FORMATTING TEXT WITH PRINTF

zyBook 2.6, zyBook 2.7, Java API

@ Dr. Jessica Young Schmidt and NCSU Computer Science Faculty

FORMATTING TEXT WITH PRINTF

A convenience method to write a formatted string to this output stream using the specified format string and arguments.

```
System.out.printf("format string", parameters);
```

A format string can contain placeholders to insert parameters:

- %d → integer
- %f → real number
- %s → string
- %c → char

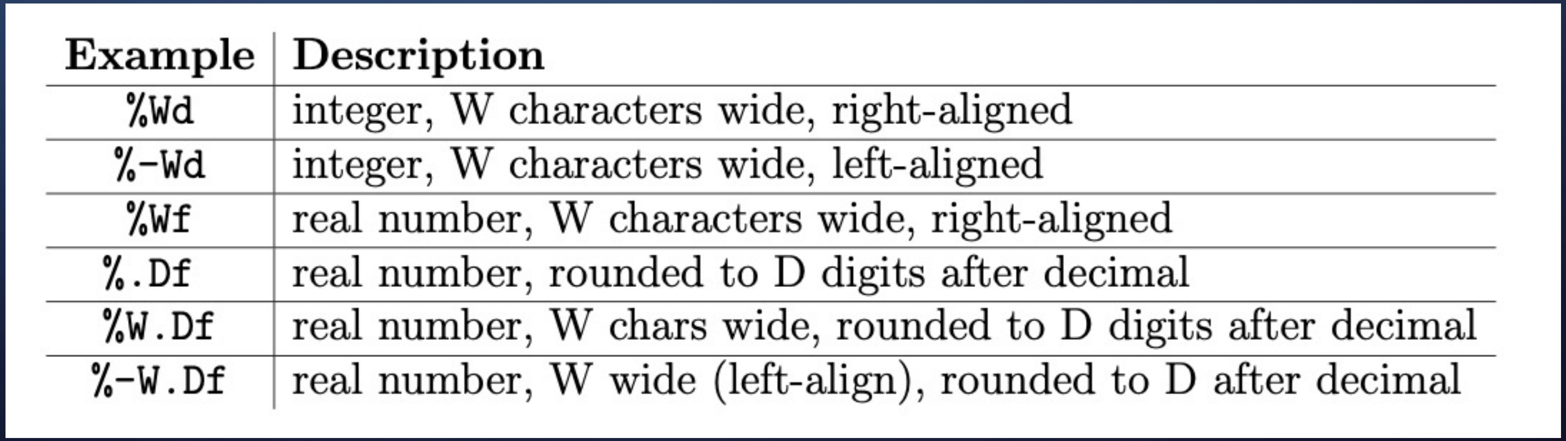
```
System.out.println("You are currently enrolled in CSC" + course + ".");
OR
System.out.printf("You are currently enrolled in CSC%d.\n", course);
```

These placeholders are used instead of + concatenation

The format specifiers for general, character, and numeric types have the following syntax:

"%[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. (‘-’ left-aligned the result; otherwise right-aligned)

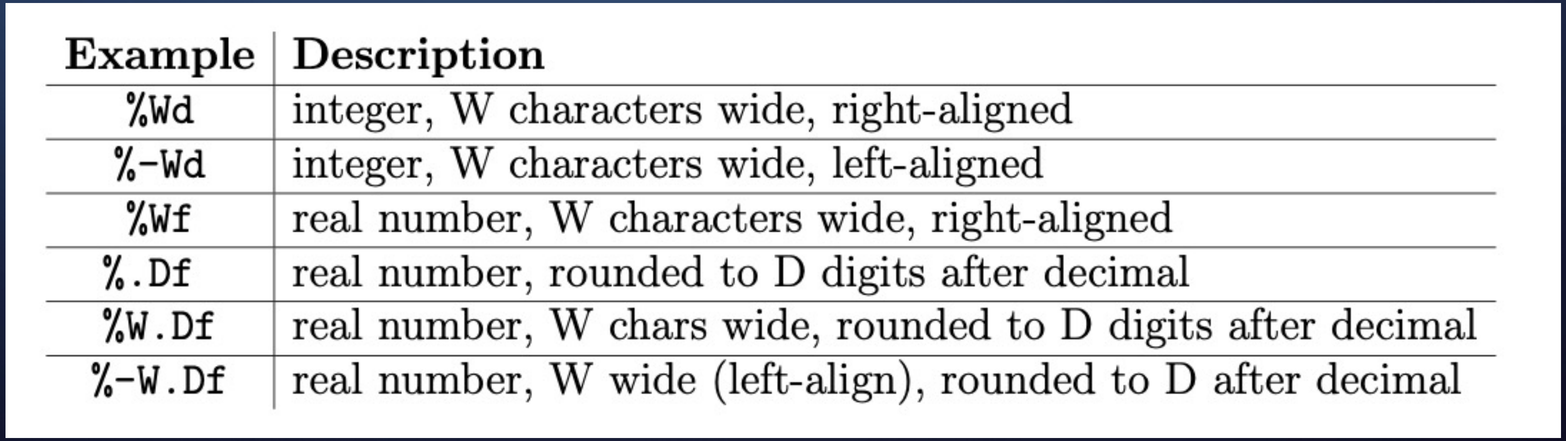


Example	Description
%Wd	integer, W characters wide, right-aligned
%-Wd	integer, W characters wide, left-aligned
%Wf	real number, W characters wide, right-aligned
%.Df	real number, rounded to D digits after decimal
%W.Df	real number, W chars wide, rounded to D digits after decimal
%-W.Df	real number, W wide (left-align), rounded to D after decimal

The format specifiers for general, character, and numeric types have the following syntax:

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

- The optional width is a positive 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.



Example	Description
`%Wd`	integer, W characters wide, right-aligned
`%-Wd`	integer, W characters wide, left-aligned
`%Wf`	real number, W characters wide, right-aligned
`%.Df`	real number, rounded to D digits after decimal
`%W.Df`	real number, W chars wide, rounded to D digits after decimal
`%-W.Df`	real number, W wide (left-align), rounded to D after decimal

```java
public class ReceiptFormatted {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax and 20% tip
        int subtotal = 38 + 40 + 30;
        double tax = subtotal * .08;
        double tip = subtotal * .2;
        double total = subtotal + tax + tip;

        System.out.println("\nWithout Formatting...");
        System.out.println("Subtotal: " + subtotal);
        System.out.println("Tax: " + tax);
        System.out.println("Tip: " + tip);
        System.out.println("Total: " + total);

        System.out.println("\nWith Formatting...");
        System.out.printf("%-13s $%7.2f%n", "Subtotal", (double) subtotal);
        System.out.printf("%-13s $%7.2f%n", "Tax", tax);
        System.out.printf("%-13s $%7.2f%n", "Tip", tip);
        System.out.printf("%-13s $%7.2f%n", "Total", total);
    }
}
```

```
$ javac -d bin -cp bin src/ReceiptFormatted.java

$ java -cp bin ReceiptFormatted

Without Formatting...
Subtotal: 108
Tax: 8.64
Tip: 21.6
Total: 138.24

With Formatting...
Subtotal      $  108.00
Tax           $    8.64
Tip           $   21.60
Total         $  138.24
```