Java Reference – Printf style formatting

By John Bentley

Quick Reference Examples

Basic Quick Reference Examples

```
// Common form
System.out.printf("Life expectancy for %s: %2.2f", "John", 78.7);
| "Life expectancy for John: 78.70"

// Case
System.out.printf("%s", "Apple");  |       "Apple" | Case - Pass through
System.out.printf("%S", "Apple");  |       "APPLE" | Case - Upper

// Line Separator
System.out.printf("The lazy dog%njumped over the log", );
| "The lazy dog
jumped over the log"

// Argument Order
System.out.printf("Given Name: %s; Family Name: %s;   ", "John", "Bentley");
| "Given Name: John; Family Name: Bentley;   " | Default Order

System.out.printf("Given Name: %s", "John", "Bentley");
| "Given Name: John" | Ignored argument

System.out.printf("Family Name: %2$s; Given Name: %1$s", "John", "Bentley");
| "Family Name: Bentley; Given Name: John" | Argument reorder

System.out.printf("%1$s %1$s", "John", "Bentley");  |   "John John"
| Repeated reference indexed

System.out.printf("%s %<s %s %<s", "John", "Bentley");  | "John John Bentley Bentley"
| Repeated reference pointed to

// Advanced (including multiple flags)
System.out.printf("$%2$- ,(15.2f", "Nice", -1234.23425);  | "$(1,234.23)     "
```

(Bentley, 2013) C:\Users\John\Documents\Sda\Code\Java\Examples\TutorialAtOracle\HelloWorld – PrintfStyleFormattingDemo.java

Format specifier conversion by category type, examples.

```
// Line Separator
System.out.printf("%s%n%s", "Hello", "There");
|
"Hello
There"

// General
     System.out.printf("%b", true);  |          "true" | Boolean
       System.out.printf("%h", 15);  |             "f" | Hexadecimal String
   System.out.printf("%s", "Cool");  |          "Cool" | String

// Character
      System.out.printf("%c", 'h');  |             "h" | Character
      System.out.printf("%c", 104);  |             "h" | Character
     System.out.printf("%c", 0x68);  |             "h" | Character
   System.out.printf("%c", 150370);  |            "瓢" | Character

// Numeric - Integral
       System.out.printf("%d", 15);  |            "15" | Decimal Integer
       System.out.printf("%o", 15);  |            "17" | Octal Integer
       System.out.printf("%x", 15);  |             "f" | Hexadecimal Integer

// Numeric - Floating point
  System.out.printf("%e", 1234.45);  |  "1.234450e+03" | Scientific Notation
  System.out.printf("%f", 1234.45);  |   "1234.450000" | Decimal Float
  System.out.printf("%g", 1234.45);  |       "1234.45" | Scientific Notation or Decimal
                                                        | Float
     System.out.printf("%a", 15.0);  |       "0x1.ep3" | Hexadecimal Float
```

```
// Date/Time
System.out.printf("%tF", Calendar.getInstance());  |    "2013-07-02" |
Date/Time (with ISO 8601 suffix)

// Date/Time Advanced
System.out.printf("%1$tF %1$tR", Calendar.getInstance());  | "2013-07-02 11:22" |

// Percentage literal
System.out.printf("%s%%", "Hello");  |        "Hello%" | Percentage literal
```

Basics

Various methods use Printf Style Formatting. The methods include:

- `System.out.printf`
- `System.err.printf`
- `String.format`
- `Formatter.format`

Printf Style Formatting is defined at `java.util.Formatter ...`

(Oracle, 2012) *http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Formatter.html*

Every method which uses `printf` style formatting requires a ***format string*** and an ***argument list***.

```
System.out.printf("Life expectancy for %s: %2.2f", "John", 78.7);

// Life expectancy for John: 78.70
```

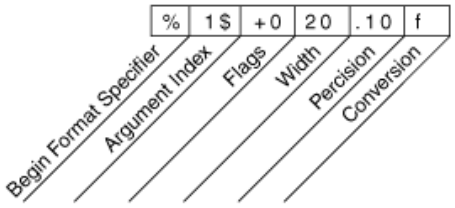
The format string contains ***fixed text*** and ***format specifiers***.

```
// Format string
"Life expectancy for %s: %2.2f"

// Fixed text
"Life expectancy for ", ": "

// Format specifiers
"%s", "%f"
```

Syntax

Syntax forms

Printf Style formatting converts several categories of type. Format Specifier syntax varies by type category:



Type Category	Syntax	Examples
[Line Separator]	%n	
General, Character, Numeric	%[argument_index$][flags][width][.precision]conversion	
Date/Time	%[argument_index$][flags][width]conversion	
[No argument]	%[flags][width]conversion	

(Oracle, 2012) http://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html

Conversion

Format Specifier conversion list by type category:

Type Category	Conversion	Result
[Line Separator]	'n'	The result is the platform-specific line separator
General	'b', 'B'	If the argument arg is null, then the result is "false". If arg is a boolean or Boolean, then the result is the string returned by String.valueOf(). Otherwise, the result is "true".
General	'h', 'H'	If the argument arg is null, then the result is "null". Otherwise, the result is obtained by invoking Integer.toHexString(arg.hashCode()).
General	's', 'S'	If the argument arg is null, then the result is "null". If arg implements Formattable, then arg.formatTo is invoked. Otherwise, the result is obtained by invoking arg.toString().
Character	'c', 'C'	Unicode character. Can handle characters, or Unicode code points in decimal or hexadecimal.
Numeric - Integral	'd'	Decimal integer

Numeric - Integral	'o'	Octal integer
Numeric - Integral	'x', 'X'	Hexadecimal integer
Numeric - Floating point	'e', 'E'	Decimal number in computerized scientific notation
Numeric - Floating point	'f'	Decimal (floating point) number
Numeric - Floating point	'g', 'G'	Computerized scientific notation or decimal format, depending on the precision and the value after rounding.
Numeric - Floating point	'a', 'A'	Hexadecimal floating-point number with a significand and an exponent
Date/time	't', 'T'	Prefix for date and time conversion characters. See Date/Time Conversions.
Percent	'%'	Literal '%' ('\u0025')

Format specifier conversion by category type, examples.

```
// Line Separator
System.out.printf("%s%n%s", "Hello", "There");
|
"Hello
There"


// General
     System.out.printf("%b", true);  |          "true" | Boolean
       System.out.printf("%h", 15);  |             "f" | Hexadecimal String
   System.out.printf("%s", "Cool");  |          "Cool" | String

// Character
      System.out.printf("%c", 'h');  |             "h" | Character
      System.out.printf("%c", 104);  |             "h" | Character
     System.out.printf("%c", 0x68);  |             "h" | Character
   System.out.printf("%c", 150370);  |            "瓢" | Character

// Numeric - Integral
       System.out.printf("%d", 15);  |            "15" | Decimal Integer
       System.out.printf("%o", 15);  |            "17" | Octal Integer
       System.out.printf("%x", 15);  |             "f" | Hexadecimal Integer

// Numeric - Floating point
  System.out.printf("%e", 1234.45);  |  "1.234450e+03" | Scientific Notation
  System.out.printf("%f", 1234.45);  |   "1234.450000" | Decimal Float
  System.out.printf("%g", 1234.45);  |       "1234.45" | Scientific Notation or Decimal
                                                        | Float
     System.out.printf("%a", 15.0);  |       "0x1.ep3" | Hexadecimal Float

// Date/Time
System.out.printf("%tF", Calendar.getInstance());  |    "2013-07-02" |
Date/Time (with ISO 8601 suffix)

// Date/Time Advanced
System.out.printf("%1$tF %1$tR", Calendar.getInstance());  | "2013-07-02 11:22" |

// Percentage literal
System.out.printf("%s%%", "Hello");  |        "Hello%" | Percentage literal
```

Numeric Conversions only apply to specific numeric categories, otherwise you get a compile error.

Integral - may be applied to Java integral types: byte, Byte, short, Short, int and Integer, long, Long, and BigInteger

Floating Point - may be applied to Java floating-point types: float, Float, double, Double, and BigDecimal

(Oracle, 2012) *http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Formatter.html*

Case

Conversion case determines output case. A lowercase conversion specifier passes the case through. An uppercase conversion specifier converts all the characters to uppercase.

```
System.out.printf("%s", "Apple"); // Apple
System.out.printf("%S", "Apple"); // APPLE
```

Line Separator

The optional *line_separator* separates lines.

```
System.out.printf("The lazy dog%njumped over the log");

// The lazy dog
// jumped over the log
```

Argument Order

The default order of the substitutions occurs in the order of the arguments presented

```
System.out.printf("Given Name: %s; Family Name: %s;", "John", "Bentley");
// "Given Name: John; Family Name: Bentley;"
```

The optional *argument_index ("explicit argument indices")* indicates the position of the argument in the argument list. The first argument is reference by "1$", the second "2$", etc.

You can use explicit argument indices to reorder the argument list substitutions.

```
System.out.printf("Family Name: %2$s; Given Name: %1$s", "John", "Bentley");
// Family Name: Bentley; Given Name: John
```

You can use explicit argument indices to repeatedly reference an argument.

```
// Date/Time Advanced
System.out.printf("%1$tF %1$tR", Calendar.getInstance());  | "2013-07-02 11:22" |
```

If no *argument_index* is supplied then the format specifiers apply to the arguments in the order presented. If there are more arguments than specifiers then the remaining excessive arguments are ignored.

```
System.out.printf("Given Name: %s;", "John", "Bentley");
// Given Name: John;
```

Another way to reference arguments by position is to use the '<', which causes the argument for the previous format specifier to be re-used.

```
System.out.printf("%s %<s %s %<s", "John", "Bentley");
| "John John Bentley Bentley" | Repeated reference pointed to
```

(Bentley, 2013) PrintfStyleFormattingDemo.Java based on
(Oracle, 2011), Class Formatter, http://docs.oracle.com/javase/6/docs/api/java/util/Formatter.html

Precision

Type Category		
General	Maximum characters to output (starting from the left)	
Numeric – Floating Point	'e', 'f': total digits after decimal separator. 'g': total digits after rounding 'a': forbidden [exception]	
Line Separator, Character, Numeric – Integral, Date/Time, Percentage Literal	forbidden [exception]	

```
// General
   System.out.printf("%.3b", true);  |           "tru" | Boolean
   System.out.printf("%.2h", 1500);  |            "5d" | Hexadecimal String
System.out.printf("%.3s", "Hello");  |           "Hel" | General - String

// Numeric - Floating point
System.out.printf("%.2e", 1234.45);  |      "1.23e+03" | Scientific Notation
System.out.printf("%.1f", 1234.45);  |        "1234.5" | Decimal Float
System.out.printf("%.3g", 1234.45);  |      "1.23e+03" | Scientific Notation or Decimal
                                                        | Float
```

Width

The optional width is the minimum number of characters. Not applicable to the line separator conversion (nor percentage literal). If the number of characters in the argument exceeds the width, the width is ignored.

```
// Basic Width Examples
     System.out.printf("%10b", true);  |    "      true" | Boolean
       System.out.printf("%10h", 15);  |    "         f" | Hexadecimal String
   System.out.printf("%10s", "Cool");  |    "      Cool" | String
    System.out.printf("%3s", "Cool");  |          "Cool" | String [Width Ignored]
```

Precision (max chars) is applied then width (min chars).

```
System.out.printf("%10.3s", "Cool");  |    "       Coo" | String
```

Flags

The optional *flags* modifies the output according to the following table:

Flag	General	Character	Integral	Floating Point	Date/Time	Description
'-'	y	y	y	y	y	The result will be left-justified. (must be accompanied by a width specifier).
'#'	y1	-	y3	y	-	The result should use a conversion-dependent alternate form
'+'	-	-	y4	y	-	The result will always include a sign (only applicable for postives).
' '	-	-	y4	y	-	The result will include a leading space for positive values
'0'	-	-	y	y	-	The result will be zero-padded
','	-	-	y2	y5	-	The result will include locale-specific grouping separators
'('	-	-	y4	y5	-	The result will enclose negative numbers in parentheses

1 Depends on the definition of Formattable.

2 For 'd' conversion only.

3 For 'o', 'x', and 'X' conversions only.

4 For 'd', 'o', 'x', and 'X' conversions applied to BigInteger or 'd' applied to byte, Byte, short, Short, int and Integer, long, and Long.

5 For 'e', 'E', 'f', 'g', and 'G' conversions only.

(Oracle, 2012) *http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Formatter.html*

Flags examples

```
// Left-Justified '-'
    System.out.printf("%10s", "Cool");  |  "      Cool" | String - Default right justified
   System.out.printf("%-10s", "Cool");  |  "Cool      " | String - Left justified
       System.out.printf("%-10d", 15);  |  "15        " | Decimal Integer - Left justified
System.out.printf("%-10.2f", 1234.45);  |  "1234.45   " | Decimal Float - Left justified

// Alternative form '#'
        System.out.printf("%o", 1234);  |        "2322" | Octal Integer
       System.out.printf("%#o", 1234);  |       "02322" | Octal Integer - Alternate
          System.out.printf("%x", 15);  |          "f" | Hexadecimal Integer
         System.out.printf("%#x", 15);  |         "0xf" | Hexadecimal Integer - Alternate
```

```
// Postives: Sign always '+'
          System.out.printf("%d", 15);  |          "15" | Decimal Integer
         System.out.printf("%+d", 15);  |         "+15" | Decimal Integer
         System.out.printf("%d", -15);  |         "-15" | Decimal Integer
        System.out.printf("%+d", -15);  |         "-15" | Decimal Integer

// Positives: Space always ' '
          System.out.printf("%d", 15);  |          "15" | Decimal Integer
         System.out.printf("% d", 15);  |         " 15" | Decimal Integer
         System.out.printf("%d", -15);  |         "-15" | Decimal Integer
        System.out.printf("% d", -15);  |         "-15" | Decimal Integer

// Zero padded '0'
       System.out.printf("%010d", 15);  |  "0000000015" | Decimal Integer
      System.out.printf("%010d", -15);  |  "-000000015" | Decimal Integer

// Locale Specific grouping separators ','
  System.out.printf("%, d", 15890890);  | " 15,890,890" | Decimal Integer
  System.out.printf("%,d", -15890890);  | "-15,890,890" | Decimal Integer

// Negatives: Bracket Enclosed '('
          System.out.printf("%(d", 15);  |          "15" | Decimal Integer
         System.out.printf("%(d", -15);  |        "(15)" | Decimal Integer
```

You can have multiple flags (but not all combinations work).

```
System.out.printf("$%2$- ,(15.2f", "Nice", -1234.23425);  | "$(1,234.23)     "
```

Date/Time Conversions

Get and display now

```
// Fetch the current date/time
Calendar calendar = Calendar.getInstance();
System.out.printf("%tF", calendar);  |  "2013-07-03"

// In one
System.out.printf("%tF", Calendar.getInstance()); |  "2013-07-03"
```

Declare, define, and display a date/time. Months are zero based.

```
Calendar myDateTime = new GregorianCalendar(2013, 1, 15, 16, 20, 45);
System.out.printf("%1$tF %1$tR", myDateTime);
// "2013-02-15 16:20"
```

Date/Time conversation format specifications entail a "t" prefix and a date/time suffix.

```
// Basic
   System.out.printf("%tF", calendar);  |  "2013-07-03" | Common Composite
System.out.printf("%1$tH:%1$tM:%1$tS", calendar);  |    "12:48:03" | Individual


// Dates - Individual
System.out.printf("%1$tY-%1$tm-%1$td", calendar);  |  "2013-07-03" | Date (ISO 8601)
System.out.printf("%1$tY-%1$tm-%1$td", calendar);  |  "2013-07-03" | Date (ISO 8601 with)


// Times - Individual
System.out.printf("%1$tH:%1$tM:%1$tS", calendar);  |    "12:48:03" | Time (ISO 8601)

System.out.printf("%1$tH:%1$tM:%1$tS %1$tz %1$tZ", calendar);  | "12:48:03 +1000 EST"
| Time (ISO 8601 with Timezone)


// Date/Time (ISO 8601) Individual
System.out.printf("%1$tY-%1$tm-%1$td %1$tH:%1$tM:%1$tS %1$tz %1$tZ", calendar);  | "2013-07-
03 12:48:03 +1000 EST" | Individual


// Date/Time (ISO 8601) Common Composite
System.out.printf("%tF", calendar);  |  "2013-07-03" | Date

System.out.printf("%1$tF %1$tR", calendar);  | "2013-07-03 12:48"
| Date, Time without seconds

System.out.printf("%1$tF %1$tT", calendar);  | "2013-07-03 12:48:03"
| Date, Time with seconds

System.out.printf("%1$tF %1$tT %1$tz %1$tZ", calendar);  | "2013-07-03 12:48:03 +1000 EST"
| Date, Time with seconds, Timezone
```

(Bentley, 2013) C:\Users\John\Documents\Sda\Code\Java\Examples\TutorialAtOracle\HelloWorld – PrintfStyleFormattingDemo.java

Date suffixes.

'B'	Locale-specific full month name, e.g. "January", "February".
'b'	Locale-specific abbreviated month name, e.g. "Jan", "Feb".
'h'	Same as 'b'.

'A'	Locale-specific full name of the day of the week, e.g. "Sunday", "Monday"
'a'	Locale-specific short name of the day of the week, e.g. "Sun", "Mon"
'C'	Four-digit year divided by 100, formatted as two digits with leading zero as necessary, i.e. 00 - 99
'Y'	Year, formatted as at least four digits with leading zeros as necessary, e.g. 0092 equals 92 CE for the Gregorian calendar.
'y'	Last two digits of the year, formatted with leading zeros as necessary, i.e. 00 - 99.
'j'	Day of year, formatted as three digits with leading zeros as necessary, e.g. 001 - 366 for the Gregorian calendar.
'm'	Month, formatted as two digits with leading zeros as necessary, i.e. 01 - 13.
'd'	Day of month, formatted as two digits with leading zeros as necessary, i.e. 01 - 31
'e'	Day of month, formatted as two digits, i.e. 1 - 31.

Time suffixes.

'H'	Hour of the day for the 24-hour clock, formatted as two digits with a leading zero as necessary i.e. 00 - 23.
'I'	Hour for the 12-hour clock, formatted as two digits with a leading zero as necessary, i.e. 01 - 12.
'k'	Hour of the day for the 24-hour clock, i.e. 0 - 23.
'l'	Hour for the 12-hour clock, i.e. 1 - 12.
'M'	Minute within the hour formatted as two digits with a leading zero as necessary, i.e. 00 - 59.
'S'	Seconds within the minute, formatted as two digits with a leading zero as necessary, i.e. 00 - 60 ("60" is a special value required to support leap seconds).
'L'	Millisecond within the second formatted as three digits with leading zeros as necessary, i.e. 000 - 999.
'N'	Nanosecond within the second, formatted as nine digits with leading zeros as necessary, i.e. 000000000 - 999999999.
'p'	Locale-specific morning or afternoon marker in lower case, e.g."am" or "pm". Use of the conversion prefix 'T' forces this output to upper case.
'z'	RFC 822 style numeric time zone offset from GMT, e.g. -0800.
'Z'	A string representing the abbreviation for the time zone. The Formatter's locale will supersede the locale of the argument (if any).
's'	Seconds since the beginning of the epoch starting at 1 January 1970 00:00:00 UTC, i.e. Long.MIN_VALUE/1000 to Long.MAX_VALUE/1000.
'Q'	Milliseconds since the beginning of the epoch starting at 1 January 1970 00:00:00 UTC, i.e. Long.MIN_VALUE to Long.MAX_VALUE.

Common composite date/time suffixes.

'R'	Time formatted for the 24-hour clock as "%tH:%tM"
'T'	Time formatted for the 24-hour clock as "%tH:%tM:%tS".
'r'	Time formatted for the 12-hour clock as "%tI:%tM:%tS %Tp". The location of the morning or afternoon marker ('%Tp') may be locale-dependent.
'D'	Date formatted as "%tm/%td/%ty".

'F'	ISO 8601 complete date formatted as "%tY-%tm-%td".
'c'	Date and time formatted as "%ta %tb %td %tT %tZ %tY", e.g. "Sun Jul 20 16:17:00 EDT 1969".

References, Word

Bentley, J. (2013, Jul). TutorialAtOracle Code Examples - HelloWorld. Retrieved from C:\Users\John\Documents\Sda\Code\Java\Examples\TutorialAtOracle\HelloWorld

Oracle. (2011). *Java Platform, Standard Edition 6, API Specification*. Retrieved Jul 09, 2013, from http://docs.oracle.com: http://docs.oracle.com/javase/6/docs/api/overview-summary.html

Oracle. (2012). *The Java Tutorials.* Retrieved Jun 14, 2012, from http://docs.oracle.com/: http://docs.oracle.com/javase/tutorial/

Document Licence