## What is Java printf?

The Java printf function helps simplify the task of printing formatted output to the console, terminal window or log files.

The Java printf function makes it easier to create informative strings of text without using String concatenation, or relying on non-standard characters that might trigger output errors.

In Java, printf is recommended as a replacement for calls to the print or println methods.

## How do you format output with Java printf?

To format text based output with the Java printf method, follow these steps:

1. Create a text String literal that includes variable substitution points
2. Use the appropriate conversion specifiers where variable substitution occurs
3. Add additional flags to the conversion specifiers to format for width, padding and annotations
4. Pass the String to the Java printf method and supply corresponding arguments for each conversion specifier

### How do you format a String with Java printf?

The easiest way to understand how the Java printf method works is to examine a sample code snippet.

The following Java printf example formats a text String, and substitutes two variables into the text where the %s and %S conversion specifiers appear:

|  |
| --- |
| **User input in Java made easy** |
| Learn the best ways to handle input from the Java user, and how to format any console output with printf.   * Top 3 ways to get [Java user input](https://www.theserverside.com/tutorial/Handling-input-in-a-stand-alone-Java-program) in your app * The importance of the [Java Scanner import](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/How-to-run-an-Apache-Docker-http-image-with-local-files-example) statement * A simple way to [input text Strings with Java’s Scanner](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Java-String-input-with-the-Scanner-class) * A simple [Scanner char input](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Java-Scanner-char-input-example) example * How to get [Java user input with the Scanner](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/User-input-with-the-Java-Console-class) |

public class FormatOutputJavaPrintf {

/\* Simple Java printf String example. \*/

public static void main(String[] args) {

String name = "Cameron";

String site = "TechTarget";

System.out.printf(**"I like the stuff %s writes on %S. %n", name, site**);

/\* Printf output: I like the stuff Cameron writes on TECHTARGET. \*/

}

}

This Java printf String example generates the following formatted output:

I like the articles Cameron writes on TECHTARGET.

This simple Java printf example provides three important takeaways:

* The %s printf specifier does not change letter casing.
* The %S printf specifier changes letter casing to uppercase.
* The variables must be listed in the order they are referenced in the printf String.

| **Java String printf examples** | | |
| --- | --- | --- |
| Pattern | Data | Printf Output |
| '%s' | Java | 'Java' |
| '%15s' | Java | ' Java' |
| '%-15s ' | Java | 'Java ' |
| '%-15S ' | Java | 'JAVA ' |

#### What does %n in Java printf mean?

When you format a String with Java printf, you can use %n in place of \n to specify a line break.

### What is the standard Java printf syntax?

The format of the Java printf conversion specifiers always follows the same pattern:

% [flags] [width] [.precision] specifier-character

The flags, width and precision settings are all optional. However, the character specifier must match the data type of the corresponding argument, or else a formatting error will occur.

With pure text Strings, the only flag that makes sense is the minus sign ( - ). This formats text to the left if the number of characters exceeds the width setting.

Most of the Java printf flags are intended for use with integers and floating point numbers.

| **Java printf format flags** | |
| --- | --- |
| printf flag | Purpose |
| – | Aligns the formatted printf output to the left |
| + | The output includes a negative or positive sign |
| ( | Places negative numbers in parenthesis |
| 0 | The formatted printf output is zero padded |
| , | The formatted output includes grouping separators |
| <space> | A blank space adds a minus sign for negative numbers and a leading space when positive |

### How do you format an integer with printf in Java?

To format a digit or integer with Java printf:

* Use %d as the conversion specifier for Base-10 numbers.
* Precede the letter d with a comma to group numbers by the thousands.
* Add an optional + flag to cause positive numbers to display a positive sign.
* Use the 0 flag to zero-pad the number to fill up the space specified by the width.

package com.mcnz.printf.example;

public class JavaPrintfInteger {

/\* Format integer output with Java printf \*/

public static void main(String[] args) {

int above = -98765;

long below = 54321L;

System.out.printf("%,d :: %d", above, below);

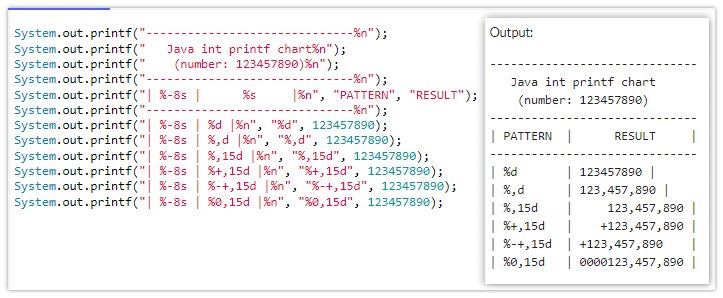
/\* Example prints: **-00098,765 :: +54,321** \*/

}

}

| **Hex, oct and dec integer printf examples** | | |
| --- | --- | --- |
| Pattern | Data | Printf output |
| ‘%d’ | 123,457,890 | '123457890' |
| ‘%,15d’ | 123,457,890 | ' 123,457,890' |
| ‘%+,15d’ | 123457890 | ' +123,457,890' |
| ‘%-+,15d’ | 123457890 | '+123,457,890 ' |
| ‘%0,15d’ | 123457890 | '0000123,457,890' |
| ‘%15o’ | 123457890 | ' 726750542' |
| ‘%15x’ | 123457890 | ' 75bd162' |

The following image shows how to perform advanced Java printf integer formatting:



Java double and float values can easily be formatted and output with Java printf.

### How do you format a Java double with printf?

To format a [Java float or double](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Float-vs-Double-Whats-the-difference) with printf, follow the same steps to format an integer with the following two addendums:

1. The conversion specifier for a floating point number is %f, not %d.
2. Use the precision parameter to truncate decimals.

Here is a simple example of how to use Java printf to format floats and double output:

package com.mcnz.printf.example;

public class FloatingPointPrintfExample {

/\* Format float and double output with printf. \*/

public static void main(String[] args) {

double top = 1234.12345;

float bottom = 1234.12345f;

System.out.printf("%+,.3f :: %,.5f", top, bottom);

/\* Example prints: +**1,234.123 :: 1234.12345** \*/

}

}

The %+,.3f setting breaks down like this:

* The plus symbol + instructs Java printf to include a plus sign if the double is positive.
* The comma instructs Java printf to add a thousandths group separator.
* The .3 instructs Java printf to limit output to three decimal places.
* %f is the specifier used to format double and floats with printf.

| **Java printf format specifiers** | |
| --- | --- |
| Printf specifier | Data type |
| %s | String of text |
| %f | floating point value (float or double) |
| %e | Exponential, scientific notation of a float or double |
| %b | boolean true or false value |
| %c | Single character char |
| %d | Base 10 integer, such as a Java int, long, short or byte |
| %o | Octal number |
| %x | Hexadecimal number |
| %% | Percentage sign |
| %n | New line, aka carriage-return |
| %tY | Year to four digits |
| %tT | Time in format of HH:MM:SS ( ie 21:46:30) |

#### Scientific notation in Java with printf

To output a floating point number using scientific notation, simply use %e instead of %f to specify the variables.

package com.mcnz.scientific.notation;

public class PrintfScientificNotationExample {

/\* Format float and double output with printf. \*/

public static void main(String[] args) {

double top = 1234.12345;

float bottom = 1234.12345f;

System.out.printf("%+,.3e :: %,.5e", top, bottom);

/\* Example prints: +1.234e+03 :: 1.23412e+03 \*/

}

}

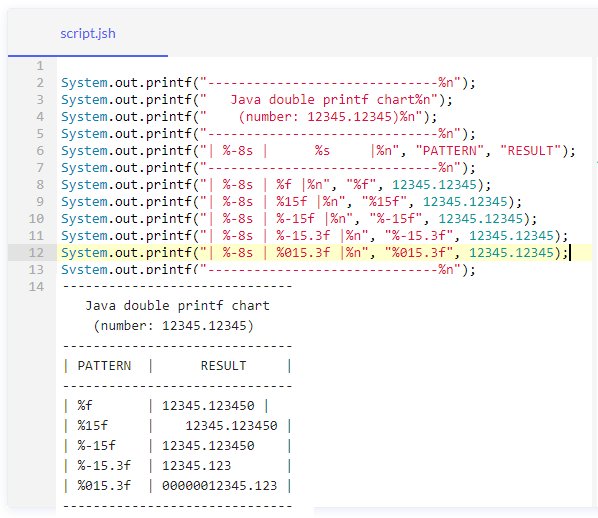
Notice how this example prints out +1.234e+03 :: 1.23412e+03 rather than +1,234.123 :: 1234.12345, which is what the preceding example generates.

##### FormatFlagsConversionMismatchException

Be sure to remove any commas in the specifier, as attempting to add thousandths groupings for scientific notation will result in a FormatFlagsConversionMismatchException.

#### Advanced double formatting with printf in Java

The following image demonstrates the use of various flags, precision settings and width specifiers to create a table containing advance formatting of doubles with printf.



Here’s the code and output of a table of Java double printf examples.

#### How do you format a Java char or boolean with printf?

In Java, a char uses the %c specifier. A boolean values uses %b.

If you use %C or %B is used to format a boolean or char with Java printf, the values are printed in uppercase.

package com.mcnz.scientific.notation;

public class PrintfCharBooleanExample {

/\* Boolean char Java printf example. \*/

public static void main(String[] args) {

boolean flag = false;

char coal = 1234.12345f;

System.out.printf("%B :: %c :: %C", flag, coal, '\u0077');

/\* Example prints: FALSE :: a :: W \*/

}

}

As you can see in this example, the printf statement not only formats char variables, but it also converts and properly prints out unicode character numbers.

### How do you format a Date in Java with printf?

The best way to output date and time values in Java is with the DateTimeFormatter class. But the printf method supports simple date formatting through the use of the %t conversion specifier along with an additional date detail parameter.

With a java.util.Date object, we can use the following options:

* %tH to display the hour
* %tM to display the minutes
* %tS to display the seconds
* %tp to print am or pm
* %tx for the time-zone offset

#### Java printf date example

The following example formats the time with printf:

Date d = new Date();

System.out.printf("%tH %tM %tS %tz", d, d, d, d);

/\* Outputs: 22 26 49 -0400 \*/

### How do you format local time in Java with printf?

Modern programs prefer to use the LocalDateTime object over java.util.Date.

The following parameters are commonly used to format DateTime output with Java printf:

* %tA to printf the day of the week in full
* %ta to printf the abbreviated day of the week
* %tB to printf the full name of the month
* %tb to printf the abbreviated month name
* %td to printf the day of the month
* %tY to output all four digits of the year
* %tY to printf the last two digits of the year
* %tp to display am or pm
* %tL to display the millisecond offset
* %tz to display the time-zone offset

#### Java printf time example

The following Java printf time example demonstrates how to format values defined in a LocalDateTime object:

LocalDateTime dt = LocalDateTime.now();

System.out.printf(" %ta %te, %tY %tT %tp ", dt, dt, dt, dt, dt);

/\* Formats LocalDateTime output as: Sat 6, 2022 21:19:56 pm \*/

This Java time printf example prints out:  
Sat 6, 2022 21:19:56 pm

### How do you format a table with printf?

Sometimes developers want to neatly format complicated text data before it is printed to the console or a log file.

The Java printf command does not provide any built-in features that help structure output. However, creative use of the Java printf method can generate data tables.

For example, the following code generates a table that documents the properties of the Java primitive types:

System.out.printf("--------------------------------%n");

System.out.printf(" Java's Primitive Types %n");

System.out.printf(" (printf table example) %n");

System.out.printf("--------------------------------%n");

System.out.printf("| %-10s | %-8s | %4s |%n", "CATEGORY", "NAME", "BITS");

System.out.printf("--------------------------------%n");

System.out.printf("| %-10s | %-8s | %04d |%n", "Floating", "double", 64);

System.out.printf("| %-10s | %-8s | %04d |%n", "Floating", "float", 32);

System.out.printf("| %-10s | %-8s | %04d |%n", "Integral", "long", 64);

System.out.printf("| %-10s | %-8s | %04d |%n", "Integral", "int", 32);

System.out.printf("| %-10s | %-8s | %04d |%n", "Integral", "char", 16);

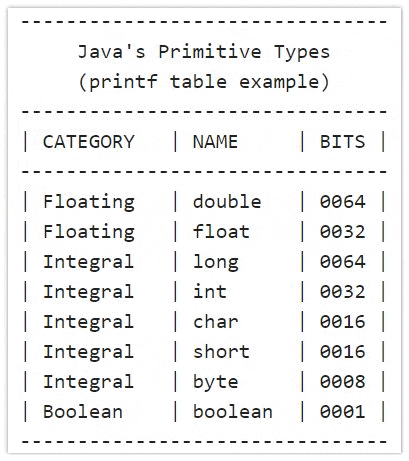
System.out.printf("| %-10s | %-8s | %04d |%n", "Integral", "short", 16);

System.out.printf("| %-10s | %-8s | %04d |%n", "Integral", "byte", 8);

System.out.printf("| %-10s | %-8s | %04d |%n", "Boolean", "boolean", 1);

System.out.printf("--------------------------------%n");

When the code runs, a tabular output is created.



Create a Java printf table through the creative use of width, precision and printf format flags.

### Java printf examples

The Java printf statement greatly simplifies the task for formatting output generated in a Java program.

Learn the fundamentals of the Java printf statement, and you will save a great deal of time when generating output for the console, logs and other text-based output streams.

| **Java printf cheatsheet** | | |
| --- | --- | --- |
| Pattern | Data | Printf output |
| ‘%s’ | Java | 'Java' |
| ‘%15s’ | Java | ' Java' |
| ‘%-15s ‘ | Java | 'Java ' |
| ‘%d’ | 123,457,890 | '123457890' |
| ‘%,15d’ | 123,457,890 | ' 123,457,890' |
| ‘%+,15d’ | 123457890 | ' +123,457,890' |
| ‘%-+,15d’ | 123457890 | '+123,457,890 ' |
| ‘%0,15d’ | 123457890 | '0000123,457,890' |
| ‘%15o’ | 123457890 | ' 726750542' |
| ‘%15x’ | 123457890 | ' 75bd162' |
| ‘%15f’ | 12345.123450 | ' 12345.123450' |
| ‘%-15.3f’ | 12345.123450 | '12345.123 ' |
| ‘%015.3f’ | 12345.123450 | '00000012345.123' |
| ‘%e’ | 12345.123450 | '1.234579e+08' |
| ‘%.2e’ | 12345.123450 | '1.23e+08' |
| ‘%7tH %<-7tM’ | new Date() | ' 22 35 ' |
| ‘%15tT’ | LocalDateTime.now() | ' 22:35:53' |