
3 String s = "java";
4 System.out.printf("x: %d y: %f s: %s\n", i, f, s);
5 System.out.printf("x: %4d y: %5.2f s: %6s\n", i, f, s);
6 System.out.printf("x: %-4d y: %-7.2f s: %-6s\n", i, f, s);
▶ System.out.printf("x: %-4d y: %-6.3f s: %-6s\n", i, f, s);
```

ÇIKTI

```
x: 3 y: -17,000000 s: java
x:   3 y: -17,00 s:   java
x: 3   y: -17,00 s: java
x: 3   y: -17,000 s: java
```

Diğer Biçimler ve String.format

```
System.out.printf("|%5s|%-5s|%05d|%.1f|%", "ab", "ab", 42, 3.14159);  
System.out.printf("%c %b %e %x %%%", 'Z', true, 1234.5, 255);  
String mesaj = String.format("%s: %d puan", "Ayşe", 85);  
System.out.println(mesaj);  
System.out.printf("%,d%", 1234567);  
System.out.printf(Locale.US, "%.2f%", 2.5); // import java.util.Locale;  
System.out.println("%s = %d".formatted("x", 7)); // Java 15+
```

ÇIKTI

```
|  ab|ab  |00042|3,1|  
Z true 1,234500e+03 ff %  
Ayşe: 85 puan  
1.234.567  
2.50  
x = 7
```

- `%05d` başa sıfır doldurur, `%,d` binlik ayırıcı koyar (tr-TR: nokta), `%e` bilimsel, `%x` onaltılık gösterim, `%%` yüzde işaretidir.
- `String.format(...)` aynı kuralları kullanır ama ekrana yazmak yerine biçimlenmiş **dizgiyi döndürür**.
- İlk parametre bir `Locale` olabilir: `Locale.US` ile ondalık ayırıcı noktadır.
- Tip uyumsuzluğu çalışma zamanında hatadır: `printf("%d", 3.5)` → `IllegalFormatConversionException`.

Palindrom Kontrolü: Adım Adım

```
► String metin = "Kayak";  
2 metin = metin.toLowerCase();  
3 String ters = "";  
4 for (int i = metin.length() - 1; i >= 0; i--) {  
5     ters += metin.charAt(i);  
6 }  
7 System.out.println(ters);  
8 System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin

K	a	y	a	k
0	1	2	3	4

ters

"" (boş)

Kullanıcının girdiği metin: "Kayak" .

Adım 1 / 11

DEĞİŞKENLER

metin = "Kayak"

ÇIKTI



Palindrom Kontrolü: Adım Adım

```
1 String metin = "Kayak";  
2 metin = metin.toLowerCase();  
3 String ters = "";  
4 for (int i = metin.length() - 1; i >= 0; i--) {  
5     ters += metin.charAt(i);  
6 }  
7 System.out.println(ters);  
8 System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin

k	a	y	a	k
0	1	2	3	4

ters

"" (boş)

`toLowerCase()` ile küçük harfe çevrilir (yeni nesne `metin` 'e atanır).

Adım 2 / 11

DEĞİŞKENLER

`metin = "kayak"`

ÇIKTI



Palindrom Kontrolü: Adım Adım

```
1 String metin = "Kayak";
2 metin = metin.toLowerCase();
3
4 ▶ String ters = "";
5 for (int i = metin.length() - 1; i >= 0; i--) {
6     ters += metin.charAt(i);
7 }
8 System.out.println(ters);
9 System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin

k	a	y	a	k
0	1	2	3	4

ters

"" (boş)

Sonuç için boş bir dizgi hazırlanır.

Adım 3 / 11

DEĞİŞKENLER

metin = "kayak"

ters = ""

ÇIKTI

Palindrom Kontrolü: Adım Adım

```
1 String metin = "Kayak";
2 metin = metin.toLowerCase();
3 String ters = "";
4 for (int i = metin.length() - 1; i >= 0; i--) {
5     ters += metin.charAt(i);
6 }
7 System.out.println(ters);
8 System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin

k	a	y	a	k
0	1	2	3	4

ters

k

`i = 4` : `charAt(4)` = 'k' sona eklenir → **"k"**.

Adım 4 / 11

DEĞİŞKENLER

`metin = "kayak"`

`ters = "k"`

`i = 4`

ÇIKTI

Palindrom Kontrolü: Adım Adım

```
1 String metin = "Kayak";
2 metin = metin.toLowerCase();
3 String ters = "";
4 for (int i = metin.length() - 1; i >= 0; i--) {
5     ters += metin.charAt(i);
6 }
7 System.out.println(ters);
8 System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin



ters



`i = 3` : `charAt(3)` = 'a' sona eklenir → **"ka"**.

Adım 5 / 11

DEĞİŞKENLER

`metin = "kayak"`

`ters = "ka"`

`i = 3`

ÇIKTI



Palindrom Kontrolü: Adım Adım

```
1 String metin = "Kayak";
2 metin = metin.toLowerCase();
3 String ters = "";
4 for (int i = metin.length() - 1; i >= 0; i--) {
5     ters += metin.charAt(i);
6 }
7 System.out.println(ters);
8 System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin

k	a	y	a	k
0	1	2	3	4

ters

k	a	y
---	---	---

`i = 2` : `charAt(2)` = 'y' sona eklenir → "kay".

Adım 6 / 11

DEĞİŞKENLER

metin = "kayak"

ters = "kay"

i = 2

ÇIKTI

```
System.out.println(metin.equals(ters));
```

Palindrom Kontrolü: Adım Adım

```
1 String metin = "Kayak";
2 metin = metin.toLowerCase();
3 String ters = "";
4 for (int i = metin.length() - 1; i >= 0; i--) {
5     ters += metin.charAt(i);
6 }
7 System.out.println(ters);
8 System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin

k	a	y	a	k
0	1	2	3	4

ters

k	a	y	a
---	---	---	---

`i = 1` : `charAt(1)` = 'a' sona eklenir → "kaya".

Adım 7 / 11

DEĞİŞKENLER

`metin = "kayak"`

`ters = "kaya"`

`i = 1`

ÇIKTI

Palindrom Kontrolü: Adım Adım

```
1 String metin = "Kayak";
2 metin = metin.toLowerCase();
3 String ters = "";
4 for (int i = metin.length() - 1; i >= 0; i--) {
5     ters += metin.charAt(i);
6 }
7 System.out.println(ters);
8 System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin

k	a	y	a	k
0	1	2	3	4

ters

k	a	y	a	k
---	---	---	---	---

`i = 0` : `charAt(0)` = 'k' sona eklenir → "kayak".

Adım 8 / 11

DEĞİŞKENLER

`metin = "kayak"`

`ters = "kayak"`

`i = 0`

ÇIKTI

Palindrom Kontrolü: Adım Adım

```
1 String metin = "Kayak";
2 metin = metin.toLowerCase();
3 String ters = "";
4 ▶ for (int i = metin.length() - 1; i >= 0; i--) {
5     ters += metin.charAt(i);
6 }
7 System.out.println(ters);
8 System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin

k	a	y	a	k
0	1	2	3	4

ters

k	a	y	a	k
---	---	---	---	---

`i = -1` : `-1 >= 0` yanlış, döngü biter.

Adım 9 / 11

DEĞİŞKENLER

`metin = "kayak"`

`ters = "kayak"`

`i = -1`

ÇIKTI

Palindrom Kontrolü: Adım Adım

```
1 String metin = "Kayak";
2 metin = metin.toLowerCase();
3 String ters = "";
4 for (int i = metin.length() - 1; i >= 0; i--) {
5     ters += metin.charAt(i);
6 }
7 System.out.println(ters);
8 System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin

k	a	y	a	k
0	1	2	3	4

ters

k	a	y	a	k
---	---	---	---	---

Ters çevrilmiş metin yazdırılır.

Adım 10 / 11

DEĞİŞKENLER

metin = "kayak"

ters = "kayak"

ÇIKTI

kayak

Palindrom Kontrolü: Adım Adım

```
1 String metin = "Kayak";
2 metin = metin.toLowerCase();
3 String ters = "";
4 for (int i = metin.length() - 1; i >= 0; i--) {
5     ters += metin.charAt(i);
6 }
7 System.out.println(ters);
▶ System.out.println(metin.equals(ters));
```

Palindrom: tersten okunuşu aynı olan metin (kayak, ana, kek). Cümlelerde önce harf olmayanlar silinir:

`metin.replaceAll("[^\\p{L}]", "")` (yalnızca harfler kalır).

metin

k	a	y	a	k
0	1	2	3	4

ters

k	a	y	a	k
---	---	---	---	---

İçerikler aynı: `equals` → **true**, "kayak" bir palindromdur.

Adım 11 / 11

DEĞİŞKENLER

metin = "kayak"

ters = "kayak"

ÇIKTI

kayak
true

UYGULAMALAR

Uygulama: Kelime Frekansı

```
String metin = "Bir iki bir üç İki bir";
String[] kelimeler = metin.toLowerCase().split(" ");
boolean[] sayildi = new boolean[kelimeler.length];
for (int i = 0; i < kelimeler.length; i++) {
    if (sayildi[i]) {
        continue; // bu kelime daha önce sayıldı
    }
    int frekans = 1;
    for (int j = i + 1; j < kelimeler.length; j++) {
        if (kelimeler[i].equals(kelimeler[j])) {
            frekans++;
            sayildi[j] = true;
        }
    }
    System.out.println(kelimeler[i] + ": " + frekans);
}
```

ÇIKTI

```
bir: 3
iki: 2
üç: 1
```

- `toLowerCase()` sayesinde "Bir/bir" ve "İki/iki" aynı kelime sayılır (tr-TR'de `İ` → `i`).
- `split(" ")` metni kelimelere böler.
- Kelimeler `equals` ile karşılaştırılır: `split` 'in ürettiği parçalar ayrı nesnelerdir, `==` yanlış sonuç verir.
- `sayildi` dizisi, aynı kelimenin ikinci kez raporlanmasını engeller.

0	1	2	3	4	5
bir	iki	bir	üç	iki	bir
false	false	true	false	false	true

i = 0 sonrası: "bir" 3 kez, kopyaları işaretlendi

Kelime Frekansı: Bayraklarla Sayma

```
1 String metin = "elma Armut elma kiraz ARMUT";  
2 ▶ String[] k = metin.toLowerCase().split(" ");  
3 boolean[] sayildi = new boolean[k.length];  
4 for (int i = 0; i < k.length; i++) {  
5     if (sayildi[i]) continue;  
6     int frekans = 1;  
7     for (int j = i + 1; j < k.length; j++) {  
8         if (k[i].equals(k[j])) {  
9             frekans++;  
10            sayildi[j] = true;  
11        }  
12    }  
13    System.out.println(k[i] + ": " + frekans);  
14 }
```

	0	1	2	3	4
k	elma	armut	elma	kiraz	armut

Küçük harfe çevrilip boşluklardan bölünür: 5 kelimelik dizi. "Armut" ve "ARMUT" artık aynı.

Adım 1 / 9

DEĞİŞKENLER

—

ÇIKTI

Kelime Frekansı: Bayraklarla Sayma

```
1 String metin = "elma Armut elma kiraz ARMUT";
2 String[] k = metin.toLowerCase().split(" ");
▶ boolean[] sayildi = new boolean[k.length];
4 for (int i = 0; i < k.length; i++) {
5     if (sayildi[i]) continue;
6     int frekans = 1;
7     for (int j = i + 1; j < k.length; j++) {
8         if (k[i].equals(k[j])) {
9             frekans++;
10            sayildi[j] = true;
11        }
12    }
13    System.out.println(k[i] + ": " + frekans);
14 }
```

	0	1	2	3	4
k	elma	armut	elma	kiraz	armut
sayildi	false	false	false	false	false

sayildi
sayildi : her kelime için bir bayrak, hepsi false .

Adım 2 / 9

DEĞİŞKENLER

—

ÇIKTI

Kelime Frekansı: Bayraklarla Sayma

```
1 String metin = "elma Armut elma kiraz ARMUT";
2 String[] k = metin.toLowerCase().split(" ");
3 boolean[] sayildi = new boolean[k.length];
4 for (int i = 0; i < k.length; i++) {
5     if (sayildi[i]) continue;
6     int frekans = 1;
7     for (int j = i + 1; j < k.length; j++) {
8         if (k[i].equals(k[j])) {
9             frekans++;
10            sayildi[j] = true;
11        }
12    }
13    System.out.println(k[i] + ": " + frekans);
14 }
```

	0	1	2	3	4
k	elma	armut	elma	kiraz	armut
sayildi	false	false	true	false	false

sayildi
i = 0 (elma): iç döngü sağdaki kelimeleri `equals` ile tarar; j = 2 eşit → frekans 2, `sayildi[2] = true`.

Adım 3 / 9

DEĞİŞKENLER

`i = 0``frekans = 2`

ÇIKTI

Kelime Frekansı: Bayraklarla Sayma

```
1 String metin = "elma Armut elma kiraz ARMUT";
2 String[] k = metin.toLowerCase().split(" ");
3 boolean[] sayildi = new boolean[k.length];
4 for (int i = 0; i < k.length; i++) {
5     if (sayildi[i]) continue;
6     int frekans = 1;
7     for (int j = i + 1; j < k.length; j++) {
8         if (k[i].equals(k[j])) {
9             frekans++;
10            sayildi[j] = true;
11        }
12    }
13    System.out.println(k[i] + ": " + frekans);
14 }
```

	0	1	2	3	4
k	elma	armut	elma	kiraz	armut
sayildi	false	false	true	false	false

sayildi
elma: 2 yazılır.

Adım 4 / 9

DEĞİŞKENLER

i = 0

frekans = 2

ÇIKTI

elma: 2

Kelime Frekansı: Bayraklarla Sayma

```
1 String metin = "elma Armut elma kiraz ARMUT";
2 String[] k = metin.toLowerCase().split(" ");
3 boolean[] sayildi = new boolean[k.length];
4 for (int i = 0; i < k.length; i++) {
5     if (sayildi[i]) continue;
6     int frekans = 1;
7     for (int j = i + 1; j < k.length; j++) {
8         if (k[i].equals(k[j])) {
9             frekans++;
10            sayildi[j] = true;
11        }
12    }
13    System.out.println(k[i] + ": " + frekans);
14 }
```

	0	1	2	3	4
k	elma	armut	elma	kiraz	armut
sayildi	false	false	true	false	true

sayildi
i = 1 (armut): j = 4 eşit → frekans 2, sayildi[4] = true .

Adım 5 / 9

DEĞİŞKENLER

i = 1 frekans = 2

ÇIKTI

elma: 2

Kelime Frekansı: Bayraklarla Sayma

```
1 String metin = "elma Armut elma kiraz ARMUT";
2 String[] k = metin.toLowerCase().split(" ");
3 boolean[] sayildi = new boolean[k.length];
4 for (int i = 0; i < k.length; i++) {
5     if (sayildi[i]) continue;
6     int frekans = 1;
7     for (int j = i + 1; j < k.length; j++) {
8         if (k[i].equals(k[j])) {
9             frekans++;
10            sayildi[j] = true;
11        }
12    }
13    System.out.println(k[i] + ": " + frekans);
14 }
```

	0	1	2	3	4
k	elma	armut	elma	kiraz	armut
sayildi	false	false	true	false	true

armut: 2 yazılır.

Adım 6 / 9

DEĞİŞKENLER

i = 1

frekans = 2

ÇIKTI

```
elma: 2
armut: 2
```

Kelime Frekansı: Bayraklarla Sayma

```
1 String metin = "elma Armut elma kiraz ARMUT";
2 String[] k = metin.toLowerCase().split(" ");
3 boolean[] sayildi = new boolean[k.length];
4 for (int i = 0; i < k.length; i++) {
5     if (sayildi[i]) continue;
6     int frekans = 1;
7     for (int j = i + 1; j < k.length; j++) {
8         if (k[i].equals(k[j])) {
9             frekans++;
10            sayildi[j] = true;
11        }
12    }
13    System.out.println(k[i] + ": " + frekans);
14 }
```

	0	1	2	3	4
k	elma	armut	elma	kiraz	armut
sayildi	false	false	true	false	true

sayildi

i = 2: bayrak true, bu elma zaten sayıldı →
`continue`

Adım 7 / 9

DEĞİŞKENLER

i = 2

ÇIKTI

```
elma: 2
armut: 2
```

Kelime Frekansı: Bayraklarla Sayma

```
1 String metin = "elma Armut elma kiraz ARMUT";
2 String[] k = metin.toLowerCase().split(" ");
3 boolean[] sayildi = new boolean[k.length];
4 for (int i = 0; i < k.length; i++) {
5     if (sayildi[i]) continue;
6     int frekans = 1;
7     for (int j = i + 1; j < k.length; j++) {
8         if (k[i].equals(k[j])) {
9             frekans++;
10            sayildi[j] = true;
11        }
12    }
13    System.out.println(k[i] + ": " + frekans);
14 }
```

	0	1	2	3	4
k	elma	armut	elma	kiraz	armut
sayildi	false	false	true	false	true

sayildi
i = 3 (kiraz): eşi yok, frekans 1.

Adım 8 / 9

DEĞİŞKENLER

i = 3

frekans = 1

ÇIKTI

```
elma: 2
armut: 2
kiraz: 1
```

Kelime Frekansı: Bayraklarla Sayma

```
1 String metin = "elma Armut elma kiraz ARMUT";
2 String[] k = metin.toLowerCase().split(" ");
3 boolean[] sayildi = new boolean[k.length];
4 for (int i = 0; i < k.length; i++) {
5     if (sayildi[i]) continue;
6     int frekans = 1;
7     for (int j = i + 1; j < k.length; j++) {
8         if (k[i].equals(k[j])) {
9             frekans++;
10            sayildi[j] = true;
11        }
12    }
13    System.out.println(k[i] + ": " + frekans);
14 }
```

	0	1	2	3	4
k	elma	armut	elma	kiraz	armut
sayildi	false	false	true	false	true

sayildi
i = 4: zaten sayıldı, atlanır. Döngü biter.

Adım 9 / 9

DEĞİŞKENLER

i = 4

ÇIKTI

```
elma: 2
armut: 2
kiraz: 1
```

ÖZET

Özet

Kavram	Özü
<code>String</code>	Karakter dizisini saklayan, değiştirilemez nesne; dizgi sabiti (<code>"..."</code>) ya da <code>new</code> ile oluşur.
İndis	<code>0 ... length() - 1</code> ; <code>charAt(i)</code> bir <code>char</code> döndürür.
<code>+</code>	Soldan sağa: bir yan dizgiyse birleştirme; sayıları toplamak için parantez.
Metotlar	<code>indexOf</code> (<code>-1</code> = yok), <code>substring(a, b)</code> (b hariç), <code>toUpperCase</code> , <code>replace</code> , <code>split</code> , <code>trim</code> ... hepsi yeni değer döndürür.
<code>StringBuilder</code>	Döngüde çok birleştirme için değiştirilebilir tampon.
<code>Scanner</code>	<code>next()</code> kelime, <code>nextLine()</code> satır; <code>nextInt()</code> sonrası ↵ tamponda kalır.
<code>==</code> / <code>equals</code>	Aynı nesne mi? / Aynı içerik mi? Dizgiler için daima <code>equals</code> .
Dizgi havuzu	Aynı dizgi sabitleri tek nesneyi paylaşır; <code>new</code> ve çalışma zamanı birleştirmesi yeni nesne üretir.
Karşılaştırma	<code>equalsIgnoreCase</code> , <code>startsWith</code> , <code>endsWith</code> , <code>contains</code> , <code>compareTo</code> (Unicode sırası).
<code>printf</code>	<code>%d %f %s</code> ; <code>%ld</code> sağa, <code>%-ld</code> sola hizalı; <code>%.2f</code> hassasiyet; ayırıcılar yerel ayara bağlı.

Sık Yapılan Hatalar (1): Derleyici Hataları

Dizgiyi tek tırnakla yazmak

error: unclosed character literal

`String s = 'Merhaba';` Tek tırnak yalnızca tek karakter içindir; dizgi çift tırnakla yazılır.

char'a dizgi atamak

error: incompatible types: String cannot be converted to char

`char c = "a";` yerine `char c = 'a';` ya da `"a".charAt(0)`.

length'i parantezsiz yazmak

error: cannot find symbol

symbol: variable length

Dizgide `s.length()` metottur. Terside hatadır: dizide `d.length()` → symbol: method length().

string küçük harfle

error: cannot find symbol

symbol: class string

Sınıf adı `String` 'dir; Java büyük/küçük harfe duyarlıdır.

Dizgiyi sayıya doğrudan atamak

error: incompatible types: String cannot be converted to int

`int n = "42";` yerine `Integer.parseInt("42")`.

Karakteri yerinde değiştirmeye çalışmak

error: unexpected type
required: variable

`s.charAt(0) = 'j';` yazılamaz: dizgi değiştirilemez. Yeni dizgi oluşturun ya da `StringBuilder.setCharAt` kullanın.

Sık Yapılan Hatalar (2): Mantık ve Çalışma Zamanı

İçeriği == ile karşılaştırmak

```
if (girdi == "evet") // derlenir ama yanlış
```

Kullanıcı girdisi yeni bir nesnedir; koşul `false` çıkar.
`"evet".equals(girdi)` yazın.

Sonucu atamayı unutmak

```
s.toUpperCase(); // s değişmez
```

Dizgi değiştirilemez; `s = s.toUpperCase();` yazılmalıdır.

Son indisi `length()` sanmak

```
StringIndexOutOfBoundsException: Index 4 out of bounds for length 4
```

Son karakter `s.charAt(s.length() - 1)`; döngü koşulu `i < s.length()`.

`nextInt` sonrası `nextLine`

```
String ad = klavye.nextLine(); // ""
```

Artakalan ↵ okunur. Araya bir `klavye.nextLine();` ekleyin.

null dizgide metot çağırmak

```
NullPointerException: Cannot invoke "String.equals(Object)"
```

Sabiti öne yazın: `"evet".equals(cevap)`.

`substring`'in bitişini dahil sanmak

```
s.substring(5, 12) // 7 karakter
```

Bitiş indisi **hariçtir**: `uzunluk = bitiş - başlangıç`.

ÖZET

Örnek Problemler (1)

Problem 1. Ne yazdırır?

```
String s = "Karadeniz";  
s.toUpperCase();  
String t = s.substring(2, 5);  
System.out.println(t + " " + t.length());  
System.out.println(s.indexOf("de"));  
System.out.println(s.indexOf("x"));  
System.out.println(1 + 2 + "3" + 4 + 5);
```

Problem 2. Ne yazdırır?

```
String a = "ders";  
String b = "de" + "rs";  
String c = new String("ders");  
String d = c;  
System.out.println(a == b);  
System.out.println(a == c);  
System.out.println(c == d);  
System.out.println(a.equals(c));
```

► Cevapları göster

ÖZET

Örnek Problemler (2)

Problem 3. Çıktıyı karakter karakter yazın (tr-TR).

```
String ad = "Ali";  
double not1 = 72.456;  
System.out.printf("[%5s][%3d][%7.2f]%n", ad, 7, not1);
```

Problem 4. Ne yazdırır?

```
public static String bas(String ad, String soyad) {  
    return ad.charAt(0) + "." + soyad.charAt(0) + ".";  
}  
System.out.println(bas("Sercan", "Külcü"));  
String e = "ogrenci@giresun.edu.tr";  
int at = e.indexOf('@');  
System.out.println(e.substring(0, at));  
System.out.println(e.substring(at + 1));
```

► Cevapları göster

ÖZET

Alıştırmalar (1)

1. `tersCevir(String s)` metodunu yazın: `tersCevir("Giresun")` → `"nuseriG"`.
2. `say(String s, char aranan)` : karakterin kaç kez geçtiğini döndürsün. `say("Karadeniz", 'a')` → ?
3. `basHarfBuyuk(String kelime)` : ilk harf büyük, gerisi küçük. `basHarfBuyuk("jAVA")` → `"Java"`.

► Cevapları göster

ÖZET

Alıştırmalar (2)

4. "Java ile nesne yönelimli programlama" cümlesinde kaç kelime vardır? Tek satırda bulun.
5. Bir kelimedeki sesli harfleri sayan programı yazın (Türkçe sesliler dahil). "Ordu" → ?
6. Kullanıcıdan önce yaş, sonra ad-soyad isteyen ve `printf` ile "Ad: ..., Yaş: ..." yazan programı `nextInt` tuzağına düşmeden yazın.

► Cevapları göster

Kaynaklar

- Oracle, *The Java Tutorials*: "Strings", "Comparing Strings and Portions of Strings", "The StringBuilder Class", "Formatting Numeric Print Output".
- Oracle, *Java SE 21 API Documentation*: `java.lang.String`, `java.lang.StringBuilder`, `java.util.Scanner`, `java.util.Formatter`, `java.util.Locale`.
- J. Gosling ve diğ., *The Java Language Specification, Java SE 21 Edition* (§3.10.5 String Literals, §15.18.1 String Concatenation Operator +).
- P. Deitel ve H. Deitel, *Java: How to Program*, 11. baskı, Pearson, 2017 (Bölüm 14: Strings, Characters and Regular Expressions).
- C. S. Horstmann, *Core Java, Volume I: Fundamentals*, 12. baskı, 2022 (Bölüm 3.6: Strings).
- A. B. Downey ve C. Mayfield, *Think Java*, 2. baskı, O'Reilly, 2019 (Bölüm 6: Loops and Strings).

Son

Bölüm 5: Dizgi (String)

Sonraki bölüm: Bölüm 6: Hata Ayıklama ve İstisnalar · [Terim sözlüğü](#)