Java Basics: Printing Data

System.out

- When creating applications for desktop computers, Java defines an object called `System.out` which corresponds to the computer's console
- Anything printed to `System.out` gets displayed on the computer's screen
- Printing to the screen of an Android device is a bit more complicated, so for this MOOC we have defined an object called `out` that will act similar to `System.out`

Printing to the Screen

```
out.println("Whatever you want to print");
```

- `out` is an object for sending output to a text box on the Android screen
- `println` is a method to print whatever is in the parentheses to the screen

- Let's look at an earlier example one more time…

A Java program

```
public class Logic implements LogicInterface {
    ...

    public void process() {
        Log.d(TAG, "prcoess()");
        out.println("Hello World!");
    }
}
```

A Java program

```
public class Logic implements LogicInterface {
    ...

    public void process() {
        Log.d(TAG, "prcoess()");
        out.println("Hello World!");
    }
}
```

- Output:

```
Hello World!
```

println

- A statement that prints a line of output on the screen
  - pronounced "print-linn"

- Two ways to use **`println`**:
  - **`out.println("text");`**

    Prints the given message on the screen
  - **`out.println();`**

    Prints a blank line on the screen

Strings

- A *string* is a sequence of characters that can be printed
  - Sometimes also called a *string literal*
  - Starts and ends with a double quote character: `"`
    - The quotes do not appear in the output
  - Examples:

```
"Hello world!"
"This is a string too.  @$%&^>:?[] It's funky!"
```

Strings

- There are some restrictions with strings:

May not span multiple lines

```
"This is not a legal string
 since it spans two lines"
```

May not contain a `"` character (the `'` character is okay)

```
"This string is not "legal" either."
```

```
"This one is 'okay' though."
```

Escape Characters

- How would you print:

```
"Java" refers to a language.
```

- The compiler needs to be told that the quotation marks (") in this case do not signal the start or end of a string, but instead are to be printed as a part of the string

```
out.println("\"Java\" refers to a language.");
```

Escape sequences

- An *escape sequence* is a combination of characters used to represent certain special characters within a string

  `\"` quotation mark character
  `\\` backslash character
  `\t` tab character
  `\n` new line character

- Each escape sequence is treated as a single character even though it is written with two symbols
  - Example:

```
out.println("I'm\tOK\nYou're \"OK\".");
```

Escape sequences

- An *escape sequence* is a combination of characters used to represent certain special characters within a string

`\"`	quotation mark character
`\\`	backslash character
`\t`	tab character
`\n`	new line character

- Each escape sequence is treated as a single character even though it is written with two symbols
  - Example:

```
out.println("I'm\tOK\nYou're \"OK\".");
```

  - Output:

```
I'm     OK
You're "OK".
```

Printing a variable's value

- Use + to print a string and a variable's value

```
double average = (52 + 73 + 18 + 47) / 4.0;
out.println("The average was " + average);

int numCars = 13 + 21 + 14 + 18 + 15;
out.println("There are " + numCars +
                    " cars in the parking lot.");
```

- Output:

```
The average was 47.5
There are 81 cars in the parking lot.
```

Printing a variable's value

- The + sign in this situation is the string concatenation operator; it is what happens when you "*add*" two strings
- It concatenates two strings together to create a larger string
- In the case of printing a variable, the *value* of the variable is converted into a string so that it can be concatenated

```
double average = (52 + 73 + 18 + 47) / 4.0;
out.println("The average was " + average);
```

`print` command

- Recall: `println` prints a full line of output and then advances to a new line

- Another command named `print` prints the given output without moving to the next line
  - This allows you to print partial messages on the same line

`print` command

- Example:

```
out.print("Hello.");
out.println("It's me,");
out.print("I was wondering ");
out.println("if after all these years");
out.println("you'd like to meet.");
```

Output:

```
Hello.It's me,
I was wondering if after all these years
you'd like to meet.
```

printf

- An advanced command for printing formatted text

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

A *format string* contains *placeholders* to be replaced by parameters:

`%d`	an integer
`%f`	a floating point number
`%s`	a string

- Example:

```
int x = 3;
int y = 2;
out.printf("(%d, %d)\n", x, y);   // (3, 2)
```

`printf` cont'd

- A placeholder can specify the parameter's *width* or *precision*:

`%8d`	an integer, 8 characters wide, right-aligned
`%-8d`	an integer, 8 characters wide, left-aligned
`%.4f`	a real number, 4 characters after decimal
`%6.2f`	a real number, 6 characters wide, 2 after decimal (rounded)

  - Note that width specifications set a minimum width (it can overflow)