Selected Java Numeric & Text Format Specifiers
Collected and Revised by Charles S. Tritt, Ph.D.
Version 1.0, Last revised 4/18/05

This is a revised and simplified version of the Java Format class documentation. It applies to the *java.util.Formatter.format(), java.io.PrintStream.format()* and *java.io.PrintStream.printf()* methods.

Format String Syntax

Every method which produces formatted output requires a *format string* and an *argument list*. The format string is a `String` which may contain literal text and one or more embedded *format specifiers*. Consider the following example:

```
double t = ...;
int c = ...;
Formatter.format("After %3d cycles, the temperature is %6.1f.", c, t);
```

This format string is the first argument to the `format` method. It contains literal text and two format specifiers, "`%3d`" and "`%6.1f`". The format specifiers indicate how the remaining arguments should be processed and where they should be inserted in the text. The argument list consists of all arguments passed to the method after the format string. In the above example, the argument list is of size two and consists of an *Integer* and a *Double* (autoboxed versions of the specified *int* and *double* arguments).

The format specifiers for general, character, and numeric types have the following syntax (items in square brackets, "`[ ]`" are optional):

```
%[argument_index$][flags][width][.precision]conversion
```

The optional *argument_index* is a decimal integer indicating the position of the argument in the argument list. The first argument is referenced by "`1$`", the second by "`2$`", etc.

The optional *flags* is a set of characters that modify the output format. The set of valid flags depends on the conversion.

The optional *width* is a non-negative decimal integer indicating the minimum number of characters to be written to the output.

The optional *precision* is a non-negative decimal integer usually used to restrict the number of characters. The specific behavior depends on the conversion.

The required *conversion* is a character indicating how the argument should be formatted. The set of valid conversions for a given argument depends on the argument's data type.

Selected Conversion Specifiers

Conversion	Argument Category	Description
`'b'`, `'B'`	general	If *arg* is a *`boolean`* or *Boolean*, then the result is the string returned by `String.valueOf()`.
`'s'`, `'S'`	general	Generally used for Strings. The result is obtained by invoking `arg.toString()`.
`'c'`, `'C'`	character	The result is a Unicode character
`'d'`	integral	The result is formatted as a decimal integer
`'o'`	integral	The result is formatted as an octal integer
`'x'`, `'X'`	integral	The result is formatted as a hexadecimal integer
`'e'`, `'E'`	floating point	The result is formatted as a decimal number in computerized scientific notation
`'f'`	floating point	The result is formatted as a decimal number
`'g'`, `'G'`	floating point	The result is formatted using computerized scientific notation or decimal format, depending on the precision and the value after rounding.
`'t'`, `'T'`	date/time	Prefix for date and time conversion characters. See my Date/Time format handout for details.
`'%'`	percent	The result is a literal `'%'` (`'\u0025'`)
`'n'`	line separator	The result is the platform-specific line separator

Width

The width is the minimum number of characters to be written to the output. For the line separator conversion, width is not applicable; if it is provided, an exception will be thrown.

Precision

For general argument types, the precision is the maximum number of characters to be written to the output.

For the floating-point conversions `'e'`, `'E'`, and `'f'` the precision is the number of digits after the decimal separator. If the conversion is `'g'` or `'G'`, then the precision is the total number of digits in the resulting magnitude after rounding. If the conversion is `'a'` or `'A'`, then the precision must not be specified.

For character, integral, and date/time argument types and the percent and line separator conversions, the precision is not applicable; if a precision is provided, an exception will be thrown.

Flags

The following table summarizes the supported flags. *y* means the flag is supported for the indicated argument types.

Flag	General	Character	Integral	Floating Point	Date/Time	Description
'-'	y	y	y	y	y	The result will be left-justified.
'#'	y	-	y	y	-	The result should use a conversion-dependent alternate form
'+'	-	-	y	y	-	The result will always include a sign
' '	-	-	y	y	-	The result will include a leading space for positive values
'0'	-	-	y	y	-	The result will be zero-padded
','	-	-	y	y	-	The result will include locale-specific grouping separators
'('	-	-	y	y	-	The result will enclose negative numbers in parentheses