# String Formatting

https://dzone.com/articles/java-string-format-examples

Most common way of formatting a string in java is using *String.format()*. If there were a "java sprintf" then this would be it.

```
String output = String.format("%s = %d", "joe", 35);
```

For formatted console output, you can use *printf()* or the *format()* method of *System.out* and *System.err*PrintStreams.

```
System.out.printf("My name is: %s%n", "joe");
```

Create a *Formatter* and link it to a *StringBuilder*. Output formatted using the *format()* method will be appended to the *StringBuilder*.

```
StringBuilder sbuf = new StringBuilder();
Formatter fmt = new Formatter(sbuf);
fmt.format("PI = %f%n", Math.PI);
System.out.print(sbuf.toString());
// you can continue to append data to sbuf here.
```

# Format Specifiers

Here is a quick reference to all the conversion specifiers supported.

| SPECIFIER | APPLIES TO | OUTPUT |
|---|---|---|
| %a | floating point (except *BigDecimal*) | Hex output of floating point number |
| %b | Any type | “true” if non-null, “false” if null |
| %c | character | Unicode character |
| %d | integer (incl. byte, short, int, long, bigint) | Decimal Integer |
| %e | floating point | decimal number in scientific notation |
| %f | floating point | decimal number |
| %g | floating point | decimal number, possibly in scientific notation depending on the precision and value. |
| %h | any type | Hex String of value from hashCode() method. |
| %n | none | Platform-specific line separator. |
| %o | integer (incl. byte, short, int, long, bigint) | Octal number |
| %s | any type | String value |
| %t | Date/Time (incl. long, Calendar, Date and TemporalAccessor) | %t is the prefix for Date/Time conversions. More formatting flags are needed after this. See Date/Time conversion below. |
| %x | integer (incl. byte, short, int, long, bigint) | Hex string. |

# Date and Time Formatting

Note: Using the formatting characters with "%T" instead of "%t" in the table below makes the output uppercase.

| FLAG | NOTES |
|---|---|
| %tA | Full name of the day of the week, e.g. "`Sunday`", "`Monday`" |
| %ta | Abbreviated name of the week day e.g. "`Sun`", "`Mon`", etc. |
| %tB | Full name of the month e.g. "`January`", "`February`", etc. |
| %tb | Abbreviated month name e.g. "`Jan`", "`Feb`", etc. |
| %tC | Century part of year formatted with two digits e.g. "00" through "99". |
| %tc | Date and time formatted with "`%ta %tb %td %tT %tZ %tY`" e.g. "`Fri Feb 17 07:45:42 PST 2017`" |
| %tD | Date formatted as "`%tm/%td/%ty`" |
| %td | Day of the month formatted with two digits. e.g. "`01`" to "`31`". |
| %te | Day of the month formatted without a leading 0 e.g. "1" to "31". |
| %tF | ISO 8601 formatted date with "`%tY-%tm-%td`". |
| %tH | Hour of the day for the 24-hour clock e.g. "`00`" to "`23`". |
| %th | Same as %tb. |
| %tI | Hour of the day for the 12-hour clock e.g. "`01`" – "`12`". |
| %tj | Day of the year formatted with leading 0s e.g. "`001`" to "`366`". |
| %tk | Hour of the day for the 24 hour clock without a leading 0 e.g. "`0`" to "`23`". |
| %tl | Hour of the day for the 12-hour click without a leading 0 e.g. "`1`" to "`12`". |
| %tM | Minute within the hour formatted a leading 0 e.g. "`00`" to "`59`". |
| %tm | Month formatted with a leading 0 e.g. "`01`" to "`12`". |
| %tN | Nanosecond formatted with 9 digits and leading 0s e.g. "000000000" to |

| | |
|---|---|
| | “999999999”. |
| %tp | Locale specific “am” or “pm” marker. |
| %tQ | Milliseconds since epoch Jan 1 , 1970 00:00:00 UTC. |
| %tR | Time formatted as 24-hours e.g. “`%tH:%tM`“. |
| %tr | Time formatted as 12-hours e.g. “`%tI:%tM:%tS %Tp`“. |
| %tS | Seconds within the minute formatted with 2 digits e.g. “00” to “60”. “60” is required to support leap seconds. |
| %ts | Seconds since the epoch Jan 1, 1970 00:00:00 UTC. |
| %tT | Time formatted as 24-hours e.g. “`%tH:%tM:%tS`“. |
| %tY | Year formatted with 4 digits e.g. “`0000`” to “`9999`“. |
| %ty | Year formatted with 2 digits e.g. “`00`” to “`99`“. |
| %tZ | Time zone abbreviation. e.g. “`UTC`“, “`PST`“, etc. |
| %tz | Time Zone Offset from GMT e.g. “<br>`-0800`<br>“. |

# Argument Index

An argument index is specified as a number ending with a “`$`” after the “`%`” and selects the specified argument in the argument list.

```
String.format("%2$s", 32, "Hello"); // prints: "Hello"
```

# Formatting an Integer

With the `%d` format specifier, you can use an argument of all integral types including byte, short, int, long and BigInteger.

Default formatting:

```
String.format("%d", 93); // prints 93
```

Specifying a width:

```
String.format("|%20d|", 93); // prints: |                  93|
```

Left-justifying within the specified width:

```
String.format("|%-20d|", 93); // prints: |93                  |
```

Pad with zeros:

```
String.format("|%020d|", 93); // prints: |00000000000000000093|
```

Print positive numbers with a “+”:

*(Negative numbers always have the “-” included):*

```
String.format("|%+20d|', 93); // prints: |                 +93|
```

A space before positive numbers.

A “-” is included for negative numbers as per normal.

```
String.format("|% d|", 93); // prints: | 93| String.format("|% d|", -36); //
prints: |-36|
```

Use locale-specific thousands separator:

For the US locale, it is “,”:

```
String.format("|%,d|", 10000000); // prints: |10,000,000|
```

Enclose negative numbers within parentheses (“()”) and skip the "-":

```
String.format("|%(d|", -36); // prints: |(36)|
```

Octal output:

```
String.format("|%o|"), 93); // prints: 135
```

Hex output:

```
String.format("|%x|", 93); // prints: 5d
```

Alternate representation for octal and hex output:

Prints octal numbers with a leading "0" and hex numbers with leading "0x".

```
String.format("|%#o|", 93);
// prints: 0135
String.format("|%#x|", 93);
// prints: 0x5d
String.format("|%#X|", 93);
// prints: 0X5D
```

# String and Character Conversion

Default formatting:

Prints the whole string.

```
String.format("|%s|", "Hello World"); // prints: "Hello World"
```

Specify Field Length

```
String.format("|%30s|", "Hello World"); // prints: | Hello World|
```

Left Justify Text

```
String.format("|%-30s|", "Hello World"); // prints: |Hello World |
```

Specify Maximum Number of Characters

```
String.format("|%.5s|", "Hello World"); // prints: |Hello|
```

Field Width and Maximum Number of Characters

```
String.format("|%30.5s|", "Hello World"); | Hello|
```