Java's printf() Method

Syntax: System.out.printf("format-string" <, arg1, arg2, …, argn>);
Syntax: String s = System.format("format-string" <, arg1, arg2, …, argn>);

- format-string
  It is composed of literals and format specifiers.
  Arguments are required only if there are format specifiers in the format-string.
  Format specifiers include: flags, width, precision, and conversion characters in the following sequence:
  %<flags><width><.precision><conversion-character>
  <> denote optional parameters

- flags
  - : Left-justify (default is right-justify)
  +: Output a plus (+) or minus (-) sign for a numerical value
  ' ': Space will display a minus sign if the number is negative or a space if it is positive
  0: Force numerical values to be zero-padded (default is blank padding)
  , : Comma grouping separator (for numbers > 1000)
  (: Encloses negative numbers in parentheses
  #: Uses an alternate form for octal and hexadecimal output

- width
  Specifies the data width for outputting the argument and represents the minimum number of characters to be written to the output. Include space for expected commas and a decimal point in the determination of the width for numerical values

- precision
  Used to restrict the output depending on the conversion. It specifies the number of digits of precision when outputting floating-point values or the length of a substring to extract from a String. Numbers are rounded to the specified precision.

- conversion-character
  a / A: floating-point in hexadecimal format

b: any type	"true" if non-null, "false" if null, capital B will upper the word
c: character	Capital C will uppercase the letter
d: decimal integer	[byte, short, int, long]
e / E: floating-point in exponential notation	[float, double]

f: floating-point number [float, double]

g / G: floating-point number, possibly in exponential notation depending on the precision and value [float, double]

h: hashcode A hashcode is like an address. This is useful for printing a reference

n: newline Platform specific newline character – use %n instead of \n for greater compatibility

o: integer in octal format [byte, short, int, long]

s: String Capital S will uppercase all the letters in the string

t: date / time

x: integer in hexadecimal format [byte, short, int, long]

Integer Formatting

"%d"	Format a string with as many numbers as are needed
"%4d"	Format a string with the specified number of integers, and right-justify. Will pad with spaces to the left if not enough integers.
"%-4d"	Format a string with the specified number of integers, and left-justify. Will pad the spaces to the right if not enough integers.
"%04d"	Format a string with the specified number of integers. Will pad with zeros to the left if not enough integers.
"%.4d"	Format a string with maximum number of digits of the integer.

Floating Point Number Formatting

"%f"	Format a string with as many numbers as are needed. Will always give you six decimal places.
"%10f"	Format a string with as many numbers as are needed. If the number has less than 10 digits, the output will be padded on the left.
"%.2f"	Format a string with as many numbers as are needed. Gives 2 decimal places.
"%10.2f"	Format to 2 decimal places, but the whole string occupies 10 characters. If there's not enough numbers, then spaces are used to the left of the numbers.

String Formatting

"%s"	Format a string with as many characters as are needed
"%15s"	Format a string with the specified number of characters, and right-justify
"%-15s"	Format a string with the specified number of characters, and left-justify
"%.15s"	Format a string with maximum number of characters of the string

Example:

```
import java.util.Scanner;

public class TestPrintf
{
  public static void main(String[] args)
  {
    // ======================
    //  Displaying integers
    // ======================
    // ----
    //  %d
    // ----

    // Defult formatting
    // Prints 99
    System.out.printf("%d%n", 99);

    // Specifying a width
    // Prints |                  99|
    System.out.printf("|%20d|%n", 99);

    // Left-justifying within the specified width
    // Prints |99                  |
    System.out.printf("|%-20d%n", 99);

    // Padding with zeros
    // Prints |00000000000000000093|
    System.out.printf("|%020d|%n", 99);

    // Print positive numbers with a "+"
    // (Negative numbers always have the "-" included)
    // Prints |                 +99|
    System.out.printf("|%+20d|%n", 99);

    // A space before positive numbers
    // A "-" is included for negative numbers as per normal
    // Prints |  93|
    System.out.printf("|% d|%n", 99);

    // Use locale-specific thousands separator
    // Prints |10,000,000|
    System.out.printf("|%,d|%n", 10000000);

    // Enclose negative numbers within paratheses ("()") and skip the "-"
    // Prints |(36)|
    System.out.printf("|%(d|%n", -36);

    // ----
    //  %o
    // ----

    // Octal output
    // Prints 135
    System.out.printf("|%o|%n", 93);

    // ----
    //  %x
    // ----

    // Hex output
    // Prints 5d
    System.out.printf("|%x|%n", 93);
```

```java
// ==================================
//  Displaying floating point values
// ==================================
// ----
//  %f
// ----

// Floating-point number
// Prints 10.200000
System.out.printf("%f%n", 10.2);

// Floating-point number
// Prints 0.000001
System.out.printf("%f%n", 0.000001234);

// Floating-point number
// Prints |123456.79|
System.out.printf("|%.2f|%n", 123456.789);

// Floating-point number
// Prints |123456.79|
System.out.printf("|%8.2f|%n", 123456.789);

// Floating-point number
// Prints | 123456.79|
System.out.printf("|%10.2f|%n", 123456.789);

// Floating-point number
// Prints |123456.79 |
System.out.printf("|%-10.2f|%n", 123456.789);

// Floating-point number
// Prints |0123456.79|
System.out.printf("|%010.2f|%n", 123456.789);

// ----
//  %e
// ----

// Floating-point in exponential notation
// Prints 1.020000e+01
System.out.printf("%e%n", 10.2);

// Floating-point in exponential notation
// Prints 1.234000e-06
System.out.printf("%e%n", 0.000001234);

// Floating-point number
// Prints |1.23e+05|
System.out.printf("|%.2e|%n", 123456.789);

// Floating-point number
// Prints |1.23e+05|
System.out.printf("|%8.2e|%n", 123456.789);

// Floating-point number
// Prints |  1.23e+05|
System.out.printf("|%10.2e|%n", 123456.789);

// Floating-point number
// Prints |1.23e+05  |
System.out.printf("|%-10.2e|%n", 123456.789);

// Floating-point number
// Prints |001.23e+05|
System.out.printf("|%010.2e|%n", 123456.789);
```

```
// ----
//  %g
// ----

// Floating-point number, possibly in exponential notation depending
// on the precision and value
// Prints 10.2000
System.out.printf("%g%n", 10.2);

// Floating-point number, possibly in exponential notation depending
// on the precision and value
// Prints 1.23400e-06
System.out.printf("%g%n", 0.000001234);

// Floating-point number
// Prints |1.2e+05|
System.out.printf("|%.2g|%n", 123456.789);

// Floating-point number
// Prints | 1.2e+05|
System.out.printf("|%8.2g|%n", 123456.789);

// Floating-point number
// Prints |   1.2e+05|
System.out.printf("|%10.2g|%n", 123456.789);

// Floating-point number
// Prints |1.2e+05   |
System.out.printf("|%-10.2g|%n", 123456.789);

// Floating-point number
// Prints |0001.2e+05|
System.out.printf("|%010.2g|%n", 123456.789);

// ----
//  %a
// ----

// Floating-point in hexadecimal format
// Prints 0x1.4b3fd5942cd96p-20
System.out.printf("%a%n", 0.000001234);

// ====================
//  Displaying strings
// ====================

// ----
//  %s
// ----

// Prints the whole string
// Prints |Hello World|
System.out.printf("|%s|%n", "Hello World");

// Specify field length
// Prints |                   Hello World|
System.out.printf("|%30s|%n", "Hello World");

// Left justify text
// Prints |Hello World                   |
System.out.printf("|-30s|%n", "Hello World");

// Specify maximum number of characters
// Prints |Hello|
System.out.printf("|%.5s|%n", "Hello World");
```

```
    // Field width and maximum number of characters
    // Prints |                       Hello|
    System.out.printf("|%30.5s|%n", "Hello World");

    // ----
    //  %h
    // ----

    // Prints The String object Welcome to COMP3021 is at hash code dde66912
    String str = "Welcome to COMP3021";
    System.out.printf("The String object %s is at hash code %h%n", str, str);

    // ----
    //  %b
    // ----

    // Any type
    // Prints true
    System.out.printf("%b%n", true);

    // Any type
    // Prints TRUE
    System.out.printf("%B%n", true);

    // ----
    //  %c
    // ----

    // Any type
    // Prints a
    System.out.printf("%c%n", 'a');

    // Any type
    // Prints A
    System.out.printf("%C%n", 'a');
  }
}
```

Adopted from the following source:

- https://www.cs.colostate.edu/~cs160/.Spring16/resources/Java_printf_method_quick_reference.pdf
- http://www.homeandlearn.co.uk/java/java_formatted_strings.html
- http://cs.middlesexcc.edu/~schatz/csc161/handouts/output.formatting.html
- http://www.java2s.com/Tutorials/Java/Java_Format/0110__Java_Format_Number.htm