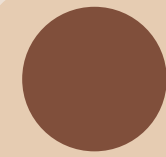


Topic 1.4: Formatting Output

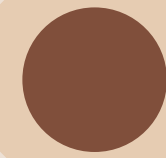
Learning Goals (Week 1):



Identify data types based on value



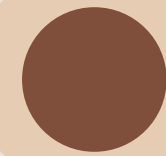
Map variables to the current values



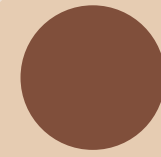
Perform basic operations on variables



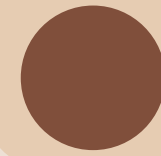
Create and use Java and user-defined methods



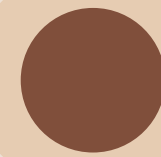
Format Printed Output



Obtain and process user input from the console



Use booleans, conditionals, and compound conditionals correctly



Select and implement different types of loops depending on scenario



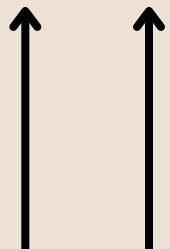
Use special String and Math operations



Successfully implement and manipulate java arrays

- Can use **printf()** or **format()** instead of **print()** or **println()**

```
System.out.printf("Hello %s!%n", "World");
```



%s means
place string
here (first
string after ,)

%n means new
line when used
in printf
(similar to \n)

```
System.out.printf("Casting %f to int gives %d %n", 23.8, (int)23.8 );
```

- The first parameter is a String that indicates exactly how you want the data printed
- The **bold** codes that start with % are where the data goes (not %n though, that's new line)
- There can be any number of other parameters
 - These supply the actual data to print (in red)

Common Codes for Formatting

- **%d** – print a **d**ecimal integer here (base 10 integer)
 - **%6d** – use at least **6 characters** to do that
- **%f** – print a **f**loating-point value here
 - **%6f** – use at least **6 characters** to do that
 - **%6.2f** – with exactly **2** of them **after the decimal point**
- **%s** – print a **S**tring here
- **%n** – print a **n**ewline (`\n` character) here

The best way to see if something works (and learn it) is to try it out yourself!

Remember:

There must be one additional parameter (after the String) for each code used (except `%n`), and it must be the correct type

Pause & Practice

1. Decimal Integer Formatting

- Write a Java program that uses `printf` to print an integer `123` using the `%d` format specifier. Then, print the same integer using `%6d` format specifier to ensure it occupies at least 6 characters.

2. Floating-Point Value Formatting

- Create a Java program to print a floating-point number `123.4567` using three different format specifiers: `%f`, `%6f`, and `%6.2f`. The first should print the number as it is, the second should ensure the printed value is at least 6 characters long, and the third should format the number to occupy at least 6 characters with exactly 2 digits after the decimal point.

3. String Formatting

- Use `printf` in a Java program to print the string `"Hello World"` using the `%s` format specifier. Experiment with different string lengths and observe how the output changes.

4. Combining Different Types

- Write a Java program that combines different format specifiers in a single `printf` statement. Print an integer, a floating-point number, and a string on the same line using `%d`, `%6.2f`, and `%s` respectively, followed by a newline using `%n`. For example, print `42`, `3.14159`, and `"Java"` in this format.

Pause & Practice (answers)

1. Decimal Integer Formatting

```
System.out.printf("%d%n%6d%n", 123, 123);
```

2. Floating-Point Value Formatting

```
System.out.printf("%f%n%6f%n%6.2f%n", 123.4567, 123.4567, 123.4567);
```

3. String Formatting

```
System.out.printf("%s%n", "Hello World");
```

4. Combining Different Types

```
System.out.printf("%d %6.2f %s%n", 42, 3.14159, "Java");
```