**We Have a printf for That!**

Even in programming, appearances matter. Attention to the details of formatting distinguishes an accomplished programmer from a novice.

Remember, your program communicates with anyone who uses it and will leave a lasting impression. Regardless of the quality of your code, if your program is not user-friendly, it will leave an unfavorable reaction. Formatting is actually very simple, but perfecting horizontal and vertical alignment can be time-consuming. You can drastically reduce the effort (and frustration) associated with formatting by practicing one simple strategy: plan before you print. Most newbie programmers don't heed this advice until after they have wasted a lot of time. Establish good habits now and you will save time in the future.

Formatting output in Java has not always been easy, but Java solved this problem by adding a method commonly used in the C programming language to the [PrintStream class](https://l.flvsgl.com/GSL0978f93819e481697621edf45b45af7b3): **printf()**.

Most simple programs produce output such as text, numbers, dates, or times for display on the computer's monitor. It is good programming practice to format all output so that it is easy to read. For example, which of the following three columns of numbers is easier to view?

|  |  |  |
| --- | --- | --- |
| 23.456 | 23.456 | 23.456 |
| 1.2 | 1.2 | 1.200 |
| -99.712345 | -99.7123456 | -99.712 |
| .98 | 98 | 0.980 |

There may be times when it is appropriate to center or left-justify columns of numbers, but in this case those formats are not appealing. The column on the right is easier to read because the numbers are right justified, the decimal points are lined up, and all numbers are expressed to the same number of decimal places.

In addition to numbers, you will also want to be able to format text. Which of the following do you prefer?

|  |  |  |
| --- | --- | --- |
| Tallahassee | Tallahassee | Tallahassee |
| Orlando | Orlando | Orlando |
| Los Angeles | Los Angeles | Los Angeles |
| Oklahoma City | Oklahoma City | Oklahoma City |

We generally expect to see a list of words left-justified as shown in the first column, but can you think of situations where you have seen text centered or right justified? Poems, menus, and financial reports are examples that may come to mind.

The **printf()** method solves all of these formatting problems and more. It is also straightforward to use, once you learn its rather unusual syntax.

### Part 1

Java's **print()** and **println()** methods provide no formatting capabilities—WYSIWYG. Advanced formatting methods are available to achieve precise control of output (e.g., currency, dates, time, Strings, etc.), but they require several extra steps. Recognizing the need for a simpler way to format output, developers introduced the **printf()** method. You can substitute **printf()** for **print()** and **println()**to display output to the screen or stream output to a file.

The key feature of **printf()** is the inclusion of a formatting code as part of the method's argument. This disrupts the appearance of the print statement in the source code, but actually controls the appearance of the output. It will take some getting used to, as the following examples illustrate:

System.out.printf("City: **%-15s%n**", cityName);  
System.out.printf("Zip Code: **%10d%n**", zipCode);  
System.out.printf("pi = **%10.4f%n**",pi);

These three examples indicate the typical syntax of a **printf()** statement:

* a String literal to be printed (e,g., "City: " , "Zip Code: ", and "pi: ")
* a format specifier (%-15s%n , %10d%n, %10.4f%n)
* a variable to be printed (e.g., cityName, zipCode, pi)

With a little practice, using **printf()** will become automatic!

### Part 2

Notice that a format specifier is imbedded within the **String** literal, essentially as a placeholder for a variable; it always starts with a percent sign (%) and ends with a converter (e.g., s, d, or f). The percent sign resembles the escape character (\) in that it alerts Java to not print what follows, but to do something different based on the sequence of characters. The converter is a character indicating the type of the argument to be formatted. Optional flags or specifiers can be sandwiched between the percent sign and the type. So the dissected format specifier consists of these parts.

1. The leading percent sign (%)
2. Optional flag or specifier (+, -,,)
3. Field width (optional)
4. A converter (s, d, or f)

If only one variable will be printed on a line, the format specifier must include %n to force a linefeed in the same way the escape sequence \n was used with the **print()** method.

When a variable is printed, the format specifier precisely controls how the value will be printed to the screen or written to a file.

* Create a project called 06.02 Formatting Output in the Mod06 Lessons folder.
* Open the [FormattingStrings.java](https://lti.flvsgl.com/flvs-cat-content/ebhovt6cn9e590jguh08332rtk/flvs-cat-session/apcomputersciencea_v20/module06/lesson02/javamod06/FormattingStrings.java) and [FormattingNumericValues.java](https://lti.flvsgl.com/flvs-cat-content/ebhovt6cn9e590jguh08332rtk/flvs-cat-session/apcomputersciencea_v20/module06/lesson02/javamod06/FormattingNumericValues.java) files to the new project.
* Open the [06.02 Virtual Lecture Notes (Part 1)](https://lti.flvsgl.com/flvs-cat-content/ebhovt6cn9e590jguh08332rtk/flvs-cat-session/apcomputersciencea_v20/module06/lesson02/pop/06_02b/06_02b_pop01.htm) and [06.02 Virtual Lecture Notes (Part 2)](https://lti.flvsgl.com/flvs-cat-content/ebhovt6cn9e590jguh08332rtk/flvs-cat-session/apcomputersciencea_v20/module06/lesson02/pop/06_02b/06_02b_pop02.htm) .
* Follow the instructions and experiment with the **printf()** method.

The **printf()** method allows precise manipulation of horizontal spacing on a line and within a field, as well as a way to control decimal places and special characters within a number. From now on, use the **printf()** method whenever output needs to be accurately formatted to the monitor or a file.

You will appreciate the improved formatting capabilities of the **printf()** method once you gain some experience. No more ragged columns of numbers with misaligned decimal places!