Strings

String Methods

Create and Concatenate a String

String is a **sequence of characters**, implementing CharSequence interface

```
String name = "John Wayne";

String name = new String("John Wayne");
```

Concatenation:

```
str1 + str2        or        str1.concat(str2)
```

1. If both operands are numeric, + means addition
2. If either operand is String, + means concatenation
3. Evaluation is left to right

```
System.out.println(3 + 8);                  11
System.out.println("John" + "Wayne");       JohnWayne
System.out.println("John" + 8);             John8
System.out.println("John" + 3 + 8);         John38
System.out.println("John" + (3 + 8));       John11


// it's allowed to concatenate with null string
System.out.println("John" + null);          Johnnull
System.out.println(null + "John");          nullJohn


// with assignment operators
String name = "John";
name += "Wayne";            // name = name + "Wayne"
System.out.println(name)                    JohnWayne
```

```
// length ()

String name = "John";

System.out.println(name.length());

  => 4


// charAt()
               0    6
String name = "John Wayne";

System.out.println(name.charAt(6));

  => a


String name = "John Wayne";

System.out.println(name.charAt(12));

  => StringIndexOutOfBoundsException
```

```
// indexOf()
               0 23 5 7   11
String name = "Doctor Dolittle";

System.out.println(name.indexOf('t'));

  => 3

System.out.println(name.indexOf('t', 5));

  => 11

System.out.println(name.indexOf("cto"));

  => 2

System.out.println(name.indexOf("Do", 4));

  => 7

System.out.println(name.indexOf("A"));

  => -1   // not found
```

```
// substring()
                   0  3    8
String name = "John Wayne";

System.out.println(name.substring(3));

  => n Wayne  // from index 3 to the end

System.out.println(name.substring(3, 8));

  => n Way    // from index 3 to 8 (not included!)

System.out.println(name.substring(3, 3));

  =>          // empty string

System.out.println(name.substring(8, 3));

  => StringIndexOutOfBoundsException

System.out.println(name.substring(3, 14));

  => StringIndexOutOfBoundsException
```

```
// toLowerCase()

String name = "John Wayne";

System.out.println(name.toLowerCase());

  => john wayne


// toUpperCase()

String name = "John Wayne";

System.out.println(name.toUpperCase());

  => JOHN WAYNE
```

```
// equals(), equalsIgnoreCase()

String name1 = new String("John Wayne");

String name2 = new String("John Wayne");

String name3 = new String("john wayne");

System.out.println(name1 == name2);

  => false   // not referencing to the same object

System.out.println(name1.equals(name2));

  => true    // same content

System.out.println(name1.equals(name3));

  => false   // String is case-sensitive

System.out.println(name1.equalsIgnoreCase(name3));

  => true    // ignoring the case
```

```
// startsWith(), endsWith()

String name = "John Wayne";

System.out.println(name.startsWith("J"));

  => true

System.out.println(name.startsWith("Jo"));

  => true

System.out.println(name.endsWith("e"));

  => true

System.out.println(name.endsWith("Wayne"));

  => true

System.out.println(name.startsWith('J'));

  => DOES NOT COMPILE   // argument is a String, not char!!
```

```
// contains()

String name = "John Wayne";

System.out.println(name.contains("n"));

  => true

System.out.println(name.contains("John"));

  => true

System.out.println(name.contains("j"));

  => false
```

```
// replace()

String str = "abcdeabc";

System.out.println(str.replace('c', 'y'));

  => abydeaby   // replaces all instances of 'c' with 'y'

System.out.println(str.replace("c", "y"));

  => abydeaby   // parameters can be both String and char

System.out.println(str.replace("bcd", "xyz"));

  => axyzeabc
```

```
// strip(), trim(), stripLeading(), stripTrailing()

String str = "    abc  ";

System.out.println("|" + str.strip() + "|");

  => |abc|

System.out.println("|" + str.trim() + "|");

  => |abc|    // same as strip, but supports Unicode

System.out.println("|" + str.stripLeading() + "|");

  => |abc  |

System.out.println("|" + str.stripTrailing() + "|");

  => |    abc|


// whitespaces also includes \t (tab), \n (new line), \r (carriage return)

// all escape sequences count as one character in length
```

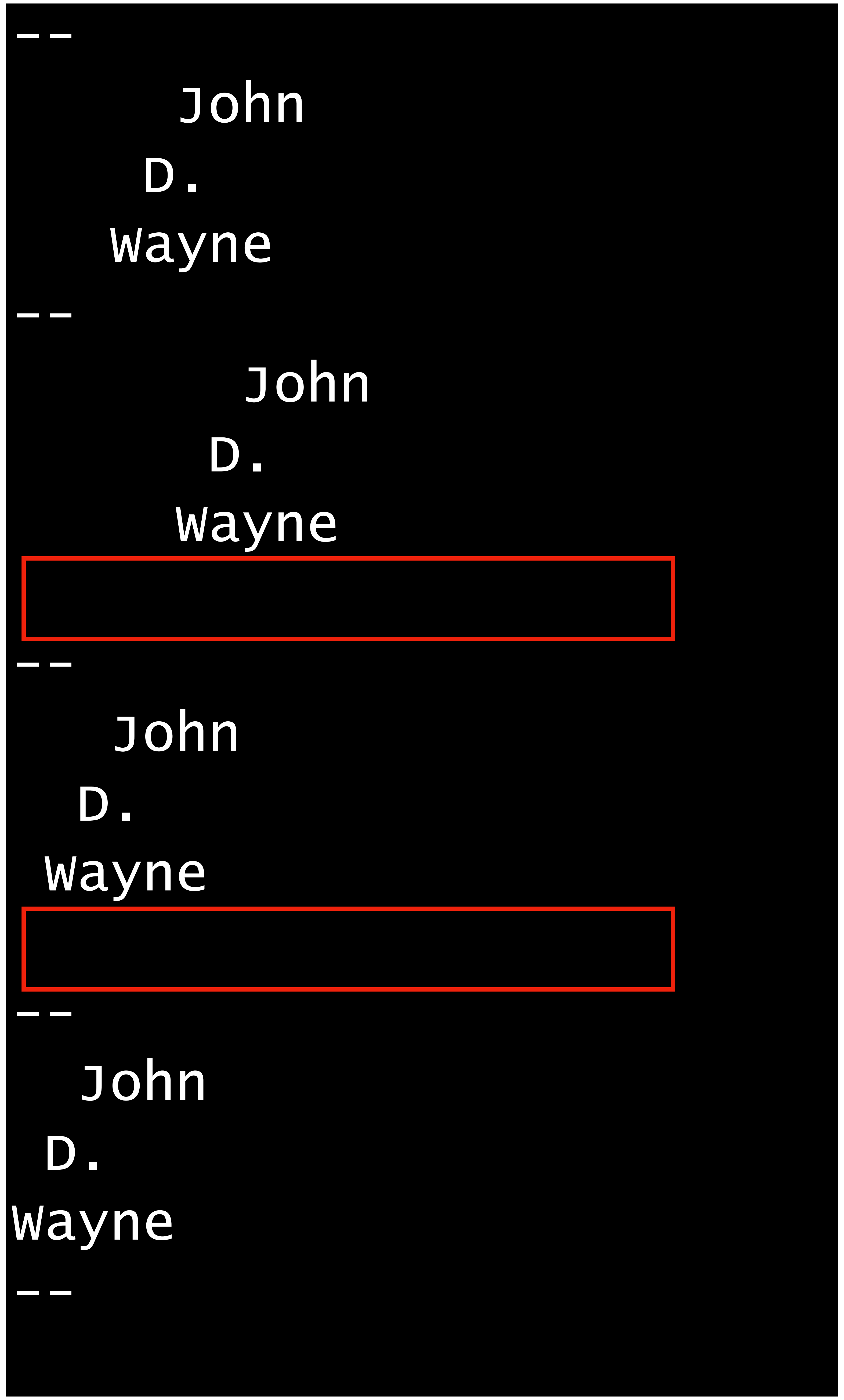
indent(n) method

- if n = 0 does nothing
- if n > 0 adds the same number of blank spaces to each line
- if n < 0 tries to remove n whitespace characters from the beginning of line
- normalizes existing line breaks
- adds line break at the end if missing

stripIndent() method

- removes all leading incidental whitespace
- normalizes existing line breaks
- **does not** add line break at the end if missing

```java
String str = "     John\n    D.\n   Wayne";
System.out.println("--");
System.out.println(str);
System.out.println("--");
System.out.println(str.indent(2));
System.out.println("--");
System.out.println(str.indent(-2));
System.out.println("--");
System.out.println(str.stripIndent());
System.out.println("--");
```



```
--
     John
    D.
   Wayne
--
       John
      D.
     Wayne

--
   John
  D.
 Wayne

--
  John
 D.
Wayne
--
```

```
// translateEscapes();

String name = "John\\tWayne";

System.out.println(name);

  => John\tWayne

System.out.println(name.translateEscapes());

  => John   Wayne
```

```
// isEmpty(), isBlank()

System.out.println("".isEmpty());

  => true

System.out.println("  ".isEmpty());

  => false

System.out.println("".isBlank());

  => true

System.out.println("  ".isBlank());

  => true
```

String formating symbols

`%s`	for any type, usualy for String
`%d`	for integral values (int and long)
`%f`	for decimal numbers (float and double)
`%n`	inserts a system-dependent line separator

```
// format(), formatted()

String name = "John";

int numberOfMarbles = 5;

String printOut1 = name + " has " + numberOfMarbles + " marbles.";

String printOut2 = String.format("%s has %d marbles.", name, numberOfMarbles);

String printOut3 = "%s has %d marbles.".formatted(name, numberOfMarbles);

System.out.println(printOut1);

System.out.println(printOut2);

System.out.println(printOut3);
```

```
John has 5 marbles.
John has 5 marbles.
John has 5 marbles.
```

```
// method chaining: left -> right

String name = "   John Wayne ";

System.out.println(name.trim().toUpperCase().replace('Y', 'R'));

  => JOHN WARNE
```

```
// Strings are immutable!
String name = "John Wayne";
name.toUpperCase();
System.out.println(name);
```

```
John Wayne
```

```
// you have to reassign the new value or create a new String
String name = "John Wayne";
name = name.toUpperCase();
System.out.println(name);
String name2 = name.toUpperCase();
System.out.println(name2);
```

```
JOHN WAYNE
JOHN WAYNE
```

String Methods