

String İşlemleri

String Nedir?

- **Karakter dizisinden** oluşan nesnelerdir
- String nesneler salt okunur (**read-only**) özelliktedir
- Bunlara **immutable** nesneler de denir
- String nesne üzerinde değişiklik yapan metotların hepsi **yeni bir String nesne** üretmektedir
- **Orjinal String** nesne kesinlikle değişikliğe uğratılmaz

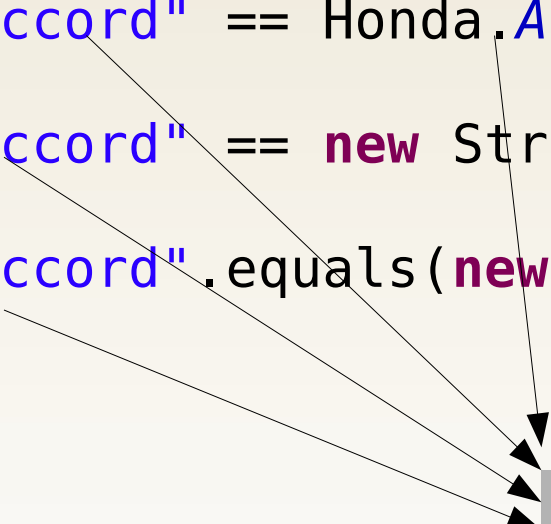
String Havuzu ve Değer Eşitliği

```
System.out.println("Accord" == Honda.ACCORD);           True
System.out.println("Accord" == new String("Accord"));    False
System.out.println("Accord".equals(new String("Accord"))); True
```

String sabitler hafızada sabit veri havuzunda tutulur

`==` nesne referanslarını kontrol eder

`String.equals()` değer eşitliğini kontrol eder



"Accord"	0
...	1
String("Accord")	2
...	3
String("Accord")	4
...	5
...	6

RAM

String intern() Metodu

- String sınıfının içerisinde **bir metottur**
- String nesnenin **değerini sabit havuzundan döndürür**
- Eğer değer halihazırda sabit havuzunda mevcut değil ise **değer önce havuza eklenir**, ardından havuzdaki referansı dönülür

```
System.out.println("Accord" == new String("Accord").intern()); True
```

String Concatenation

```
String model = "Honda " + "Accord";
```

```
model += " 2011";
```

```
System.out.println(model);
```

"Honda"	0
"Accord"	1
"Honda Accord"	2
"Honda Accord 2011"	3
...	4
...	5

```
model = model + " 2011";
```

Operator overloading
nümerik değerler dışında
sadece String nesnelerde
kullanılabilir

StringBuffer ve StringBuilder Sınıfları

- **Dinamik** olarak String nesne oluşturmayı sağlar
- **StringBuilder** Java 5 ile birlikte gelmiştir
- Daha öncesinde **StringBuffer** sınıfı kullanılmaktaydı
- Her ikisinin de yaptığı **temp String** oluşturmayı ortadan kaldırmaktır
- StringBuffer **eşzamanlı erişim kontrolü** yaptığı için kullanımı daha maliyetlidir

StringBuilder

```
StringBuilder builder = new StringBuilder(1024);
```

```
builder.append("Honda ");
```

```
builder.append("Accord ");
```

```
builder.append("2011");
```

```
String model = builder.toString();
```

Heap memory

...	0
builder	1
...	2
...	3
model	4
...	5

String Operasyonları

`"".isEmpty();`

`"abcdefabc".indexOf('c');` → 2
`"abcdefabc".indexOf("bc");`

`" abcdefabc ".trim();` → "abcdefabc"

`"abcdefabc".startsWith("abc");`
`"abcdefabc".endsWith("abc");`

`"abcdefabc".charAt(3);` → 'c'

`"abcdefabc".getBytes();` → byte[]

`"abcdefabc".substring(3, 6);` → "def"
`"abc:def:ghj".split(":");` → String[]{"abc", "def", "ghj"}

`String.valueOf(true);` → "true"
`String.valueOf(1);` → "1"
`String.valueOf(1.0f);` → "1.0"

`String.join(":", "abc", "def", "ghj");` → "abc:def:ghj"

İletişim



www.harezmi.com.tr

www.java-egitimleri.com



info@harezmi.com.tr

info@java-egitimleri.com



[@HarezmiBilisim](https://twitter.com/HarezmiBilisim)

[@JavaEgitimleri](https://twitter.com/JavaEgitimleri)