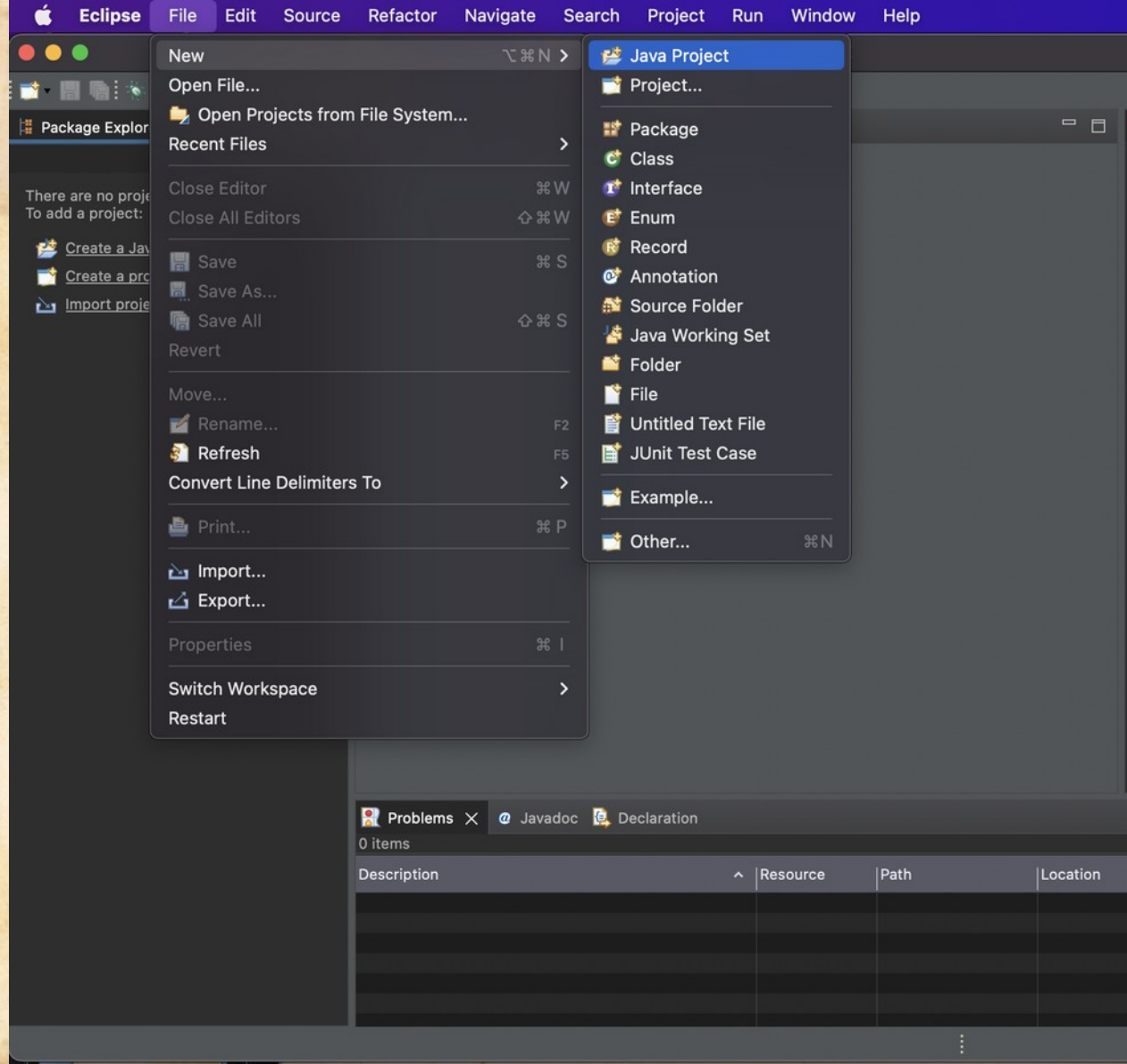
Java Basics

Basic Text Output

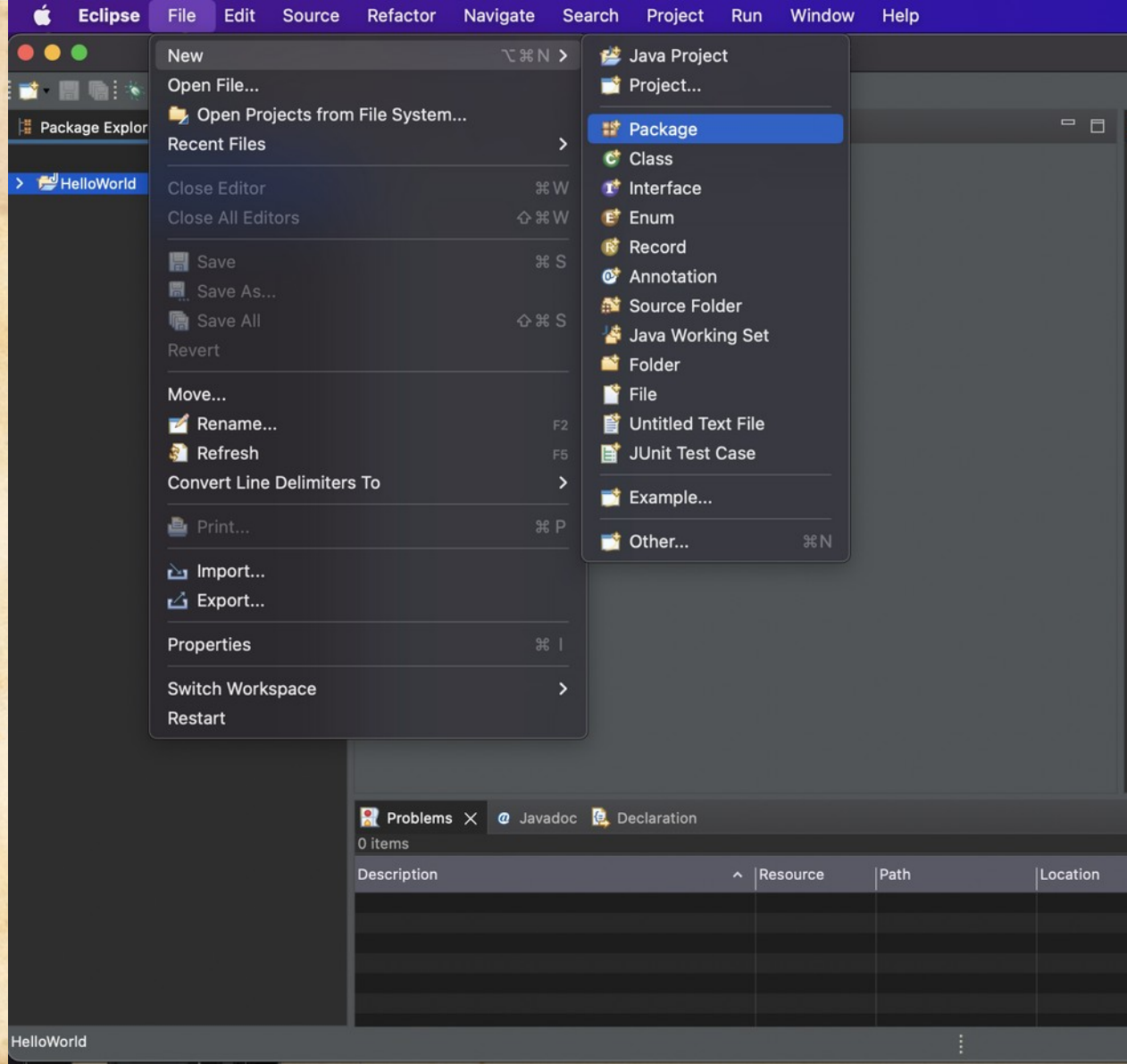
Lecture Contents

- Review of how we got to “`Hello World!`”
- Basic Output

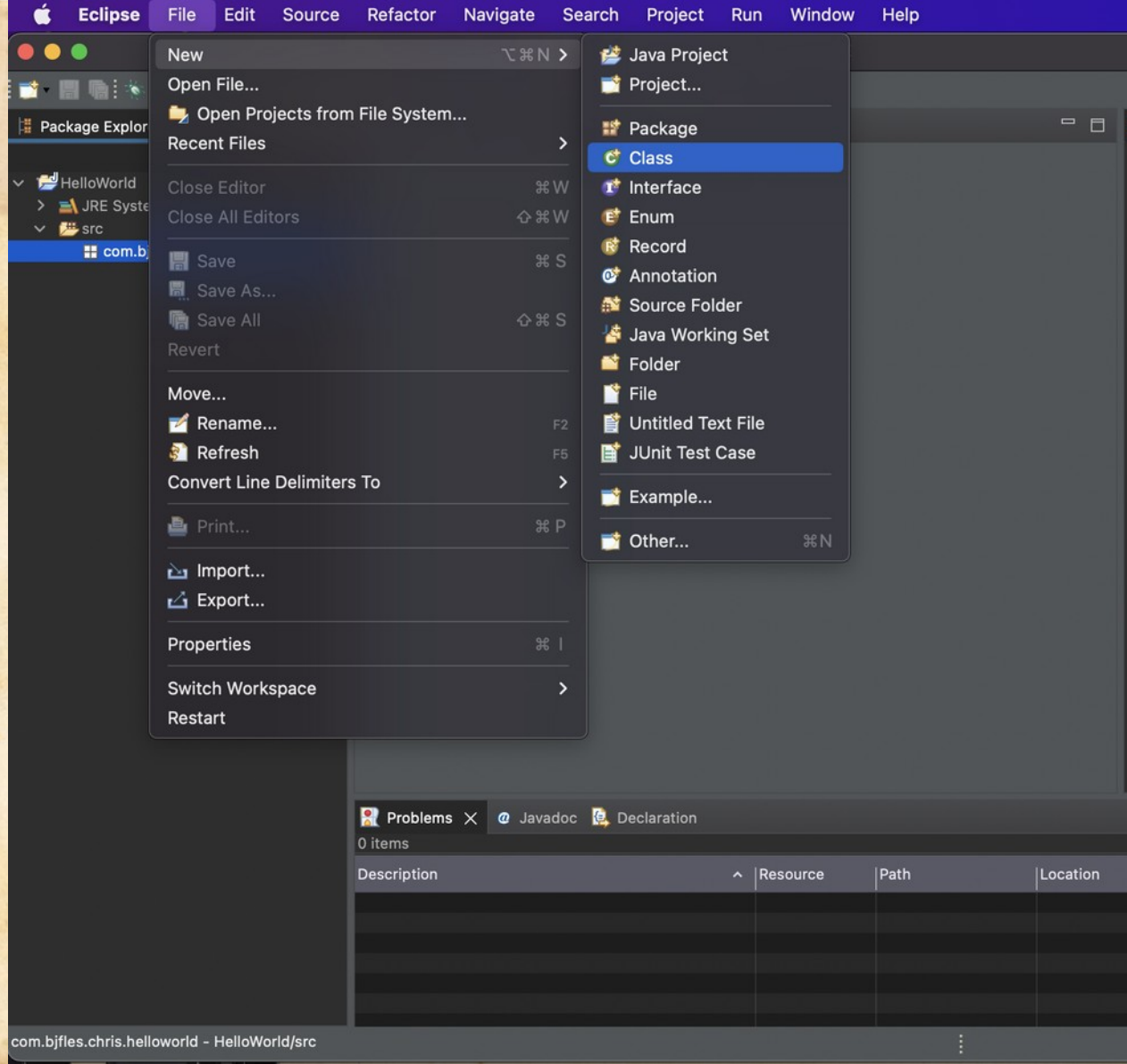
- For this course, it is okay to keep all files in the same Java project.



- At minimum, create a new package for each course topic.
- Package names use a domain name in reverse order:
  - com.bjfles.

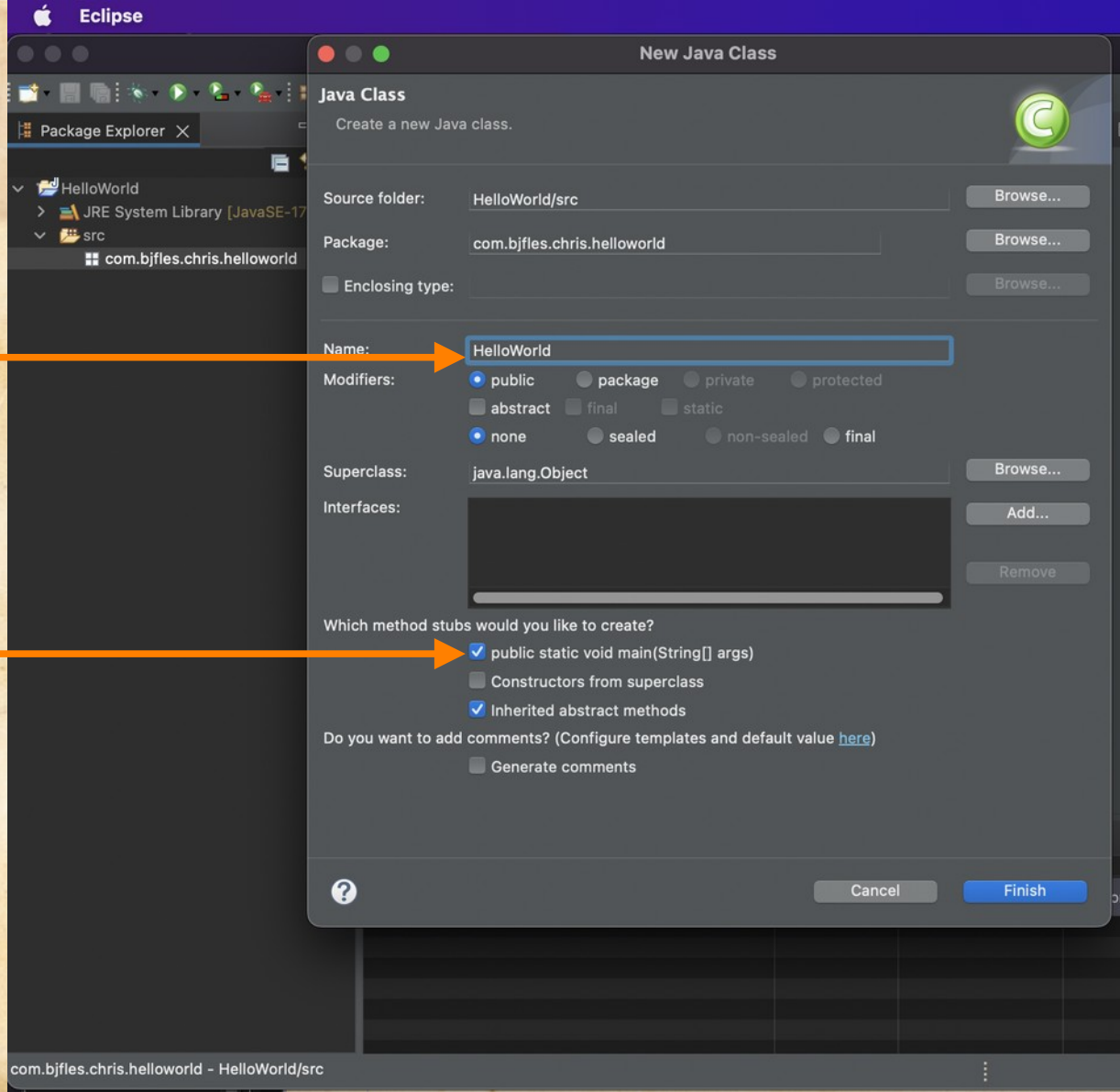


- Create a new class for each new program.
- Class names use `CamelCase`.
- For today's task, perhaps call the class:
  - `BasicOutput`

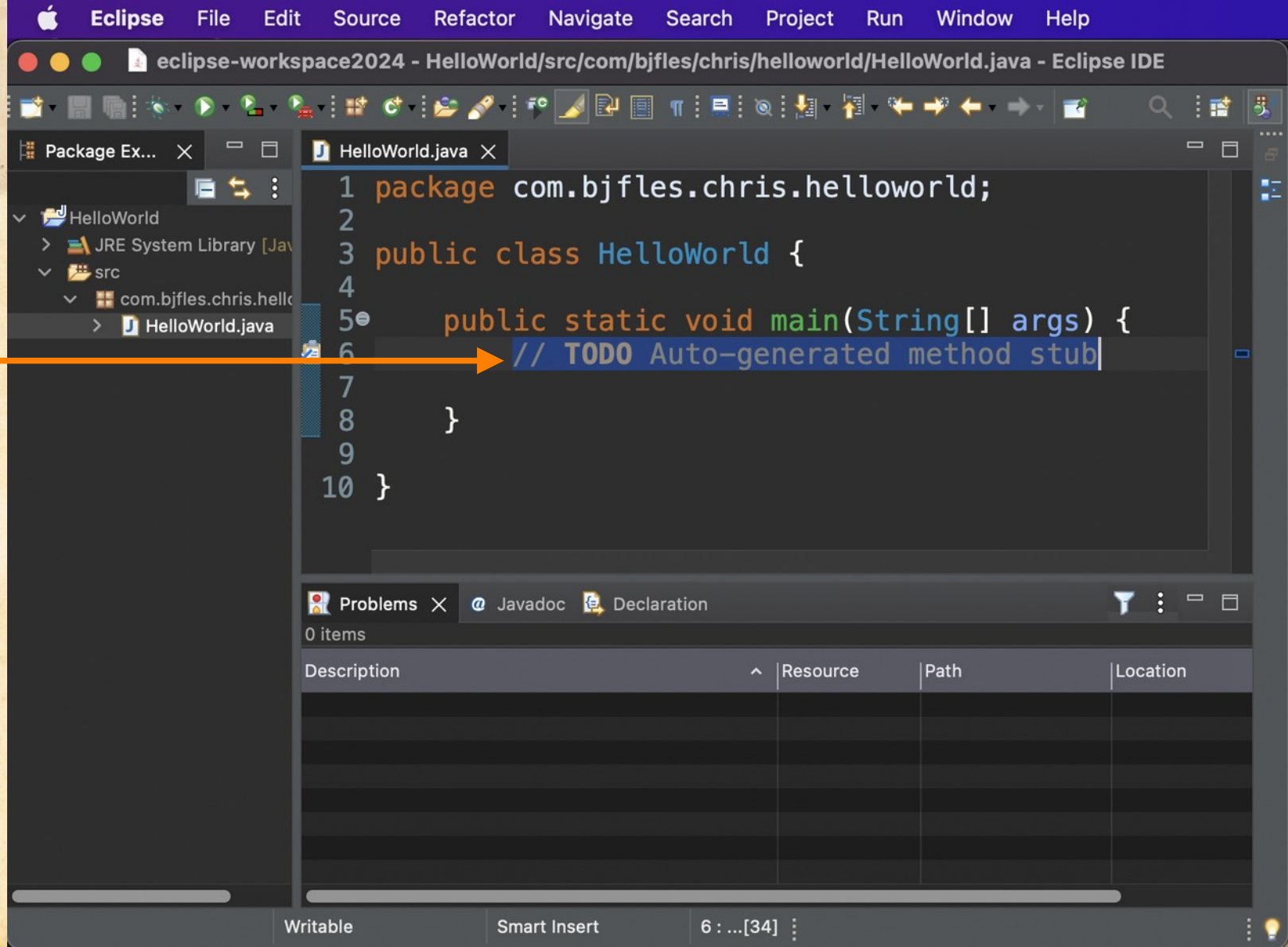


Change this to “BasicOutput” for today’s assignment.

Remember to check this box to add template code for the **`main`** method, (which is where the Java program will start running).

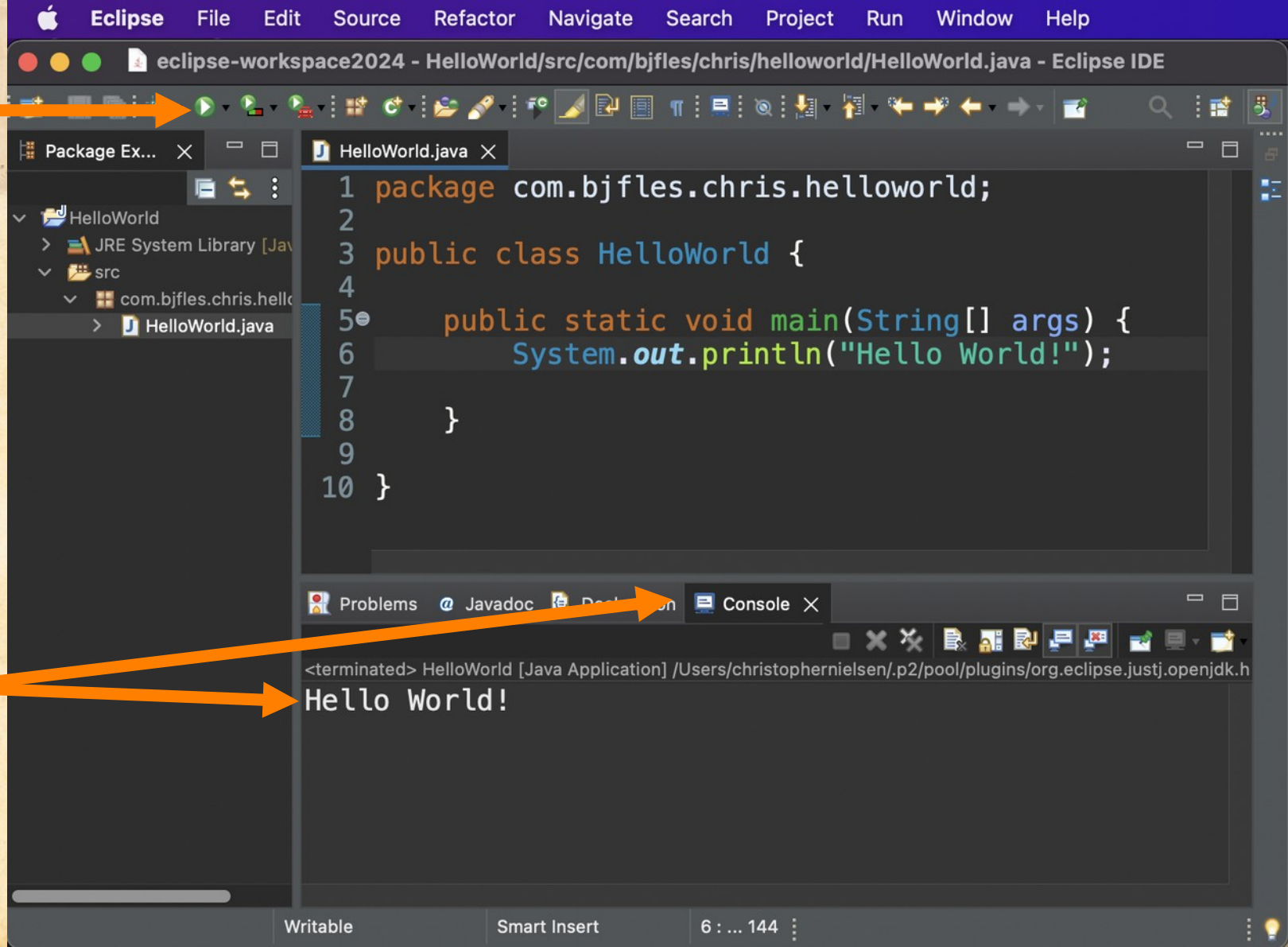


Replace line 6 with your program code.

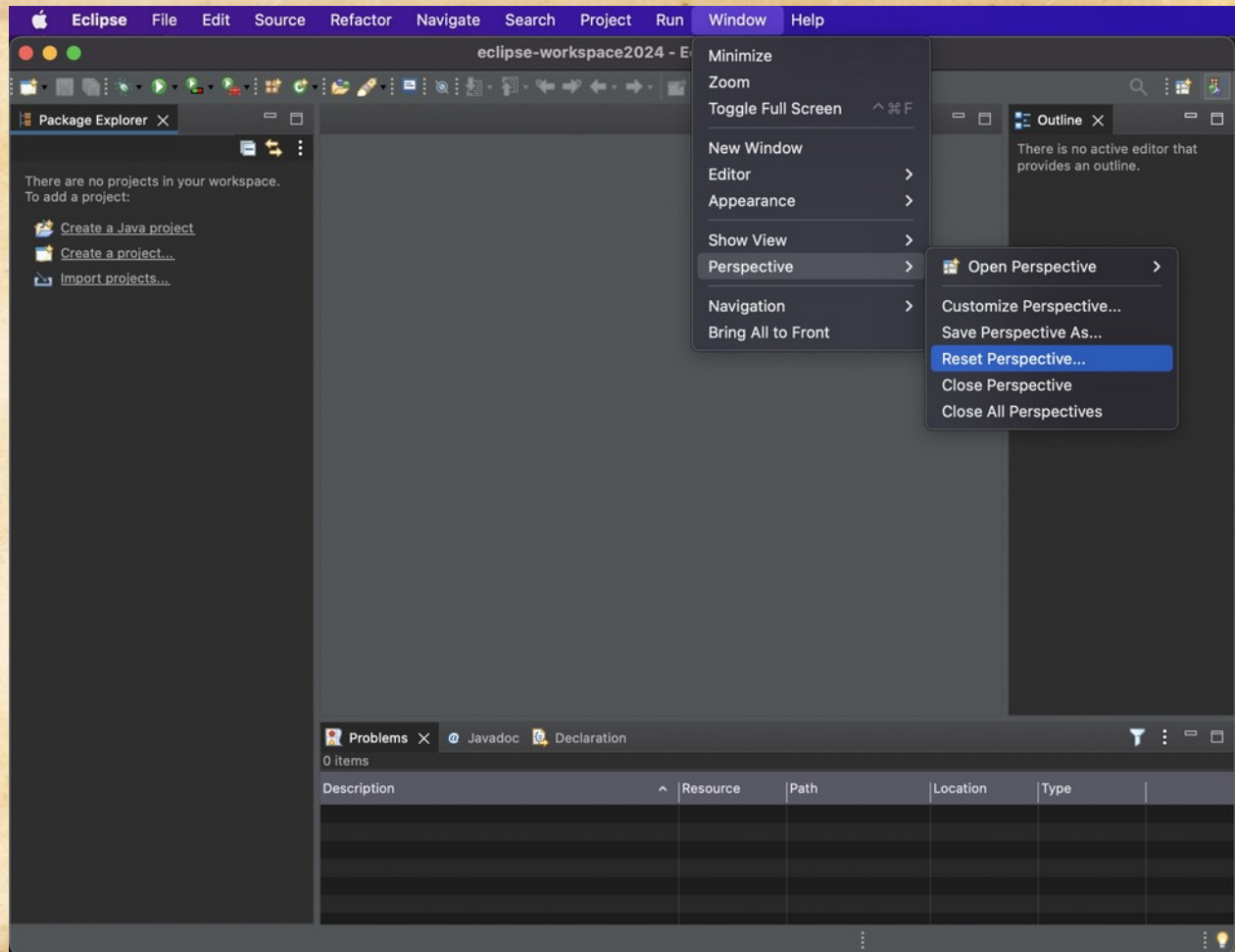


Click here to run.

Check the output in the console.



- ***Hint***: If you ever lose your "**Package Explorer**" or other tabs, "**Reset Perspective…**" might restore the interface to the default settings...



Lecture Contents

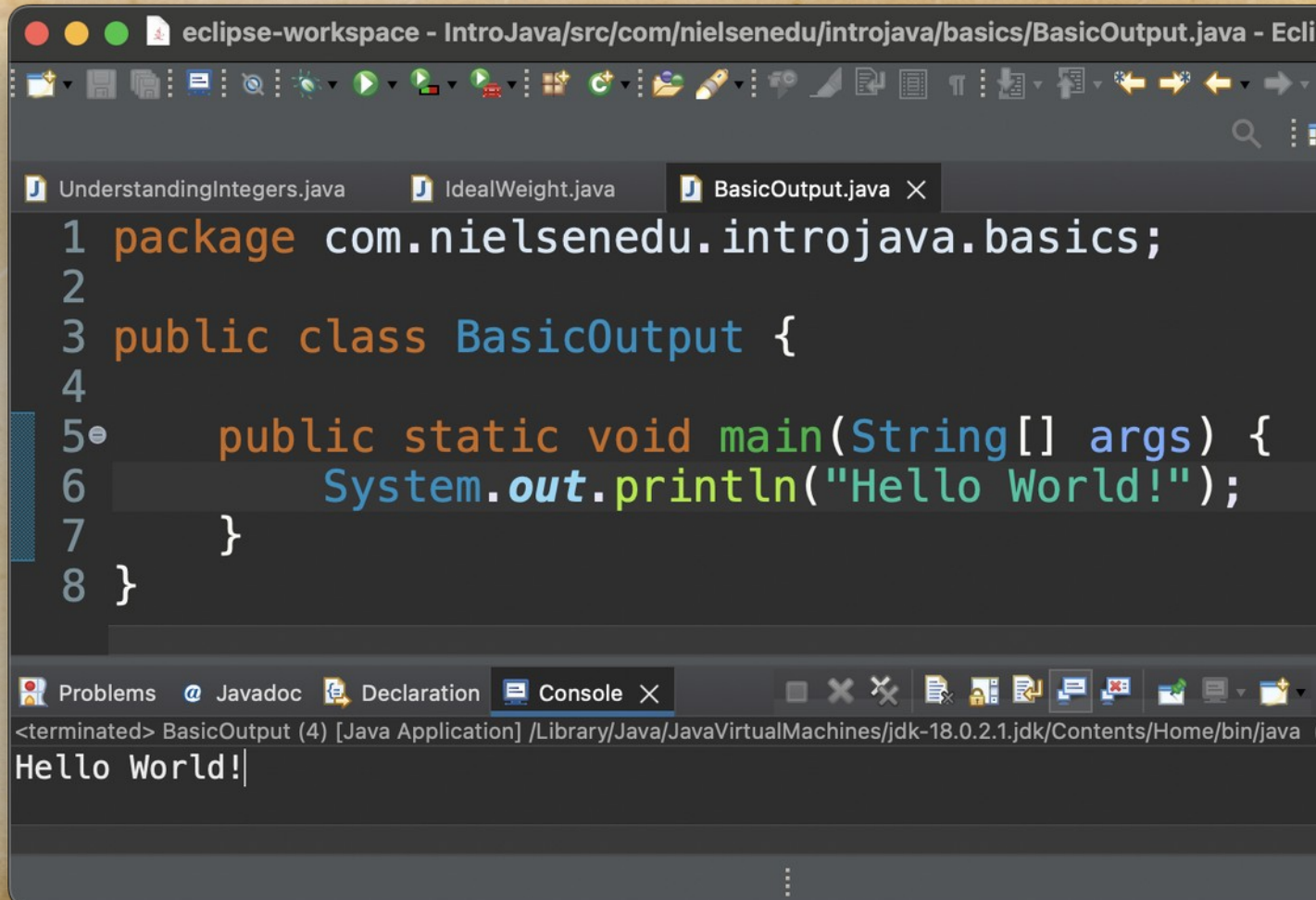
- Review of how we got to "`Hello World!`"
- **Basic Output**
  - `System.out.println` and `System.out.print`
  - *String literals*
  - *Concatenation*
  - Escape characters
- Exercises
  - Predict the output
  - Produce the output

Basic Output

- There are two methods that produce text output to the console.
  - `System.out.println()`
  - `System.out.print()`

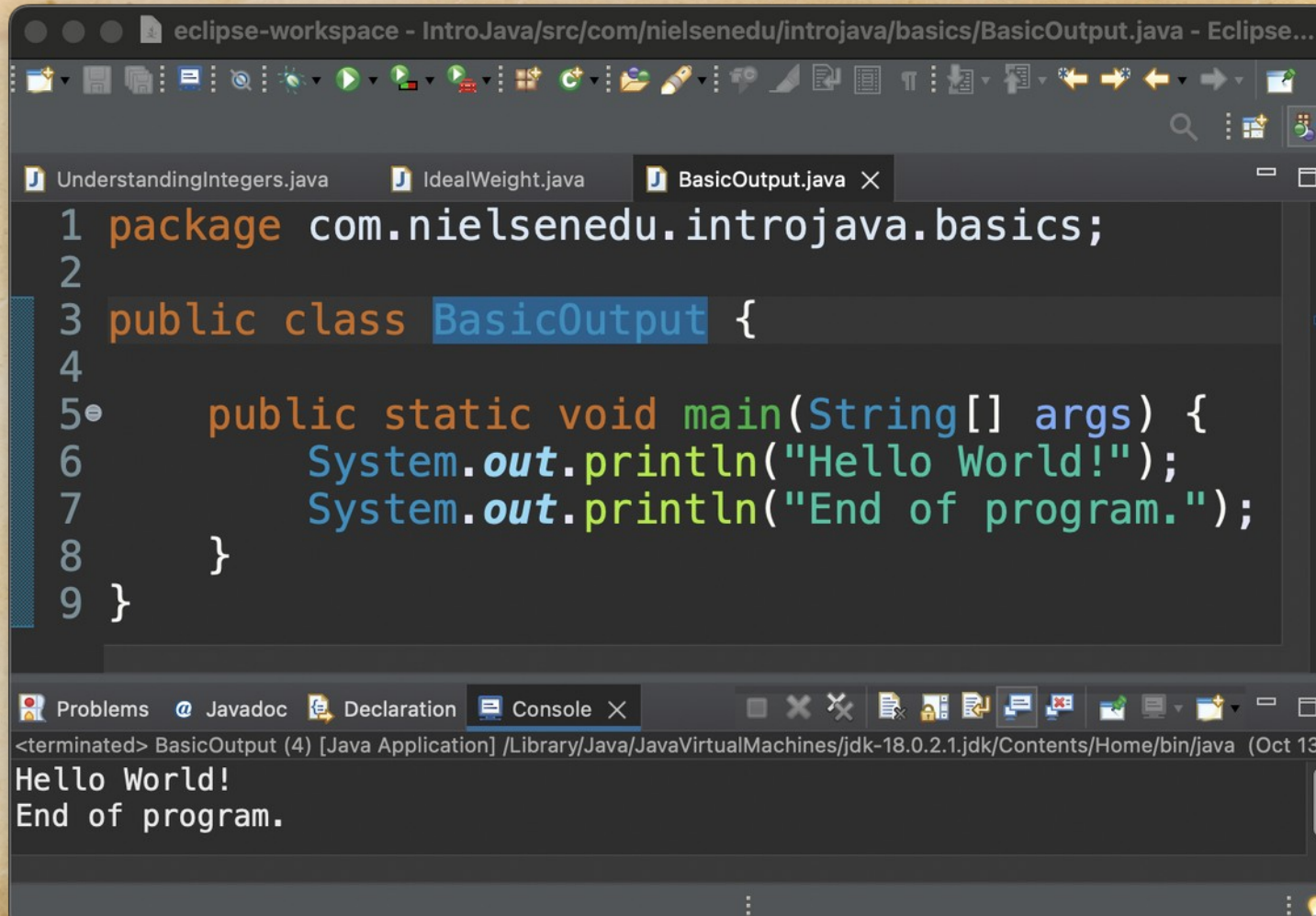
`System.out.println()`

- Prints to the console exactly what is within quotation marks and within the parentheses
  - We saw this in our `HelloWorld` program.



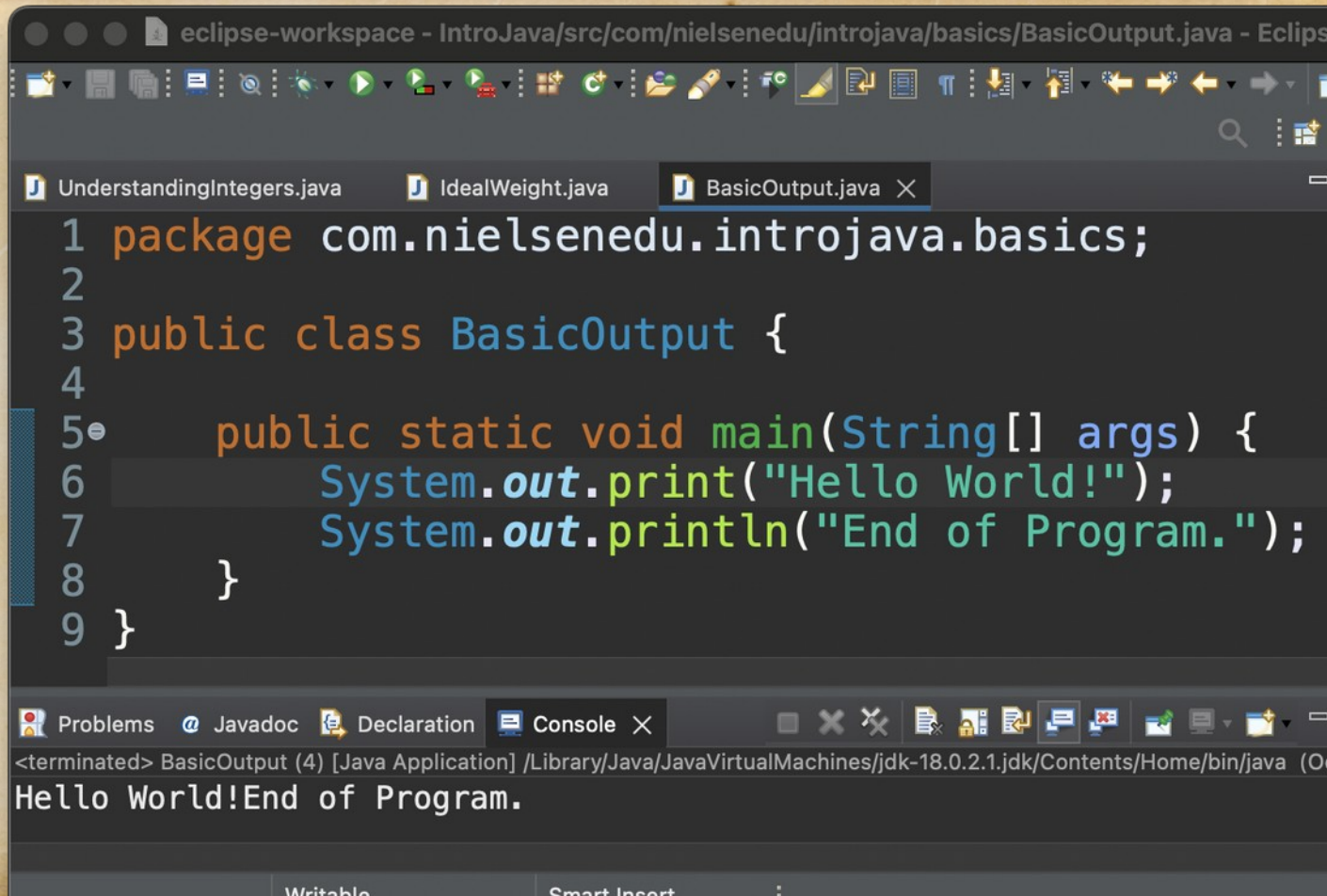
`System.out.println()`

- `println` is short for “print line”.
- *After* a `println` method, the *next* output will start on a new line.



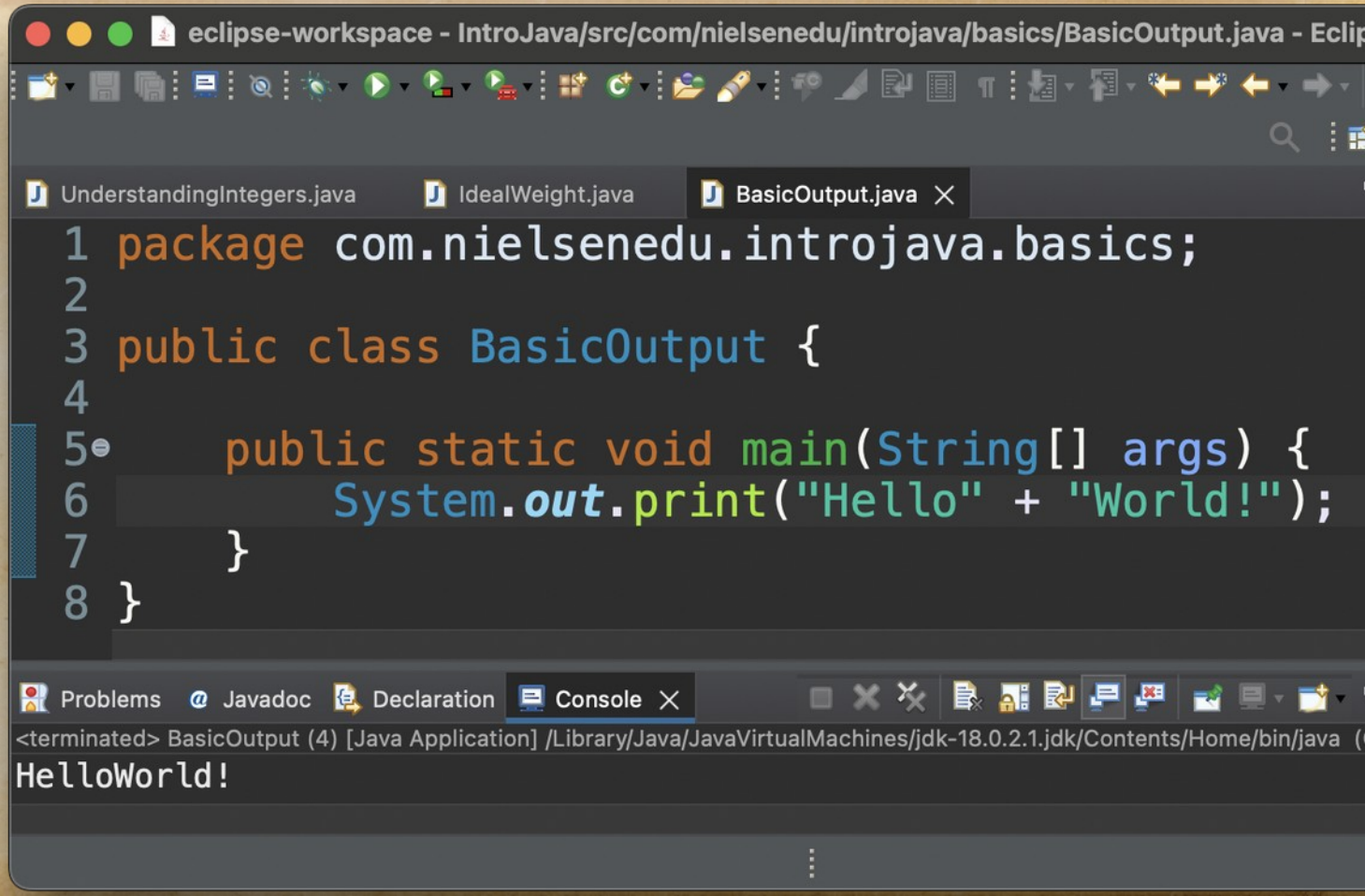
`System.out.print()`

- To continue printing on the same line, use `print` rather than `println`.



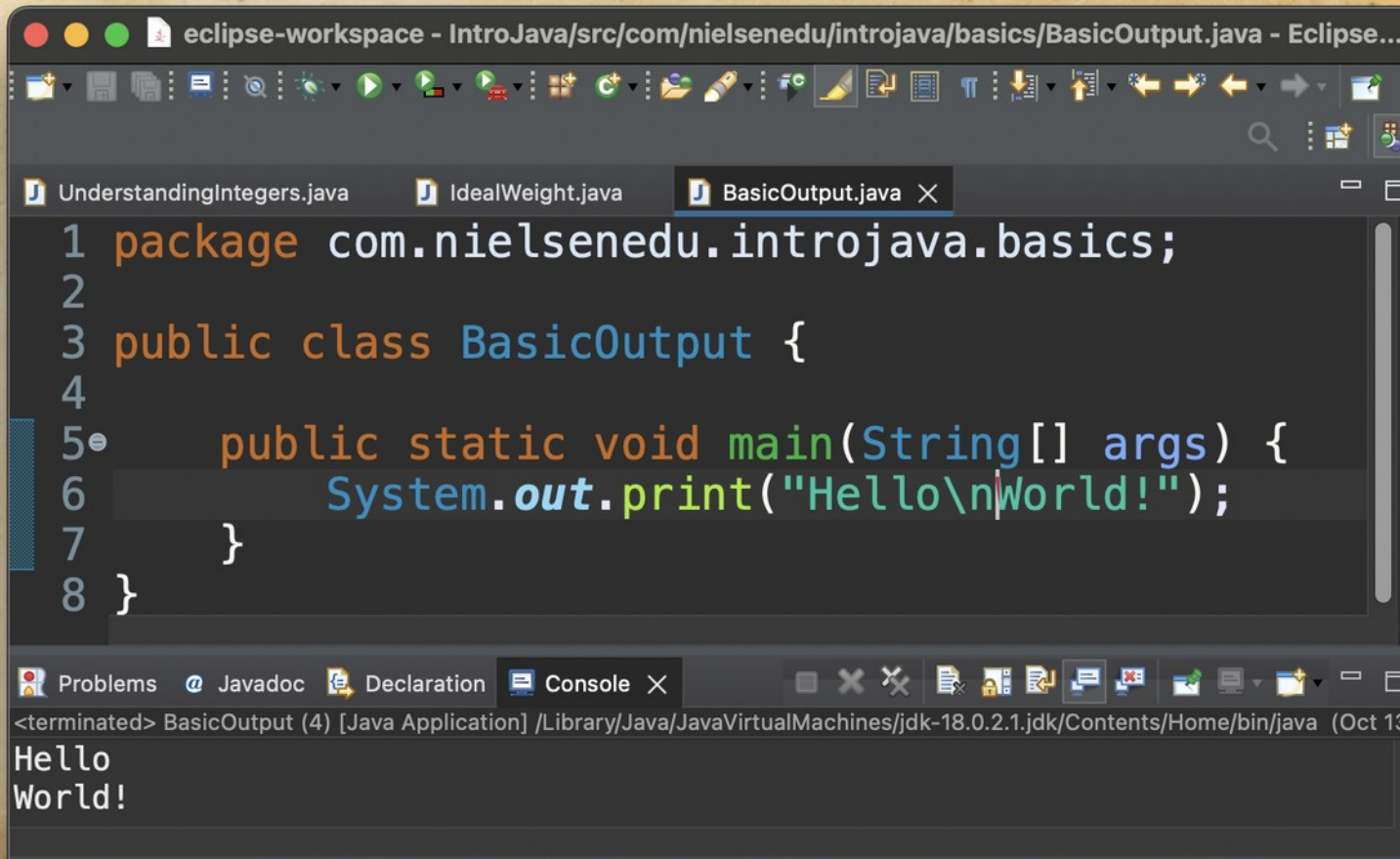
String Literals and *Concatenation*

- Text within quotation marks is called a ***string literal***.
- A plus sign (+) will ***concatenate*** (put together) two *strings*.
- No space character is added between the *strings*.



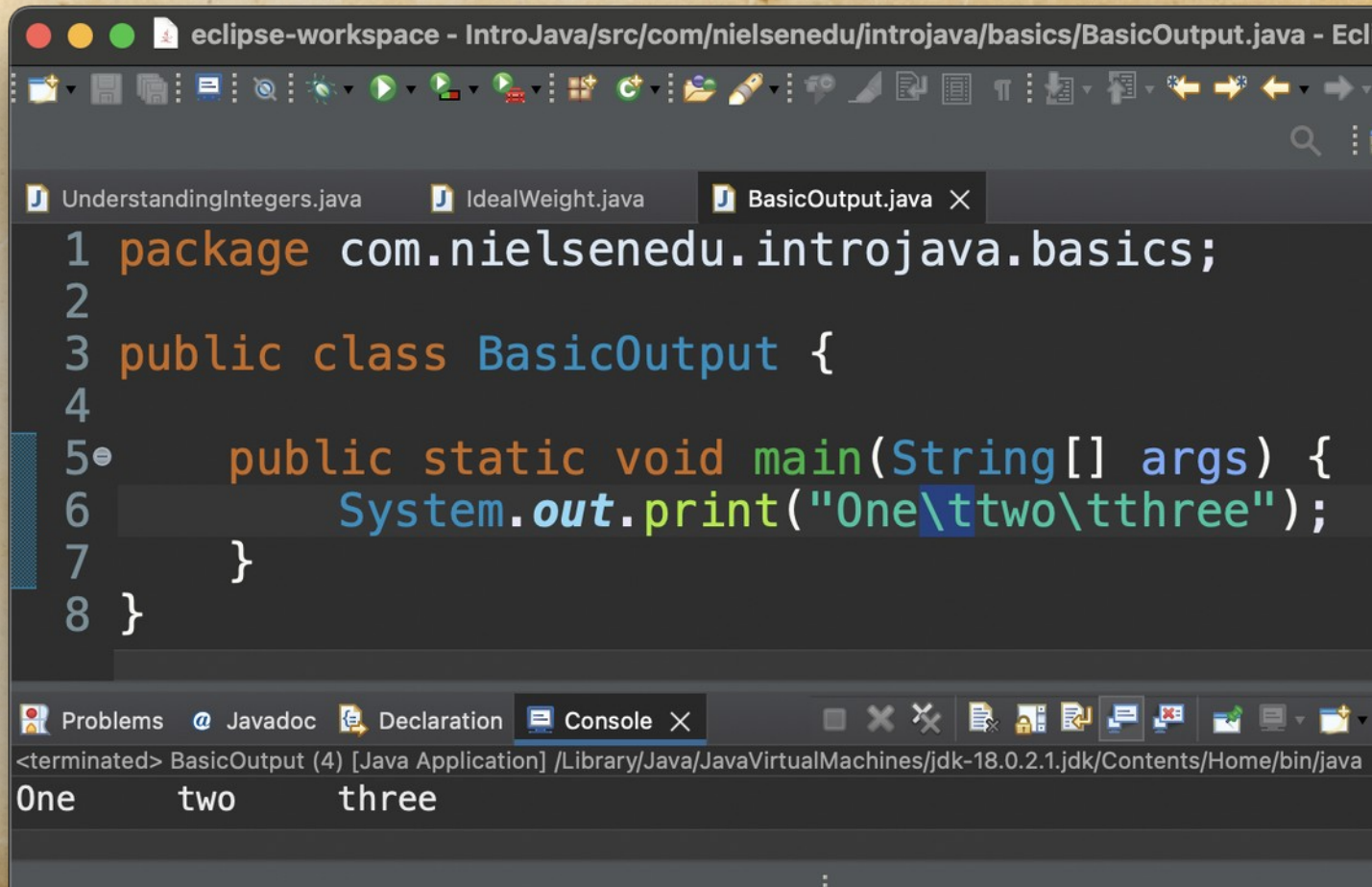
Escape Characters

- "\n"
  - adds a “new line” character
  - The subsequent text continues on a new line.



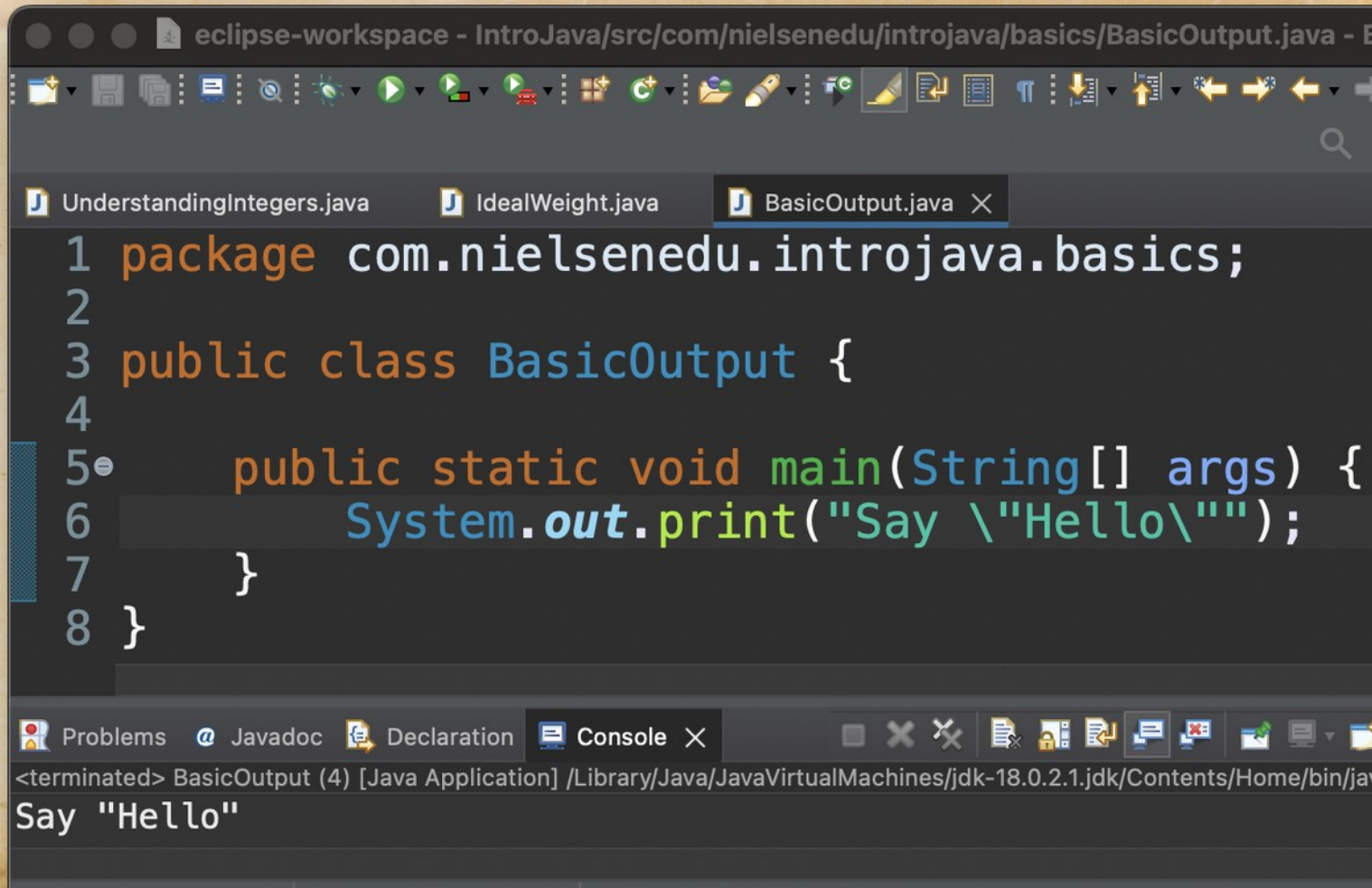
Escape Characters

- "\t"
  - adds a “tab” character
  - The subsequent text continues at the next tab stop.



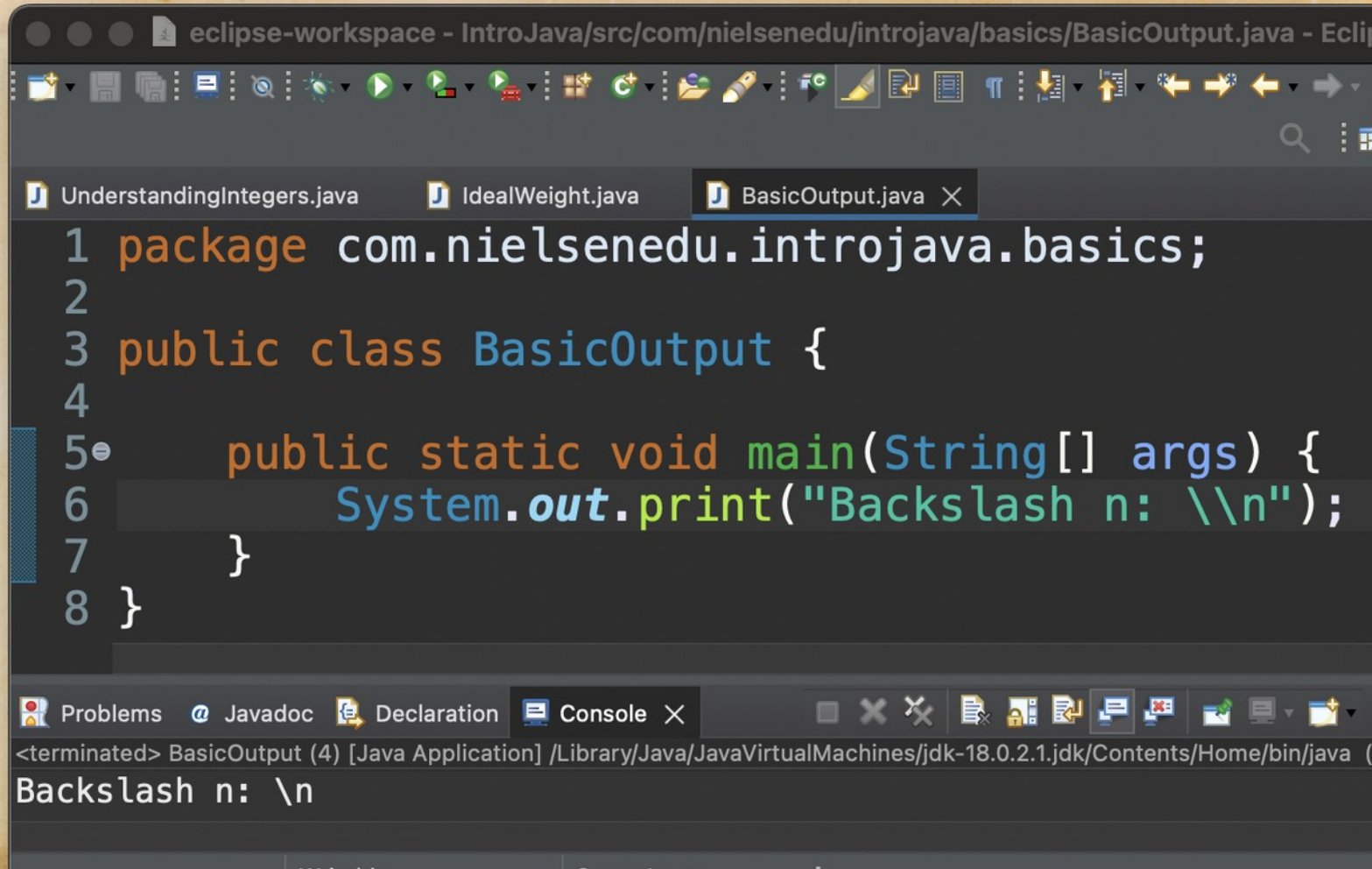
Escape Characters

- "\""
  - adds a double quotation mark character (")



Escape Characters

- "\\"
  - adds a single backslash character (\)



Escape Characters (Summarized)

Code	Output
\n	newline
\"	double quote
\'	single quote
\\	backslash
\t	tab
\b	backspace
\r	carriage return
\f	form feed

```
System.out.println("Hello\nThere.");
System.out.println("Say \"Hello\"");
System.out.println("One backslash \\");
```

```
Hello
There
Say "Hello"
One backslash \
```

Predict the Output

Code	Output
\n	newline
\"	double quote
\'	single quote
\\	backslash
\t	tab
\b	backspace
\r	carriage return
\f	form feed

```
System.out.print("Hello");
System.out.print("World!");
```

```
HelloWorld!
```

Predict the Output

Code	Output
\n	newline
\"	double quote
\'	single quote
\\	backslash
\t	tab
\b	backspace
\r	carriage return
\f	form feed

```
System.out.print("\"\n\"");
```

```
"
"
```

Predict the Output

Code	Output
\n	newline
\"	double quote
\'	single quote
\\	backslash
\t	tab
\b	backspace
\r	carriage return
\f	form feed

```
System.out.print("\"\t\\\n\"\t\\\\");
```

```
"   \
"   \\
```

Predict the Output

Code	Output
\n	newline
\"	double quote
\'	single quote
\\	backslash
\t	tab
\b	backspace
\r	carriage return
\f	form feed

```
System.out.print("'\'");
```

```
  ''
```

Produce the Output

Code	Output
\n	newline
\"	double quote
\'	single quote
\\	backslash
\t	tab
\b	backspace
\r	carriage return
\f	form feed

Produce the following output

- without using any space characters in your string literals.
- you may use multiple calls to the `print` or `println` method.

```
1     a
10    b
100   c
```

Produce the Output

Code	Output
\n	newline
\"	double quote
\'	single quote
\\	backslash
\t	tab
\b	backspace
\r	carriage return
\f	form feed

Produce the following output:

- using a single string literal
- without using any space characters in your string literals.

```
1
        2
                3
```

Produce the Output

Code	Output
\n	newline
\"	double quote
\'	single quote
\\	backslash
\t	tab
\b	backspace
\r	carriage return
\f	form feed

Produce the following output:

```
<h1 id="first">Using "\n"</h1>
```

Java Basics

Basic Text Output