Java `printf()` Method Quick Reference

System.out.printf("format-string", arg1, arg2, ...);

**Format String:**

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  ( square brackets denote optional parameters )

**Flags:**

-  : left-justify ( default is to right-justify )
+  : output a plus (+) or minus (-) sign for a numerical value
0  : forces numerical values to be zero-padded ( default is blank padding )
,  : comma grouping separator (for numbers > 1000)
   : space will display a minus sign if the number is negative or a space if it is positive

**Width:**

Specifies the field 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-Characters:**

d  : decimal integer  [byte, short, int, long]
f  : floating-point number  [float, double]
c  : character  Capital C will uppercase the letter
s  : String  Capital S will uppercase all the letters in the string
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

**Examples:**

```java
System.out.printf("Total is: $%,.2f%n", dblTotal);
System.out.printf("Total: %-10.2f: ", dblTotal);
System.out.printf("% 4d", intValue);
System.out.printf("%20.10s
", stringVal);
String s = "Hello World";
System.out.printf("The String object %s is at hash code %h%n", s, s);
```

**String class `format()` method:**

You can build a formatted String and assign it to a variable using the static format method in the String class. The use of a format string and argument list is identical to its use in the printf method. The format method returns a reference to a String. Example:

```java
String grandTotal = String.format("Grand Total: %.2f", dblTotal);
```