Java With Us

Displaying text using printf() method

In addition to the print() and println() methods, the printf () method is also quite frequently used for displaying text that needs to be formatted. Formatting here neither refers to the font size or the font colour but is related to how data is to be displayed. For example: the number of decimal digits to be displayed, the alignment of the String and other similar things are related to formatting data. The printf () method requires a format String and a set of other arguments whose number depends on the format String. The format String is similar to a normal String with the exception that it contains placeholders where appropriate data items may be placed. These place holders state the type of data that is going to be placed and also include any other necessary formatting details such as the number of decimal digits to be displayed, the alignment of the text and so on. For example, consider the following two equivalent statements assuming that the variable str holds the String "Java":

```
System.out.println ("Welcome to "+str);
System.out.printf ("Welcome to %s", str);
```

Both of the above statements print the String "Welcome to Java". However, the approach taken varies. In the first statement, we have used concatenation while in the second case, we have achieved the same result by inserting a placeholder for a String (%s) and then stating the String that needs to be placed in that defined area. We may specify as many placeholders as we wish and then state their values as arguments in the same order in which they need to be placed in the empty placeholders. The first parameter in the printf() method call is known as the format string. And the '%s' and any other similar entities are known as format specifiers. The digit that follows the % sign (in this case, an s) is known as the conversion character. Format specifiers are provided with additional information depending on the formatting that we wish to apply. Different format specifiers are used to print different types of text. The table below summarises the most commonly used.

Format specifier	Description
%d	Displays a decimal (base 10) integer
%f	Display a floating point value in decimal format
%e or %E	Display a floating point number in exponential notation
%c or %C	Display characters
%s or %S	Display Strings
%b or %B	Display boolean values
%%	Display a % sign

Therefore, to print a short, byte, int or long data item, we use the %d format specifier. To print float and double values, we use either %f or %e depending on the requirement of output. To print boolean, char and String data items, we use %b, %c and %s specifiers respectively.

The following code fragment shows the usage of these specifiers.

```
int day = 9;
String month = "March";
int year = 1993;
char grade='A';
int marks=99;
System.out.format("Born on %dth %s, %d\n", day,month,year);
System.out.format("Secured %d%% marks and %c grade", marks,grade);
```

The output would be:

```
Born on 9th March, 1993
Secured 99% marks and A grade
```

Note how %% was used to display a % sign. If we had simply written % to display a present sign, it would have resulted in a compilation error. This is because a % sign needs to be followed by a valid conversion character.

When we need to display data in the form of a table, the task might become cumbersome if we manually insert the appropriate number of spaces to align the rows properly. For this purpose, we can use the formatting capabilities provided by this method. The length of the area in which the text is to be printed is known as the field width. We can specify the field width by inserting a number between the % sign and the conversion character. If the data to be printed is smaller, then the appropriate number of spaces are inserted and the data is right justified. To left justify the data, we place a negative sign before the number. Field width can be specified for printing any data type.

```
String name1="Sai";
String name2="Gautham";
int marks1=100;
int marks2=99;
System.out.format("%-10s - %4d\n",name1,marks1);
System.out.format("%-10s - %4d\n",name2,marks2);
```

The output would be

```
Sai        -  100
Gautham    -   99
```

Note how the Strings were left justified and the integers right justified.

We can also specify the precision of the data to be printed. Precision defines the number of decimal digits that are to be printed when we use it with a floating point number. Precision is specified by placing a dot (.) followed by a number indicating the required precision.

```
System.out.format("%.3f",34.789456);
```

The output would be

```
34.789
```

Flags call also be provided in the format String to add further formatting. In addition, we can format dates and times.

Next : Java Comments
Prev : Displaying text using print and println