

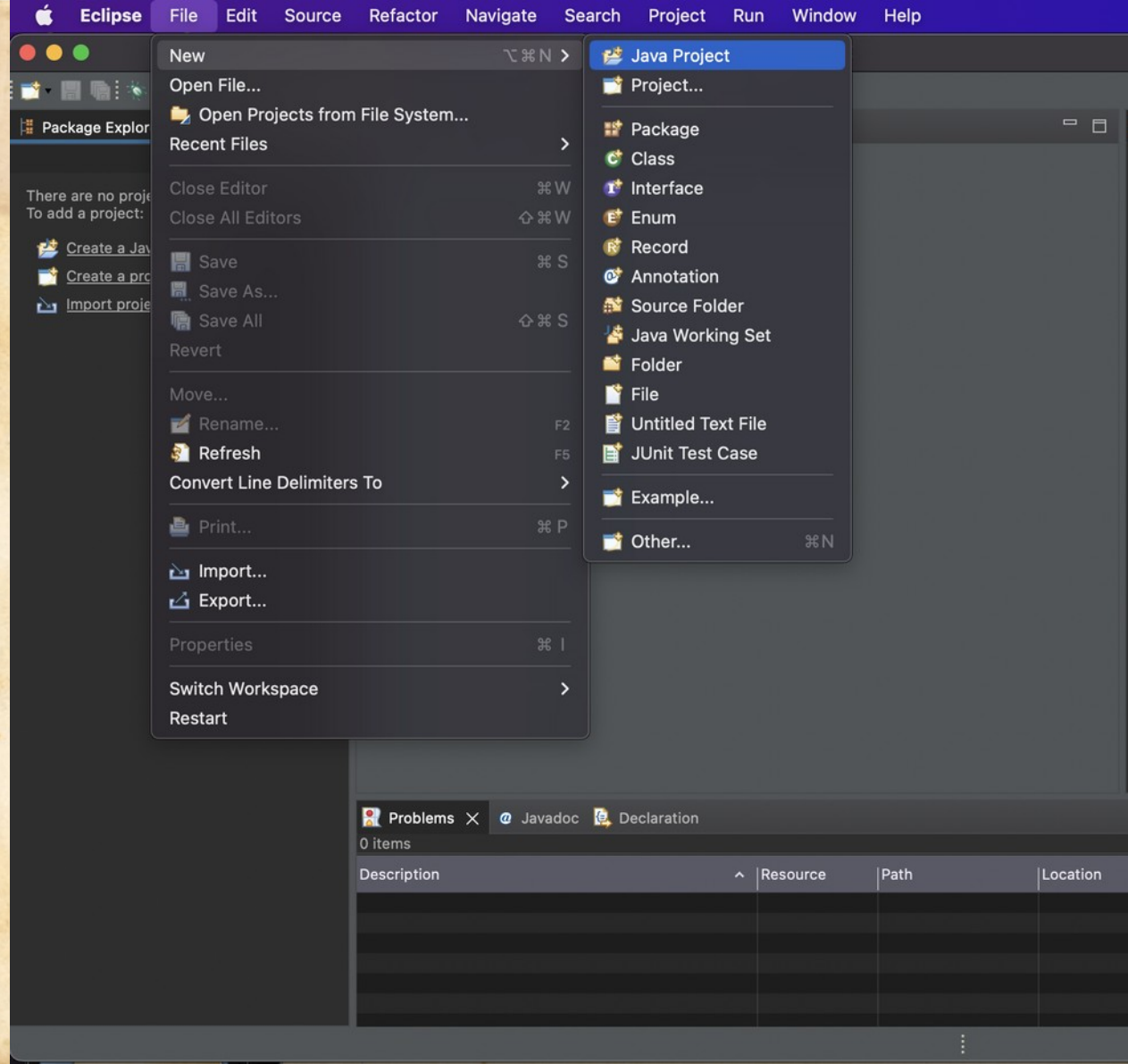
Java Basics

Basic Text Output

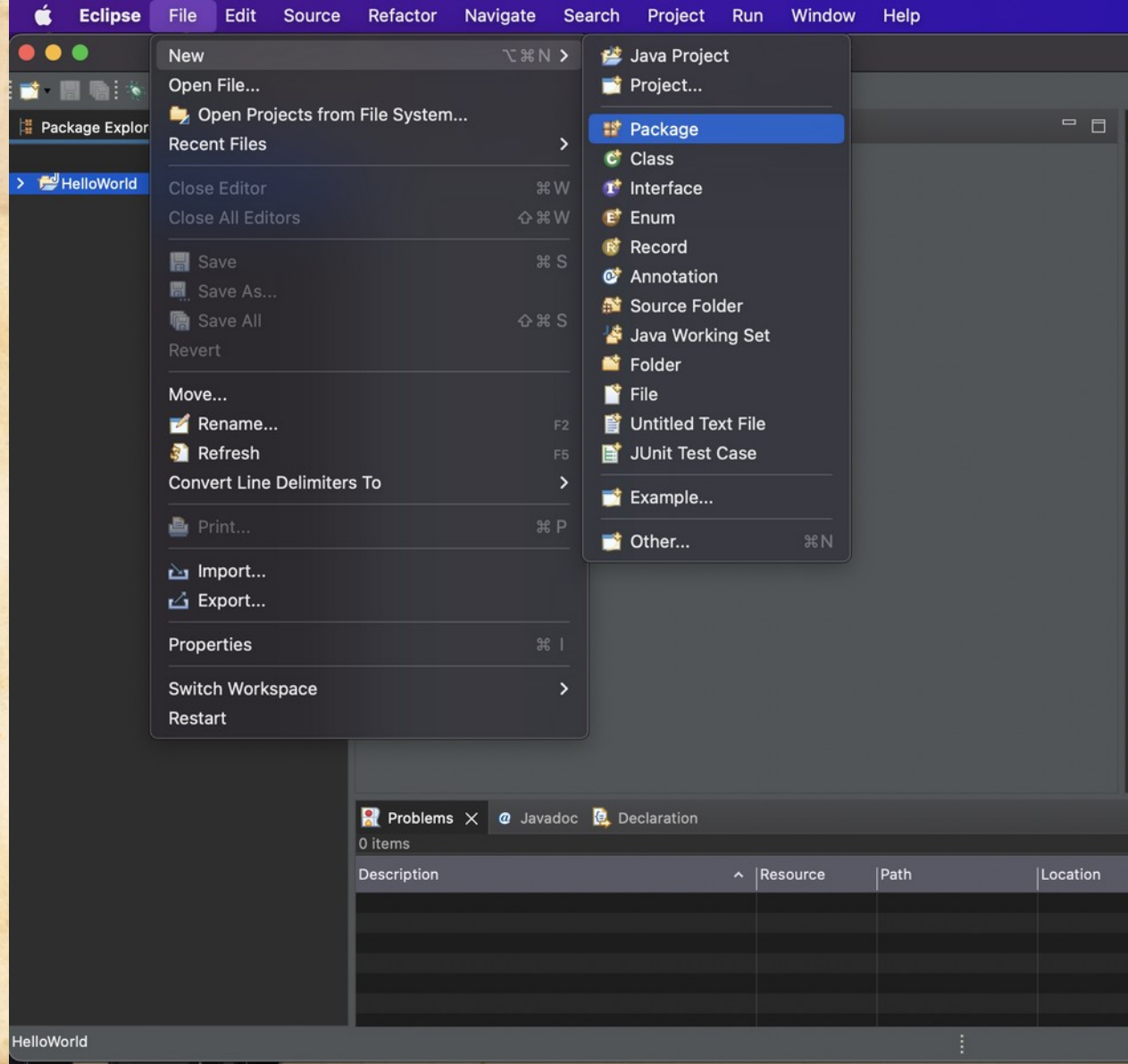
Lecture Contents

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

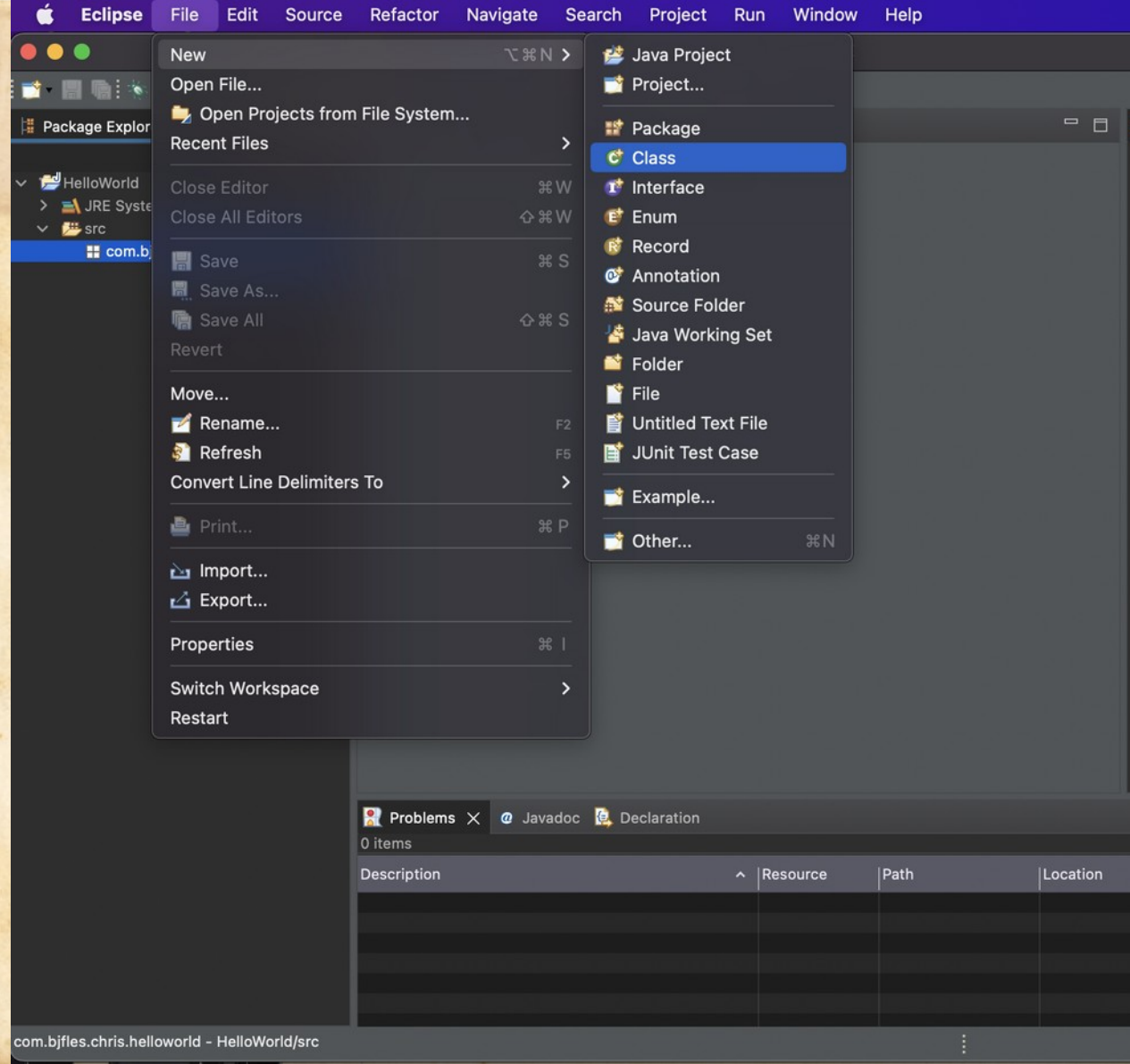
- For this course, it is okay to keep all files in the same Java project.



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

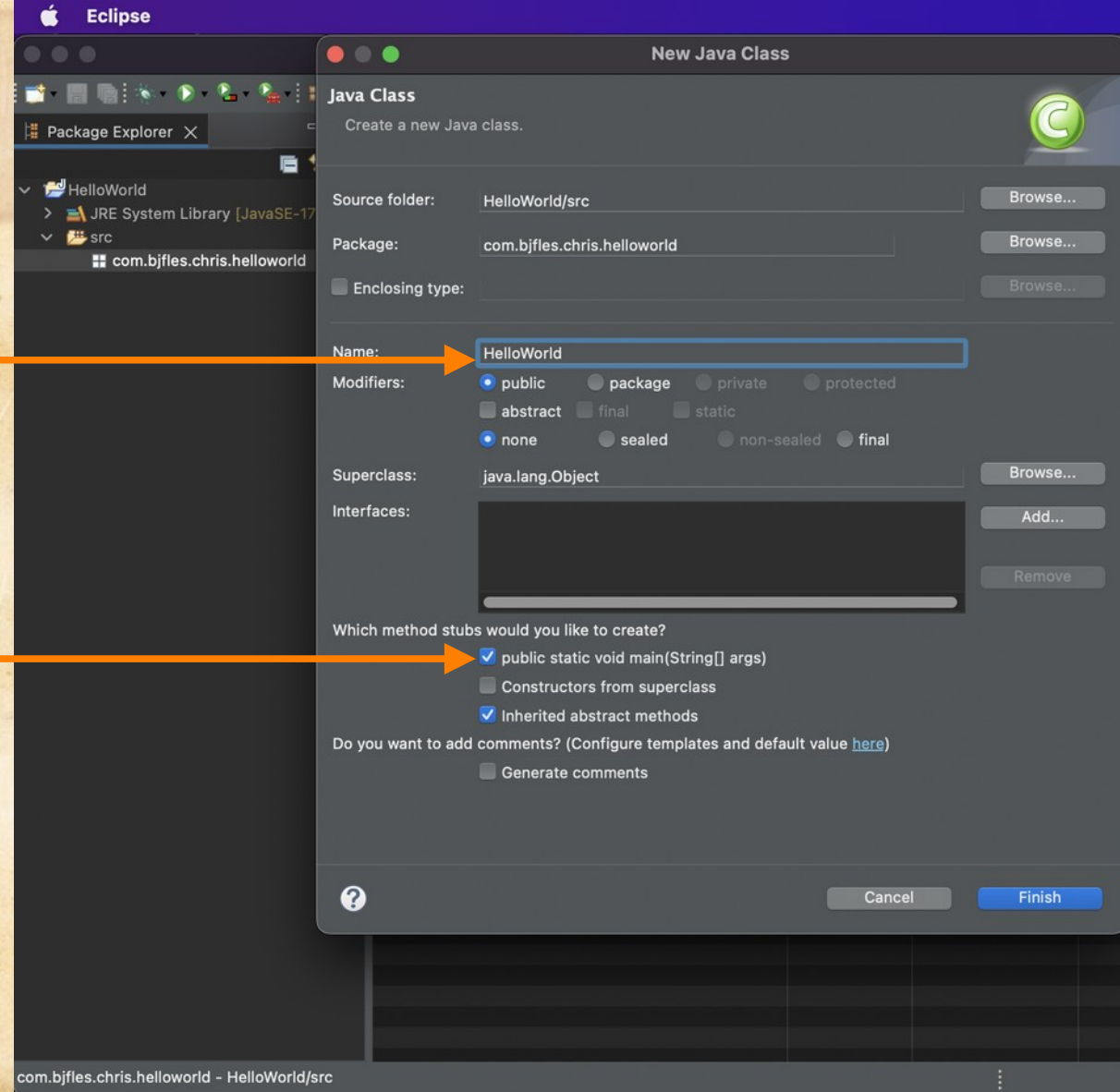


- Create a new class for each new program.
- Class names use CamelCase.
- For today's task, perhaps call the class:
 - BasicOutput

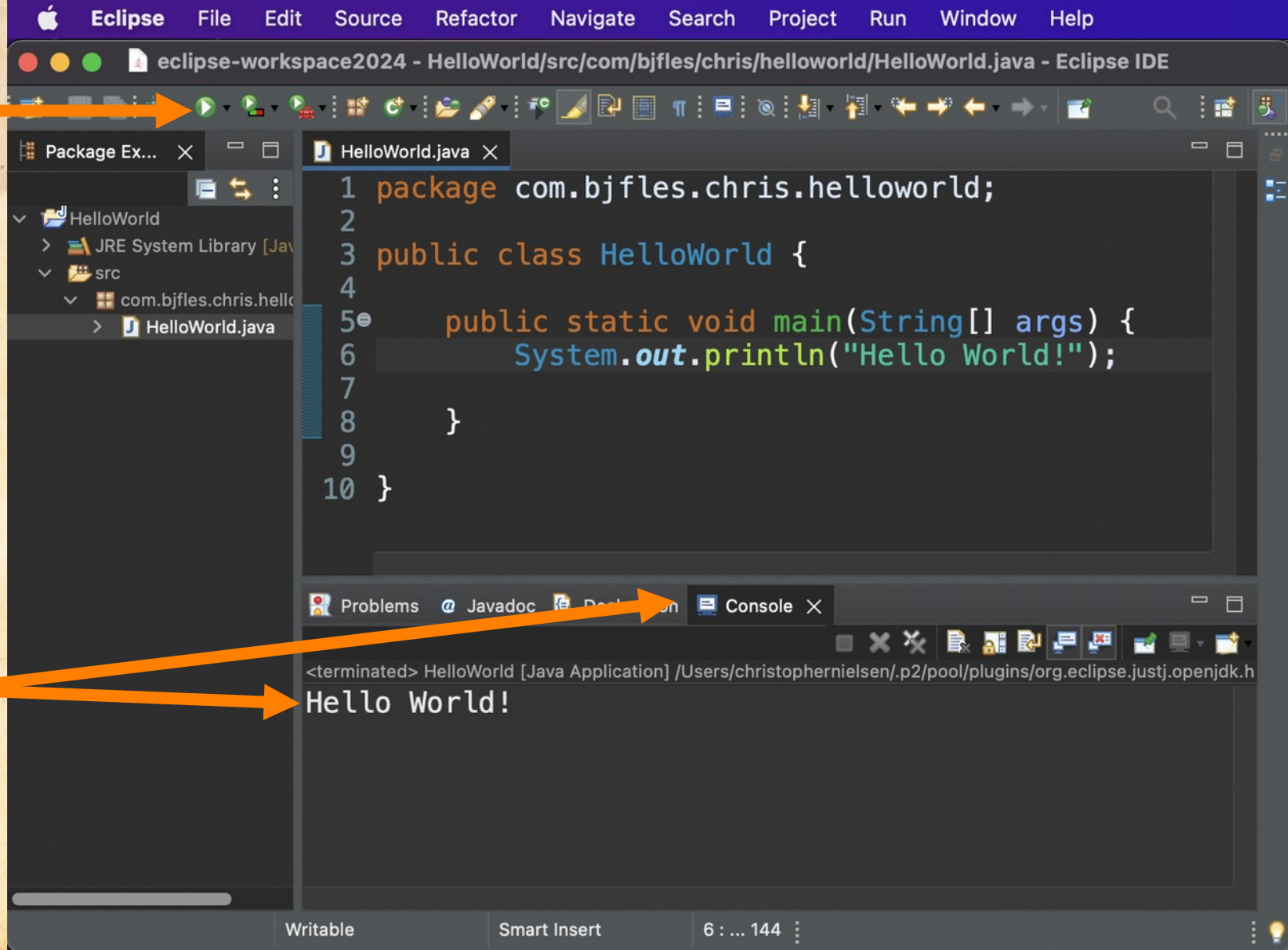


Change this to “BasicOutput”
for today’s assignment.

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

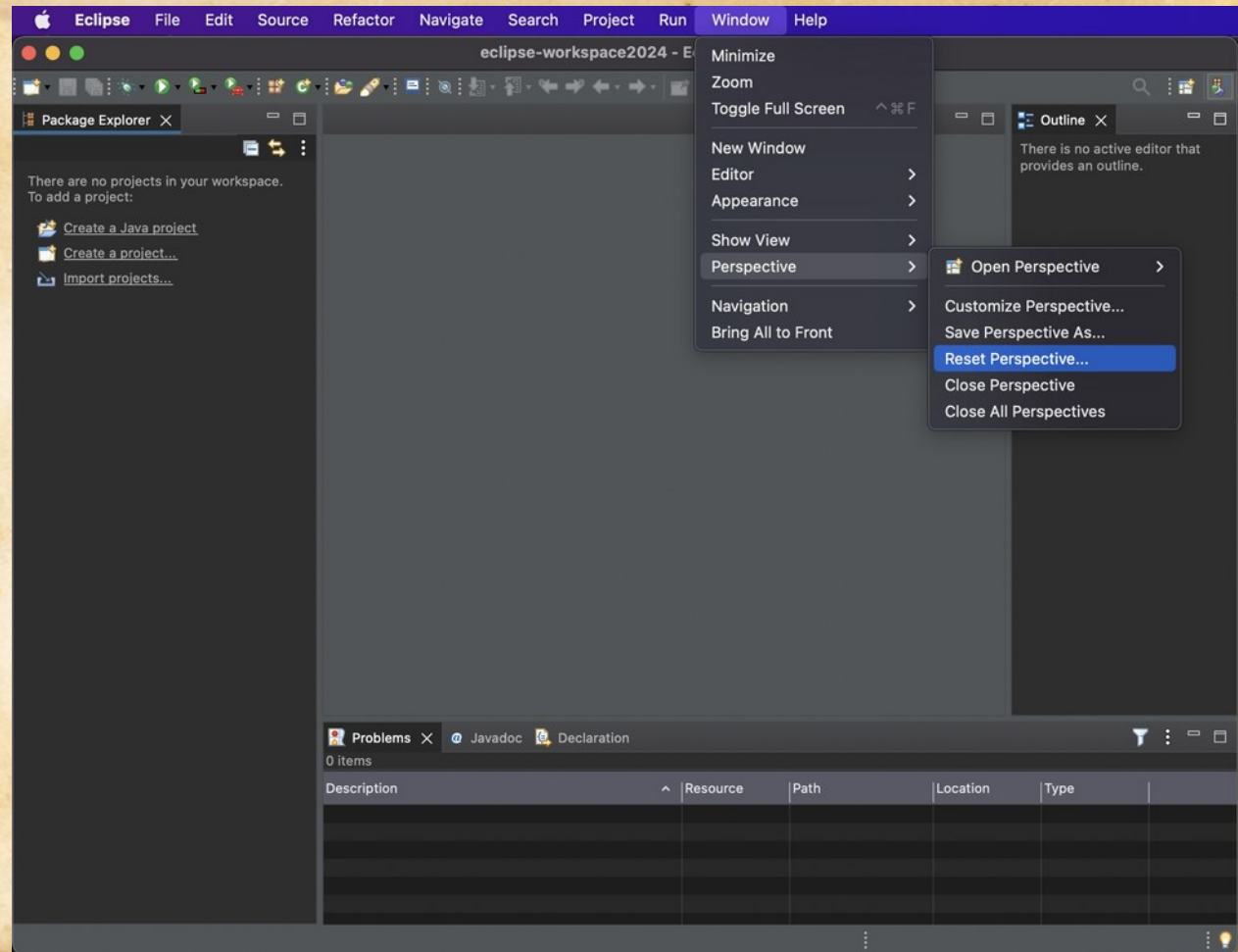


Click here
to run.



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

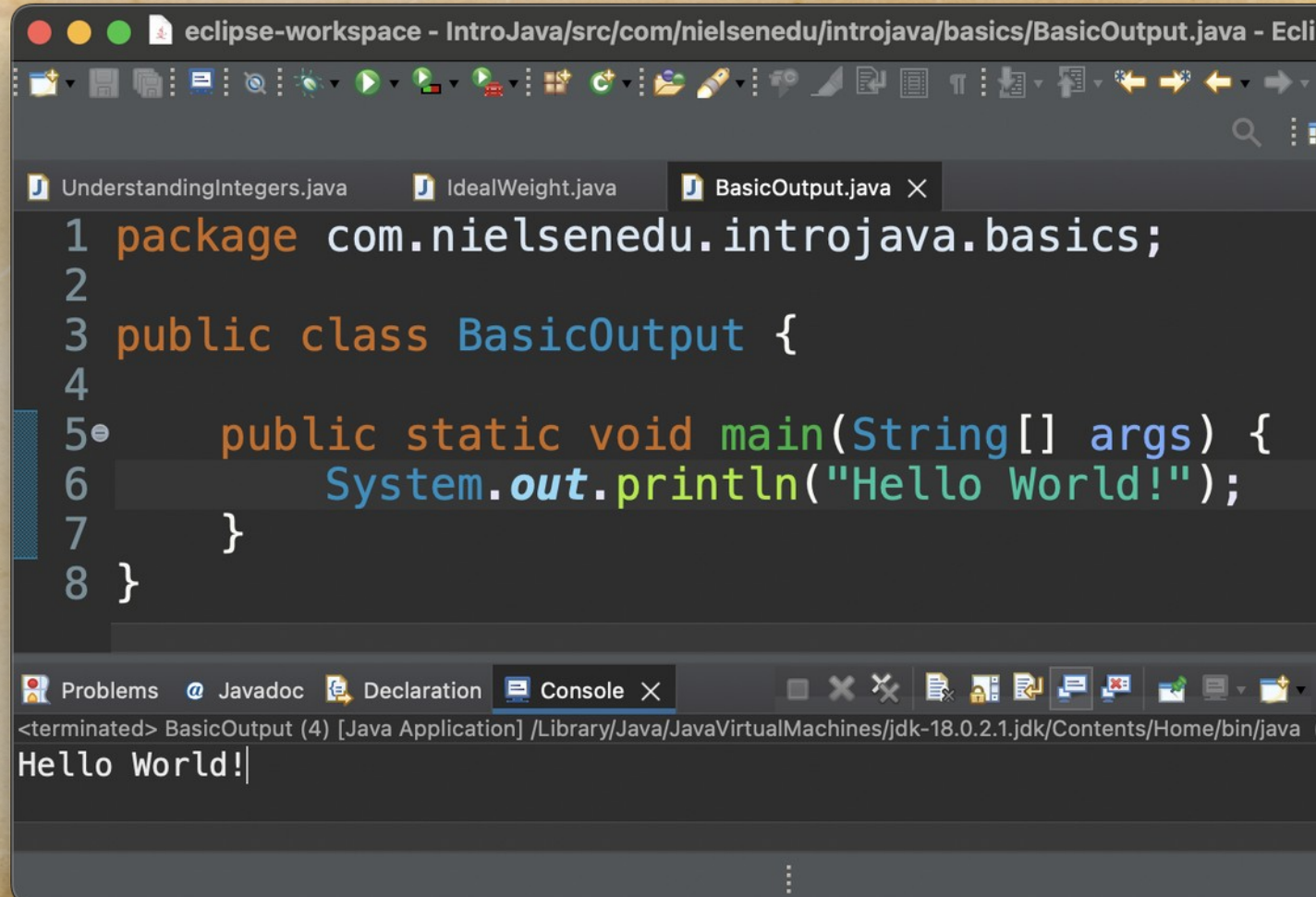
- Review of how we got to “Hello World!”
- **Basic Output**
 - `System.out.println` and `System.out.print`
 - *String literals*
 - *Concatenation*
 - Escape characters
- Exercises
 - Predict the output
 - Produce the output

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.



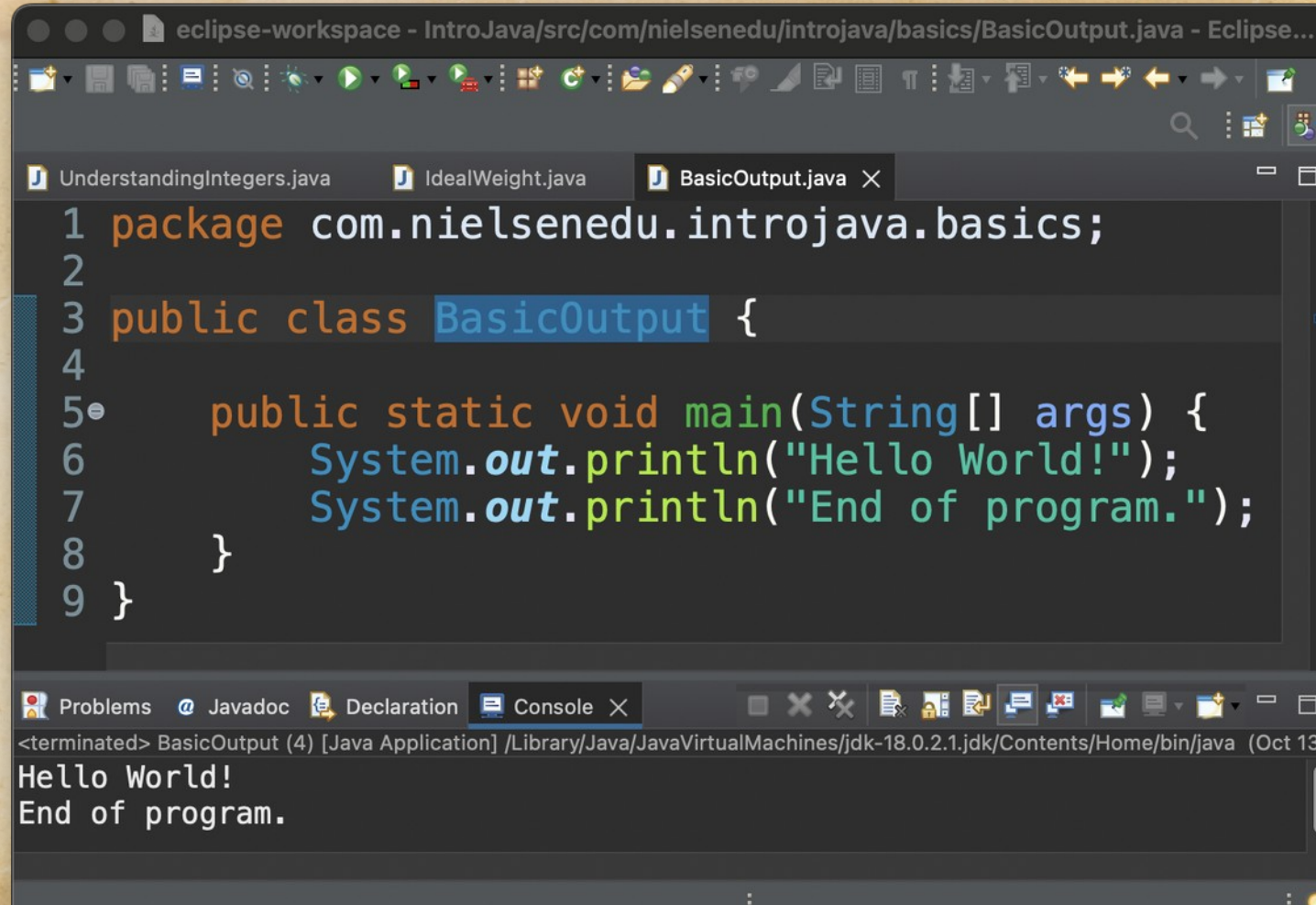
The screenshot shows the Eclipse IDE interface. The top toolbar contains various icons for file operations and development tools. The editor window displays the file 'BasicOutput.java' with the following code:

```
1 package com.nielsenedu.introjava.basics;  
2  
3 public class BasicOutput {  
4  
5     public static void main(String[] args) {  
6         System.out.println("Hello World!");  
7     }  
8 }
```

The bottom of the IDE shows the 'Console' tab, which displays the output of the program: 'Hello World!'.

System.out.println()

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



The screenshot shows the Eclipse IDE interface. The top toolbar contains various icons for file operations and development tools. The editor window displays the file `BasicOutput.java` with the following code:

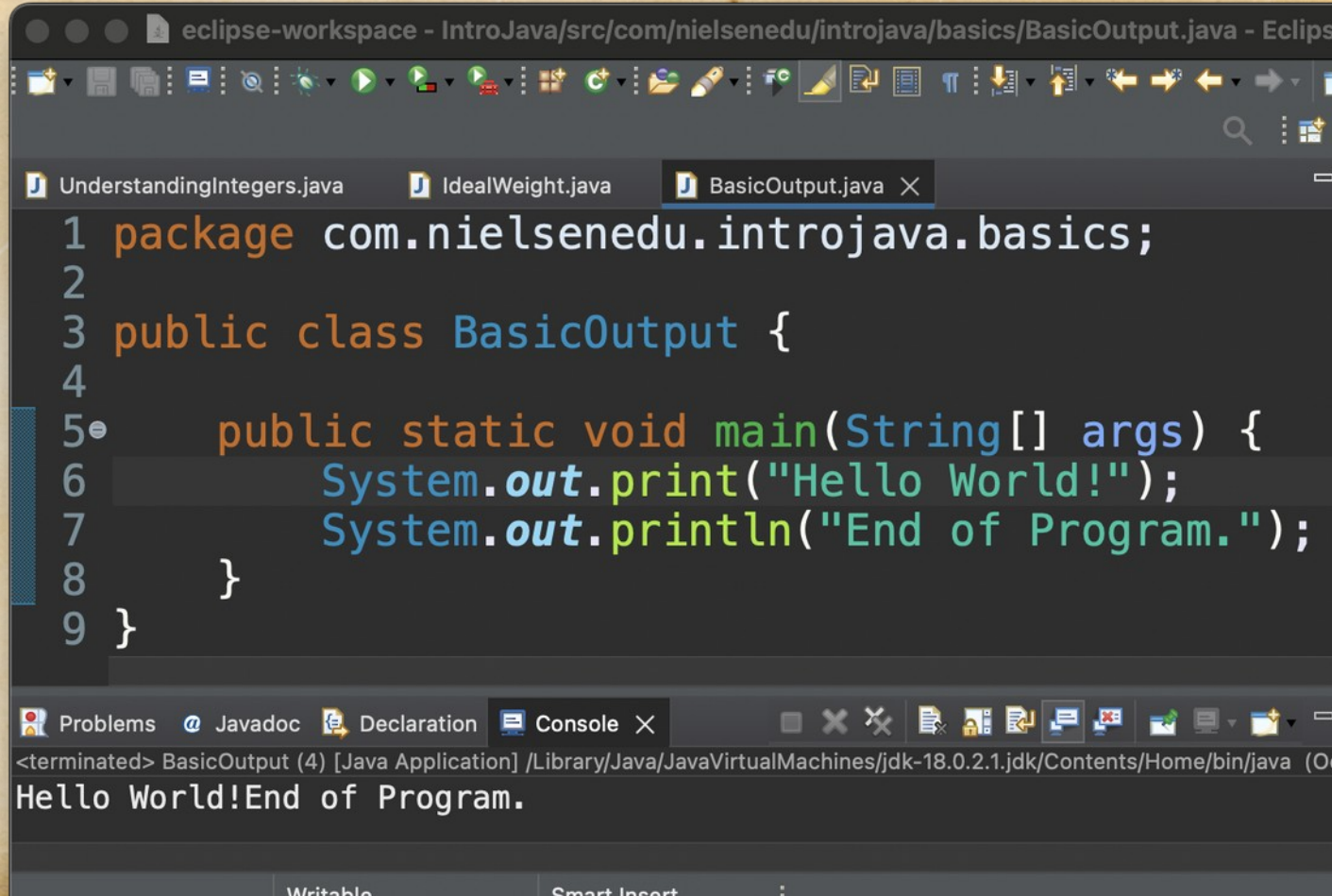
```
1 package com.nielsenedu.introjava.basics;
2
3 public class BasicOutput {
4
5     public static void main(String[] args) {
6         System.out.println("Hello World!");
7         System.out.println("End of program.");
8     }
9 }
```

The bottom of the IDE features a panel with tabs for `Problems`, `Javadoc`, `Declaration`, and `Console`. The `Console` tab is active, showing the output of the program:

```
<terminated> BasicOutput (4) [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java (Oct 13)
Hello World!
End of program.
```

System.out.print()

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



The screenshot shows the Eclipse IDE interface. The title bar indicates the workspace is 'IntroJava/src/com/nielsenedu/introjava/basics/BasicOutput.java - Eclipse'. The editor window displays the following Java code:

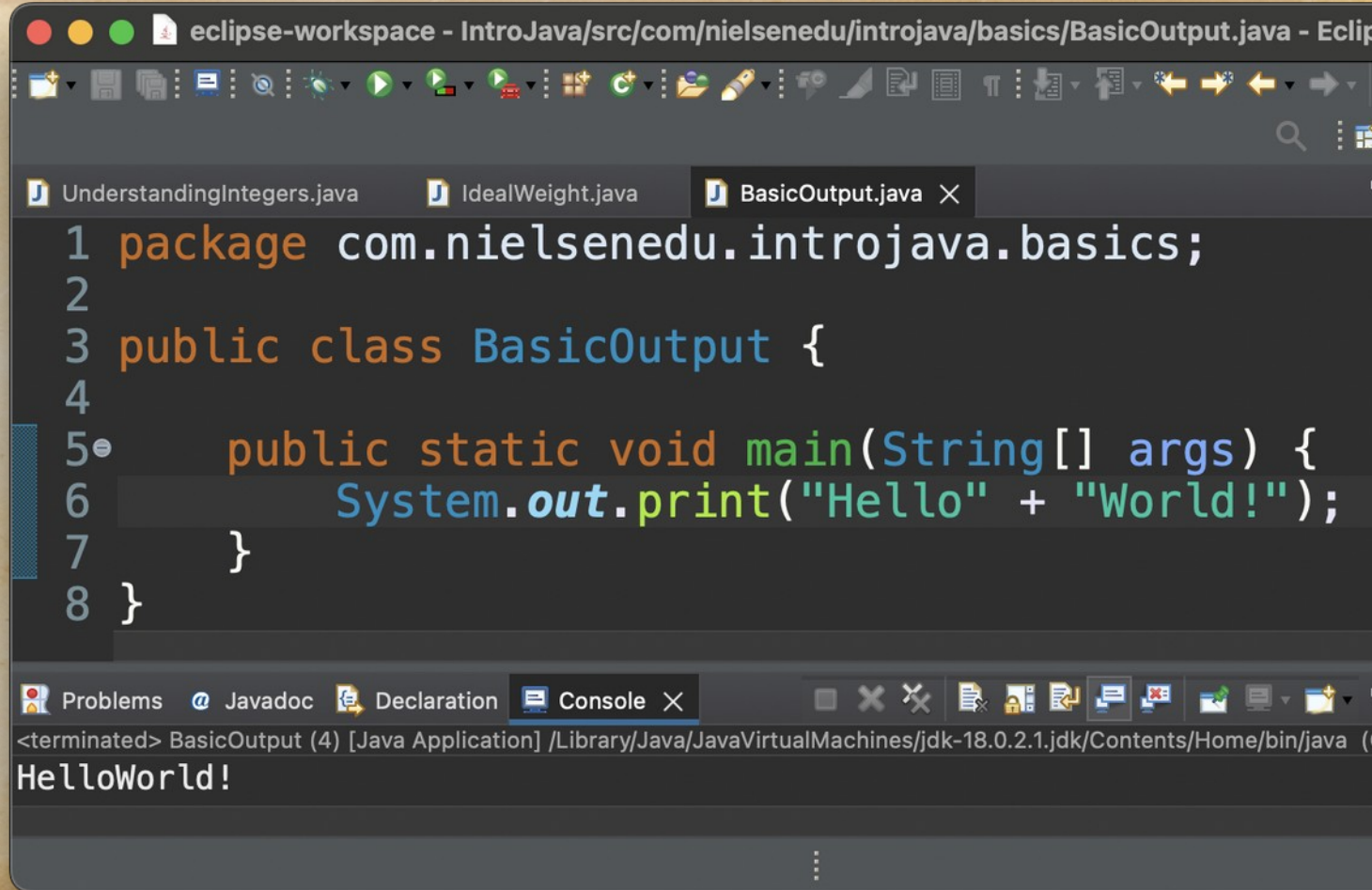
```
1 package com.nielsenedu.introjava.basics;
2
3 public class BasicOutput {
4
5     public static void main(String[] args) {
6         System.out.print("Hello World!");
7         System.out.println("End of Program.");
8     }
9 }
```

The bottom of the IDE shows the 'Console' tab with the output of the program:

```
<terminated> BasicOutput (4) [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java (O
Hello World!End of Program.
```

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*.

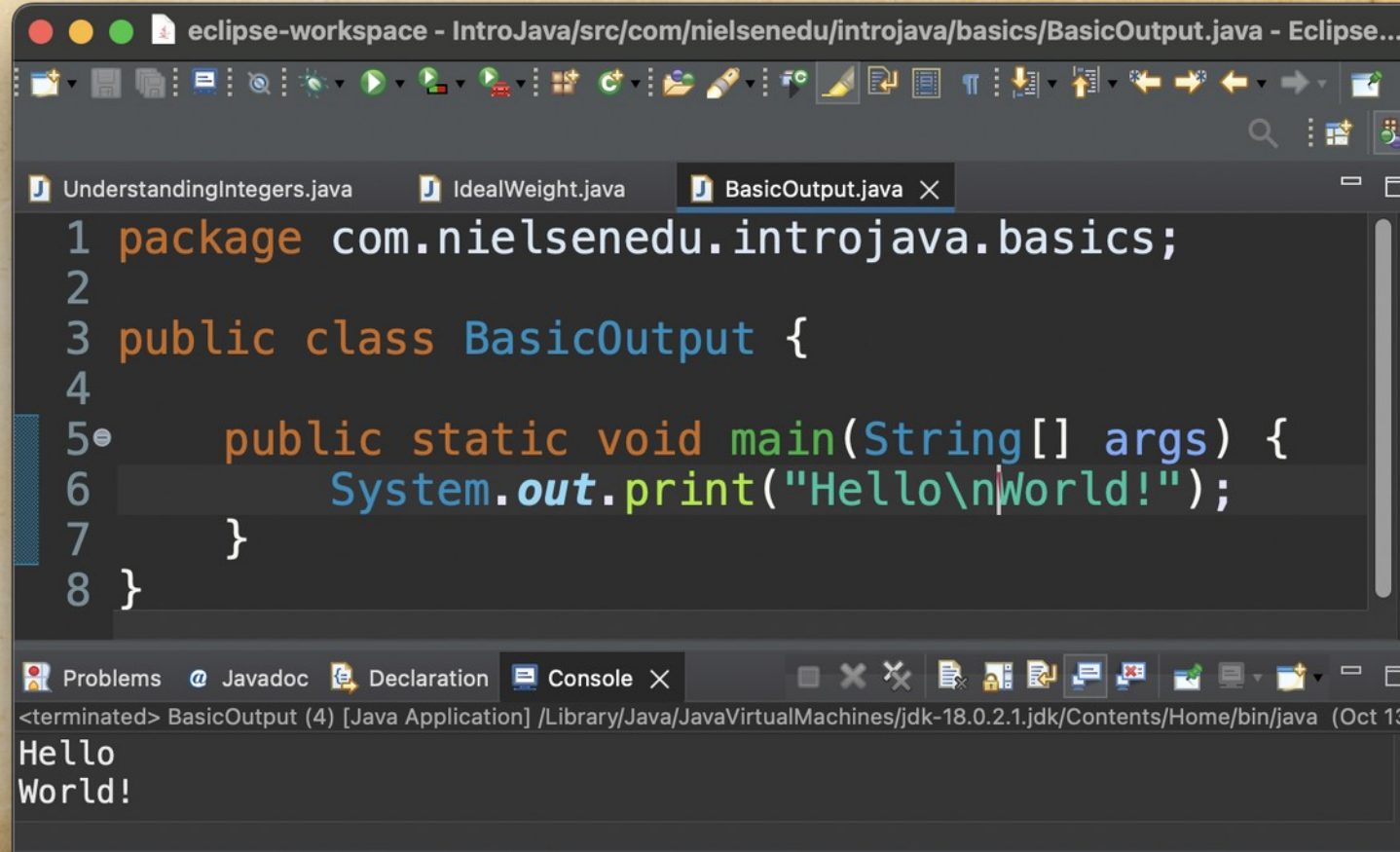
A screenshot of the Eclipse IDE interface. The title bar shows 'eclipse-workspace - IntroJava/src/com/nielsenedu/introjava/basics/BasicOutput.java - Eclipse'. The editor window displays the following Java code:

```
1 package com.nielsenedu.introjava.basics;
2
3 public class BasicOutput {
4
5     public static void main(String[] args) {
6         System.out.print("Hello" + "World!");
7     }
8 }
```

The code is color-coded: keywords like 'package', 'public', 'class', 'void', 'main', and 'String' are in orange or blue, while literals and identifiers are in green. The 'Console' tab at the bottom shows the output: '<terminated> BasicOutput (4) [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java (HelloWorld!'. A red bookmark is visible in the top right corner of the slide.

Escape Characters

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



The screenshot shows the Eclipse IDE with the file `BasicOutput.java` open. The code is as follows:

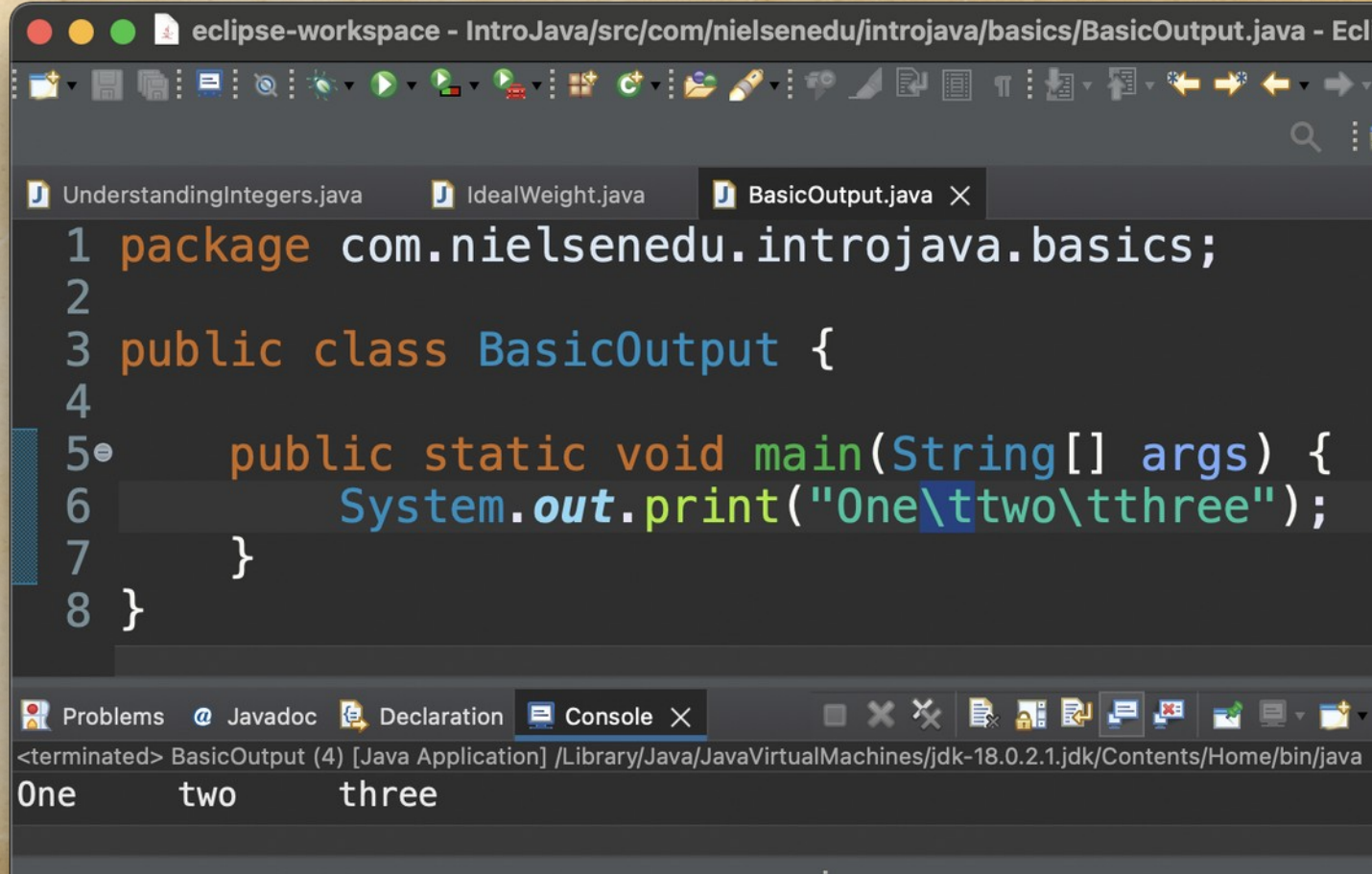
```
1 package com.nielsenedu.introjava.basics;
2
3 public class BasicOutput {
4
5     public static void main(String[] args) {
6         System.out.print("Hello\nWorld!");
7     }
8 }
```

The IDE's console at the bottom shows the output of the program:

```
<terminated> BasicOutput (4) [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java (Oct 13)
Hello
World!
```

Escape Characters

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



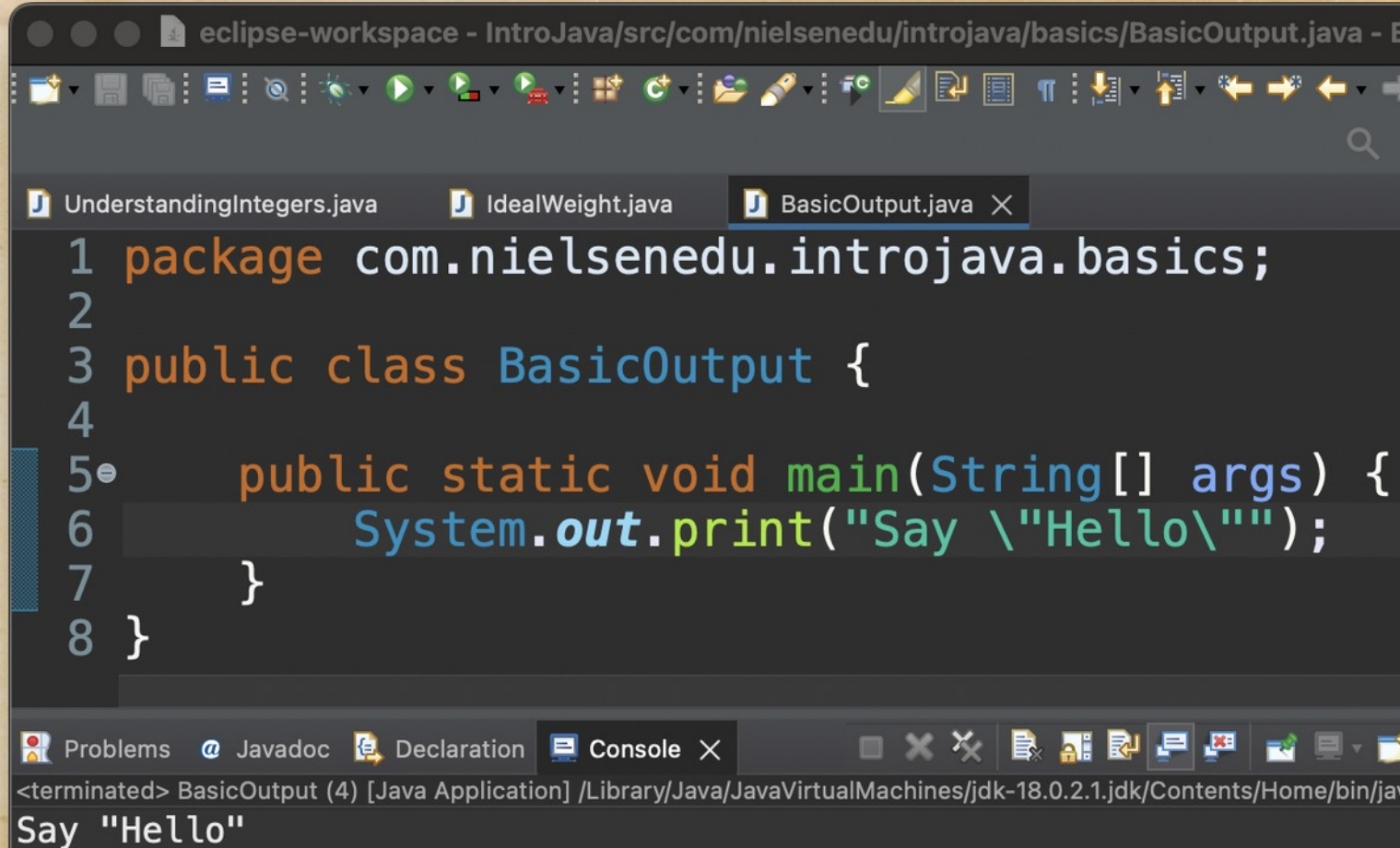
The screenshot shows the Eclipse IDE with a workspace named 'IntroJava'. The project 'com.nielsenedu.introjava' contains a package 'basics' with a class 'BasicOutput.java'. The code in 'BasicOutput.java' is as follows:

```
1 package com.nielsenedu.introjava.basics;
2
3 public class BasicOutput {
4
5     public static void main(String[] args) {
6         System.out.print("One\ttwo\tthree");
7     }
8 }
```

The IDE's console window at the bottom shows the output of the program: 'One two three', where the text is aligned to the next tab stop after 'One' and 'two'.

Escape Characters

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



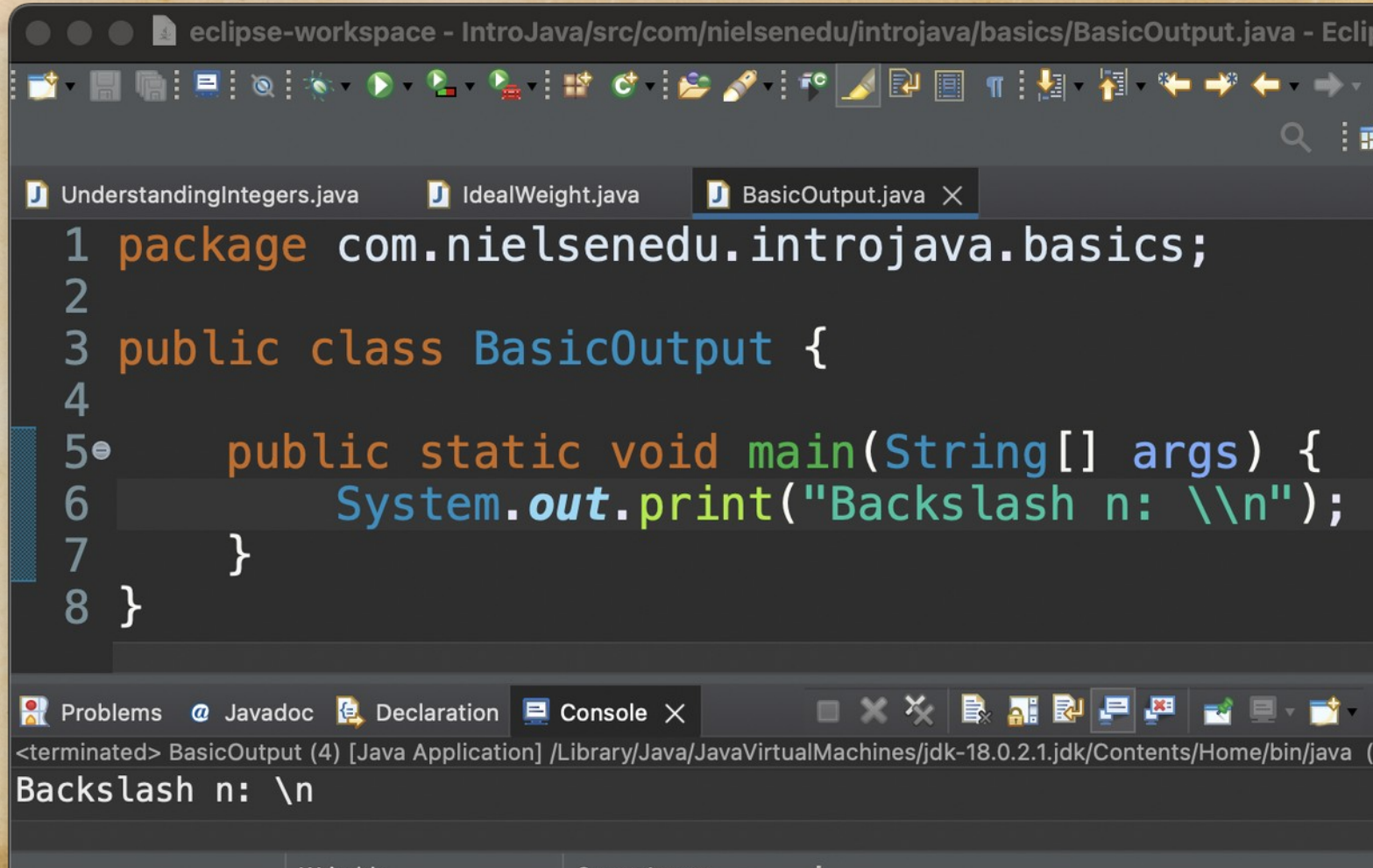
The screenshot shows the Eclipse IDE interface. The title bar indicates the workspace is 'IntroJava/src/com/nielsenedu/introjava/basics/BasicOutput.java'. The editor displays the following Java code:

```
1 package com.nielsenedu.introjava.basics;  
2  
3 public class BasicOutput {  
4  
5     public static void main(String[] args) {  
6         System.out.print("Say \"Hello\"");  
7     }  
8 }
```

The bottom of the IDE shows the 'Console' view with the output: `<terminated> BasicOutput (4) [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java Say "Hello"`. A red bookmark is visible in the top right corner of the slide.

Escape Characters

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



The screenshot shows the Eclipse IDE interface. The title bar indicates the workspace is 'eclipse-workspace' and the current file is 'IntroJava/src/com/nielsenedu/introjava/basics/BasicOutput.java'. The editor displays the following Java code:

```
1 package com.nielsenedu.introjava.basics;
2
3 public class BasicOutput {
4
5     public static void main(String[] args) {
6         System.out.print("Backslash n: \\n");
7     }
8 }
```

The bottom of the IDE shows the 'Console' tab with the output: `<terminated> BasicOutput (4) [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java (Backslash n: \n`

Escape Characters (Summarized)



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

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

HelloWorld!

Predict the Output

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

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

```
"  
"
```

Predict the Output

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

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

```
"  \\  
"  \\  
"
```

Predict the Output

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

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

```
''
```

Produce the Output

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

Produce the following output:

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

Java Basics

Basic Text Output