Output

There are more options for printing output to the console in Java than there are in C, but we are going to start with a method that is very similar between the two languages. They both have a way to print a string of text with an optional list of expressions that can be inserted into that text.

For example, we could print “John is 10 years old” by calling the appropriate function with that string. We could also use the string “John is %d years old” and use the variable ageOfJohn. The symbol “%d” is called a **conversion specifier**, and it tells the function how to use the variable.

These are the most common conversion specifiers to use:

|  |  |  |
| --- | --- | --- |
| **Specifier** | **Variable Type Used** | **Example** |
| %d | Signed integer (int, long, short) | "John is %d years old", 10  John is 10 years old |
| %f | Floating point number (float, double) | "His GPA is %f", 3.14f  His GPA is 3.780000 |
| %c | Single character (char) | "His grade is %c", 'B'  His grade is B |
| %s | String of characters (String in Java, char\* in C) | "He said %s", "Hello"  He said Hello |

The conversion specifiers determine the type of the data to display, but there are additional things we can do to control how it is displayed. We have **flags** that can control things like justification and padding, a **width** value which sets the minimum number of characters to output, and a **precision** value that controls how many digits to print after the decimal point (this obviously only applies to floating point numbers).

Including all three of these controls would look like this: **%[flags][width][.precision]type**

Some useful flags:

|  |  |
| --- | --- |
| **'+'** | Includes sign, whether positive or negative |
| **'-'** | Left-justified |
| **' ' *(space)*** | Non-negative values begin with a space character (' '). This flag is only useful for signed conversion results (%d and %f) |
| **'#'** | With %o: values are pre-pended with a zero ('0')  With %x or %X: values are pre-pended with the prefix "0x" or "0X" |
| **','** | Includes locale-specific grouping characters |

|  |  |
| --- | --- |
| **Code** | **Output** |
| int age = 21;  "|%d|%5d|%-5d", age, age, age | |21| 21|21 | |
| long number = 461012;  "|%d|", number  "|%08d|", number  "|%+8d|", number  "|%,8d|", number  "|%+,8d|", number | |461012|  |00461012|  | +461012|  | 461,012|  |+461,012| |
| float value = 165.89f;  "|%f|%10.3f|%10.3e|", value, value,  value | |165.889999| 165.890| 1.659e+02| |

In both Java and C we print to the console by passing the string and the variables into a function. In C we use printf() and in Java we use System.out.printf(). (In C, to be able to use printf() we need to add a line at the top of the file: #include <stdio.h>)

There is an additional difference between C and Java, which is how we specify a **newline** character. This is how we tell the console output to start a new line. In C we use **\n**, and in Java we use **%n**. The C-style newline character will work in Java, but %n will be translated to the correct newline character for the operating system you are using.

|  |  |
| --- | --- |
| **C** | **Java** |
| printf("John is %d\n", age); | System.out.printf("John is %d%n", age); |
| printf("Item 1\nItem 2"); | System.out.printf("Item 1%nItem 2"); |
| printf("My GPA is %.1f\n", 3.2); | System.out.printf("My GPA is %.1f%n", 3.2); |

# Java Output Methods

Java has additional functions for output that can be easier for us to use, System.out.print() and System.out.println(). The only difference between these two functions is that System.out.println() adds a newline character at the end of the output. For both of these methods we pass in a single value, function, or string.

|  |  |
| --- | --- |
| **Code** | **Output** |
| int foo = 5;  System.out.print(foo); | 5 |
| System.out.print(5);  System.out.print(8); | 58 |
| System.out.println(5);  System.out.println(8); | 5  8 |
| System.out.println("Hello World"); | Hello World |

We can also combine strings and variables by using the + operator.

|  |  |
| --- | --- |
| **Code** | **Output** |
| System.out.print("Hello " + "World"); | Hello World |
| float grade = 3.2;  System.out.print("My grade is " + grade); | My grade is 3.2 |

Input

The methods for getting input from the console in Java and C are different, but they both allow us to wait for a user to type something and then get what they typed after they press the Enter key.

# Input in C

Reading input from the console in C uses a function called scanf() and the same format specifiers that we used for formatting our output. The format string tells scanf() how to interpret the user’s input, and the variable matching the format specifier is where the input will be stored.

In order to use scanf(), we need to make sure that the line #include <stdio.h> is somewhere near the top of our file.

For example, this is how we would read in a single integer:

int integerInput;

scanf("%d", &integerInput);

Note the & symbol before the name of the variable: we will explain the reason for this in a later chapter on pointers. For now, all you need to know is that this allows the scanf() function to write data to our variable.

If we tell the user to type two floats we would read them like this:

float floatInput1, floatInput2;

scanf("%f%f", &floatInput1, &floatInput2);

If we want to read in the entire line of input as a string, we have to declare a variable which is an array of characters. We will explain arrays and strings in a later chapter: for now, just remember to make sure that the number you use to declare the length of the array is long enough for all the characters the user might type.

char stringInput[100];

scanf("%s", stringInput);

Note that the array of characters does not need the & symbol.

# Input in Java

In Java we can use something called a **Scanner** to get input from the user. Before we can use it, we need to include the following line above our code:

import java.util.Scanner;

This tells the compiler what the Scanner is. To use it, we first have to create a variable that uses Scanner as its type.

Scanner consoleInput = new Scanner(System.in);

The code new Scanner(System.in) creates a Scanner object and tells it to get its information from the user’s input. This object is then stored in our variable named consoleInput. This way of creating variables is different from what we’ve done so far, but it will be explained in much more depth in a later chapter that discusses classes.

Now that we have our Scanner object, we can use it to get the user’s entire line of input as a string, or we can use it to get an int, a float, or a double specifically.

To get the entire input as a String we use the nextLine() function.

String input = consoleInput.nextLine();

To get an int, float, or double, we can use the nextInt(), nextFloat(), and nextDouble() functions.

int age = consoleInput.nextInt();

float gpa = consoleInput.nextFloat();

double decimalNumber = consoleInput.nextDouble();