Bölüm 5: Dizgi

JAVA ile Nesne Yönelimli Programlama

Dizgi (String)

- Karakter dizilerini saklayan bir nesnedir.
- *new* kullanılmadan oluşturulabilir.
- Unicode standardı kullanılır, Türkçe karakterleri destekler.

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

Dizgi (String)

- *String*, değiştirilemez (*immutable*) bir nesnedir.
- Metin işleme sırasında yeni bir *String* nesnesi oluşturulur.
- + operatörü, *String* birleştirme işlemi için kullanılır.
- Eşitlik kontrolü için == işleci yerine, *equals()* metodu kullanılmalıdır.

İndeksleme

- İlk karakterin indeksi: *0*
- Son karakterin indeksi: *name.length() - 1*
- Her bir karakter, *char* türünde bir değerdir.
- *String*'in uzunluğunu aşan bir indeks kullanılmamalıdır.

Örnek Metotlar

- **indexOf(str):** alt dizginin başlangıç indeksini, bulunamazsa -1 döndürür.
- **length():** dizgideki karakter sayısını döndürür.
- **substring(index1, index2):** İki indeks arasındaki alt dizgiyi döndürür.
- **substring(index1):** indeksten sona kadar olan alt dizgiyi döndürür.
- **toLowerCase():** karakterleri küçük harfe çevrilmiş yeni dizgi döndürür.
- **toUpperCase():** karakterleri büyük harfe çevrilmiş yeni dizgi döndürür.

Örnek Metotlar

```
String str = "Java Programlama";
int indeks = str.indexOf("Programlama"); // 5
int uzunluk = str.length(); // 16
String alt = str.substring(5, 12); // Program
String kucuk = str.toLowerCase(); // java programlama
String buyuk = str.toUpperCase(); // JAVA PROGRAMLAMA
```

- *substring()* ve *toLowerCase()*,
  - dizgide değişiklik yapmadan yeni oluşturduğu dizgiyi döndürür.

Değiştirilemez (immutable)

- *String* nesneleri değiştirilemez.
- Orijinal dizgiyi değiştirmek yerine, yeni bir kopya oluşturur.
- Güvenli ve tutarlı kod yazımını sağlar.
- *String pooling* ve performans avantajları sağlar.

```
String s = "Giresun Üniversitesi";
s.toUpperCase();
System.out.println(s); // Giresun Üniversitesi
s = s.toUpperCase();
System.out.println(s); // GIRESUN ÜNIVERSITESI
```

Kullanıcı Girişi

- *Scanner* sınıfı, kullanıcıdan girdi almak için kullanılır.
- *next()* metodu, bir kelimeyi, *nextLine()* metodu, bir satırı *String* olarak okur.

```
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
```

Karşılaştırma

- == işleci, nesnelerin referanslarını karşılaştırır, içeriklerini değil.

```
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
```

Dizgi Havuzu (String Pooling)

- İçeriği aynı dizgi deyimlerinin, aynı bellek bölgesini paylaşmasını sağlar.
- Bellek kullanımını iyileştirir, performansı arttırır.
- Dizgi deyimleri (literal) özel bir bellek ekranında saklanır.
- *new* ile oluşturulan dizgiler, dizgi havuzuna tabi değildir.
- "giresun" gibi sabit ifadeler, dizgi deyimleridir.
- == işleci, referansları karşılaştırır.
  - içeriği aynı dizgi deyimlerinde *true* döner.
  - içeriği aynı *new* ile oluşturulan dizgilerde *false* döner.

Karşılaştırma

- *equals()* metodu, nesnelerin içeriğini karşılaştırmak için kullanılır.

```
String str1 = "Giresun";
String str2 = "Giresun";
String str3 = new String("Giresun");

System.out.println(str1.equals(str2)); // true
System.out.println(str1.equals("Giresun")); // true
System.out.println(str1.equals(str3)); // true
System.out.println(str3.equals("Giresun")); // true
```

Karşılaştırma Metotları

- **equalsIgnoreCase(str):** büyük/küçük harf göz ardı ederek karşılaştırır.
- **startsWith(str):** dizginin bir dizgi ile başlayıp başlamadığını kontrol eder.
- **endsWith(str):** dizginin bir dizgi ile bitip bitmediğini kontrol eder.
- **contains(str):** verilen dizginin içinde bulunup bulunmadığını kontrol eder.

Karşılaştırma Metotları

```
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("Gıresun üni."));//true
System.out.printl(s.equalsIgnoreCase("GIresun Uni."));//false
System.out.println(s.contains("Üni")); // true
```

printf Metodu

- System.out.printf("biçim dizisi", parametreler); şeklinde kullanılır.
- Biçim dizisi, parametreleri eklemek için yer tutucular içerir:
  - %d: Tamsayı (int)
  - %f: Ondalık sayı (float)
  - %s: Dizgi (string)
- printf bir alt satıra geçmez, \n eklemek gerekir.
- Genişlik kontrolü
  - %Wd: genişliği W karakter, sağa hizalı
  - %-Wd: genişliği W karakter, sola hizalı

printf Metodu

```
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);
```

```
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
```

SON