Week 1: Computer Science 1

Output and Math

Comments

Comments are used to explain code and make it easier to understand. They can also be used to prevent execution when testing alternative code.

The compiler ignores comments when compiling the code.

For single line comments use `//`

For multi-line comments use `/*` and `*/`

```
public class Main{
    public static void main(String[] args){
        // This is a single line comment
        System.out.println("Hello World");
        /*
        This is a
        multi-line
        comment
        */
    }
}
```

Math

Operator	Description	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y

Let's go through how to use each of these operators. We are going to use the print statement to print the result of each operation. Next week we will learn how to use variables to perform some of the same operations.

+

```
public class Main{
    public static void main(String[] args){
        // print for single line
        System.out.print("1 + 1 = ");
        System.out.println(1 + 1); // Addition
    }
}
```

Output

```
1 + 1 = 2
```

Let's use one line of code to produce the same output.

You would assume this would be the correct code:

```
public class Main{
    public static void main(String[] args){
        System.out.println("1 + 1 = " + 1 + 1);
    }
}
```

Output

```
1 + 1 = 11
```

Why is this the output?

The reason is that the `+` operator is used for both addition and concatenation. The compiler reads the code from left to right. It sees the first `+` and assumes that the rest of the code is concatenation.

Concatenation is the process of combining two strings together. We will discuss strings in more detail next week.

We want the compiler to perform the addition. We can do this by using parentheses.

```
public class Main{
    public static void main(String[] args){
        System.out.println("1 + 1 = " + (1 + 1));
    }
}
```

Output

```
1 + 1 = 2
```

This example shows **Polymorphism**. Polymorphism is the ability of an object to take on many forms. In this case the `+` operator is used for both addition and concatenation.

- number + number will perform addition. 1 + 1 = 2
- string + number will perform concatenation. "1" + 1 = "11"
- number + string will perform concatenation. 1 + "1" = "11"
- string + string will perform concatenation. "1" + "1" = "11"

What will this line of code produce?

```
System.out.println(3 + 4 + "bye" + 1 + 2 + 8 + 9);
```

Output

```
7bye1289
```

- 3 + 4 performs addition. 3 + 4 = 7
- 7 + "bye" performs concatenation. 7 + "bye" = "7bye"
- "7bye" + 1 performs concatenation. "7bye" + 1 = "7bye1"
- "7bye1" + 2 performs concatenation. "7bye1" + 2 = "7bye12"
- "7bye12" + 8 performs concatenation. "7bye12" + 8 = "7bye128"
- "7bye128" + 9 performs concatenation. "7bye128" + 9 = "7bye1289"

Let's try a few more examples.

```
System.out.println(":)" + (8-3));

System.out.println(1 + 2 + 3 + "hi" + 4 + 5);

System.out.println(1 + 2 + 3 + "hi" + (4 + 5));

System.out.println(3 * 2 - (1+5));
```

Output

```
:)5
6hi45
6hi9
0
```

Formatting Output

We already reviewed how to use the 'println' and 'print' statements to output text to the console. We can also use the 'printf' statement to format the output.

System.out.println() is one of the methods to print output to the screen. It prints the string inside the parentheses to the screen and then moves the cursor to the next line.

System.out.print() is another method to print output to the screen. It prints the string inside the parentheses to the screen and leaves the cursor on the same line.

System.out.printf() is another method to print output to the screen. It prints the string inside the parentheses to the screen and leaves the cursor on the same line. It also allows you to format the output.

Java uses the `\` character to indicate that the next character is a special character.

Character	Description
\n	New Line
\t	Tab
\b	Backspace
\'	Single Quote
\"	Double Quote
\	Backslash

Let's look at an example of how to use the `printf` statement.

```
public class Main{
    public static void main(String[] args){
        System.out.printf("- - - \n2\t" + (2 * 2));
    }
}
```

Output

```
- - -
2       4
```

Notice the `\n` and `\t` characters. `\n` is used to move the cursor to the next line. `\t` is used to insert a tab.

You may also need to generate tables in your programs. You can use the `printf` statement to format the output. printf stands for **print formatted**.

```
System.out.printf(" ", , ,);
```

- Inside `" "` place parameters/specifiers to format the output.
- After the first comma place the values to be formatted. Each separate value should be separated by a comma and have a matching specification.
- Format specifiers begin with a `%` character. The format specifiers are used to specify the type of data to be formatted.

Use the following format order:
%[flags][width][.precision]conversion-character

- The flags, width, and precision are optional. (So you need at least the % and conversion-character!)

A conversion-character is required and needs to match the data type in the second half of the print statement.

Specifier	Description
%c	Character
%C	Uppercase Character
%d	Decimal Integer (byte, short, int, long)
%f	Floating point number (float, double)
%s	String (any object with a toString method)
%S	Uppercase String
%b	Boolean
%n	New Line
%e	Scientific Notation (float, double)

Examples

```
System.out.printf("%d", 123); // 123
System.out.printf("%f", 8.7)  // 8.700000
// There are a lot of decimal places. Precision!
System.out.printf("%C", 'a'); // A
System.out.printf("%s", "Hello"); // Hello
```

```
System.out.printf("%S%S%n%s%d, "Employee", "ID", "John Smith", 1234567);
```

Output

```
EMPLOYEEID
John Smith1234567
```

Precision is the number of digits after the decimal point or the number of characters in a String. It is specified by a `.` followed by an integer.

With Precision

```
System.out.printf("%.2f", 8.7); // 8.70
System.out.printf("%.7s", "Hello World"); // Hello W
```

Without Precision

```
System.out.printf("%f", 8.7); // 8.700000
System.out.printf("%s", "Hello World"); // Hello World
```

Width is the minimum number of characters to be written as output. It is specified by an integer (whole number).

With Width

```
System.out.printf("%S%3S%n%s%8d", "Employee", "ID", "John Smith", 1234567);
```

Output with width

```
EMPLOYEE  ID
John Smith 1234567
```

- Notice that ID has a width of 3. Since ID is only 2 characters, the output is padded with a space. Also Notice that 1234567 has a width of 8. Since 1234567 is only 7 characters, the output is padded with a space.

Flags are used to format the output. They are specified by a character.

Flag	Description
-	Left-justify within the given field width; Right justification is the default
+	Displays a + in front of numbers
,	For decimal and scientific notation, the output will have the thousands separator. Example 1,000

```
System.out.printf("%-7s%-10f%-8s", "Name", 35.234, "hi"); // Name   35.234000 hi
System.out.printf(“%+d", 15); // +15
System.out.printf(“%+d", -15); // -15
System.out.printf(“%,d", 1000); // 1,000
```

Examples

Try to replicated the following output using the `printf` statement.

```
Some Random Numbers
8   2.45   90
```

- "Some Random Numbers" is just a string.
- 8, 2.45, and 90 have three spaces between them

```
Name   Job      ID
Sally  Writer   12345
```

- Three spaces between "Name" and "Job".
- Five spaces between "Job" and "ID".

Solutions

```
Some Random Numbers
8   2.45   90
```

```
System.out.println("Some Random Numbers");
System.out.printf("%-4d%-7.2f%d", 8, 2.45, 90);
```

```
Name   Job      ID
Sally  Writer   12345
```

```
System.out.printf("%-7s%-8s%s", "Name", "Job", "ID");
System.out.printf("%-7s%-8s%d", "Sally", "Writer", 12345);
```

Try to replicate the following output using the `printf` statement on your computer or with paper. This time write the full program and run your Java code. You can use the `println` statement to print the title, "Info About Earl the Cat".

```
Info About Earl the Cat
Name    Age    Weight(lbs)
Earl    8      15.50
```

- There are 4 between "Name" and "Age" and 4 between "Age" and "Weight(lbs)".

```
public class Main{

    public static void main(String[] args){
        System.out.println("Info About Earl the Cat");
        System.out.printf("%-8s%-7s%s", "Name", "Age", "Weight(lbs)");
        System.out.printf("%-8s%-7d%.2f", "Earl", 8, 15.5);
    }
}
```