



Introduction

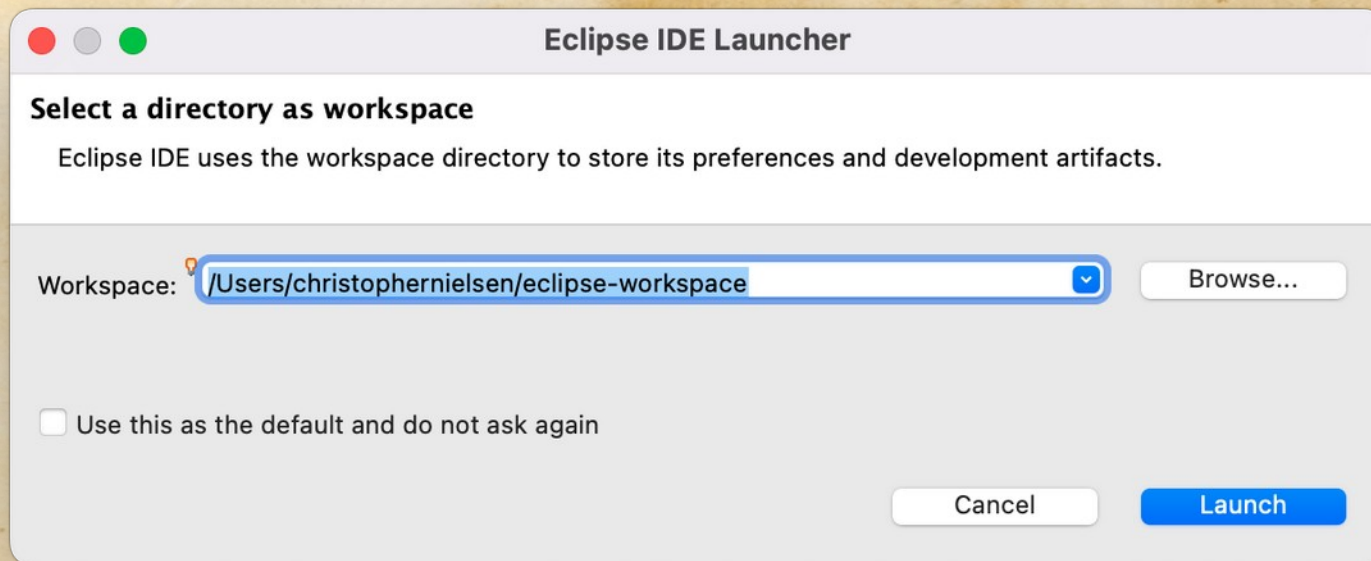
*Integrated Development Environment (IDE):
Eclipse – Hello World*

Lecture Contents

- Previously, on Installing Eclipse...
- Reset Perspective
- Hello World Program
 - Create a *project*
 - Create a *package*
 - Create a **Class**
 - `public static void main(String[] args)`
 - `System.out.println("...");`
- What about VS Code?

Previously, on Installing Eclipse...

- When you start Eclipse...
 - the default directory should be fine.
 - **this is where your code will be saved!**



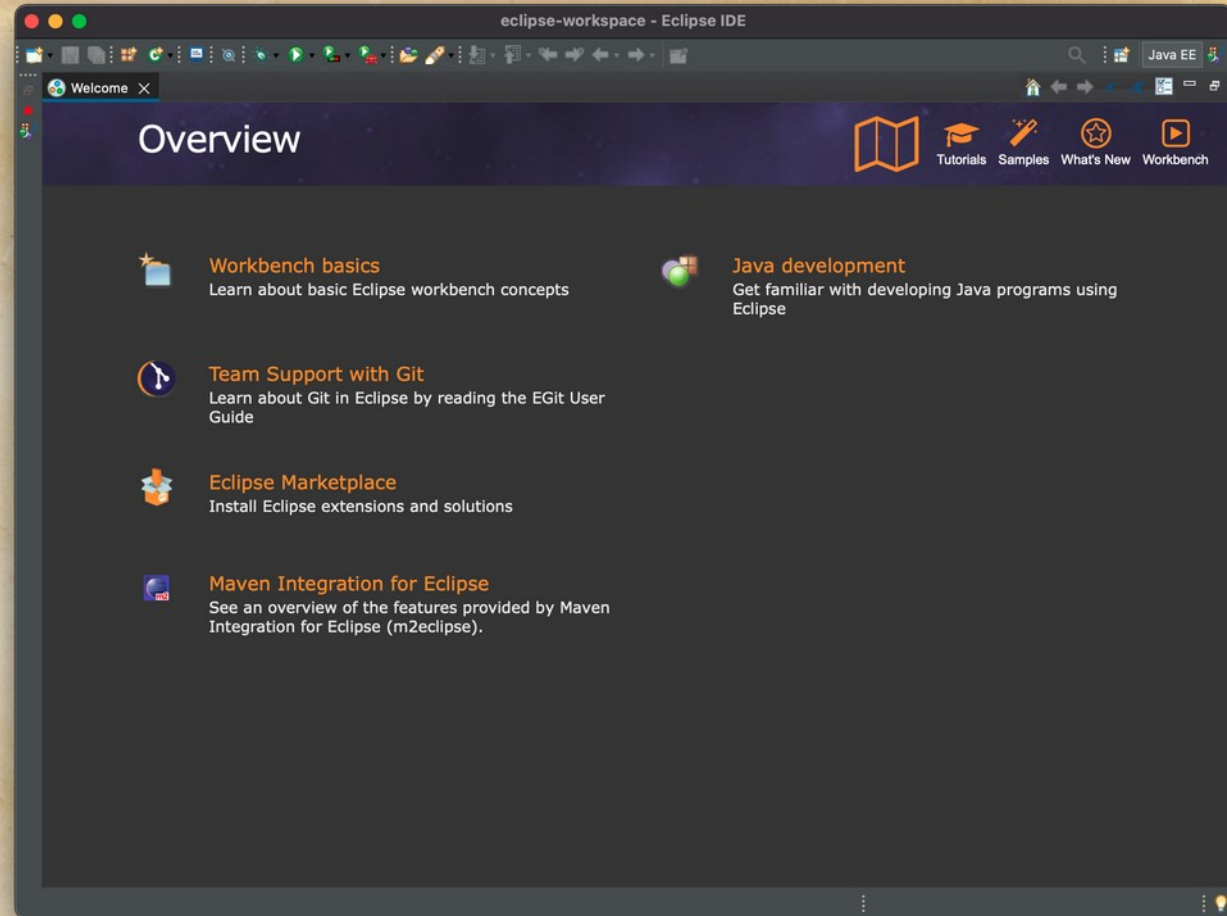
Previously, on Installing Eclipse...

- If you get here, then it seems you have Eclipse installed successfully.

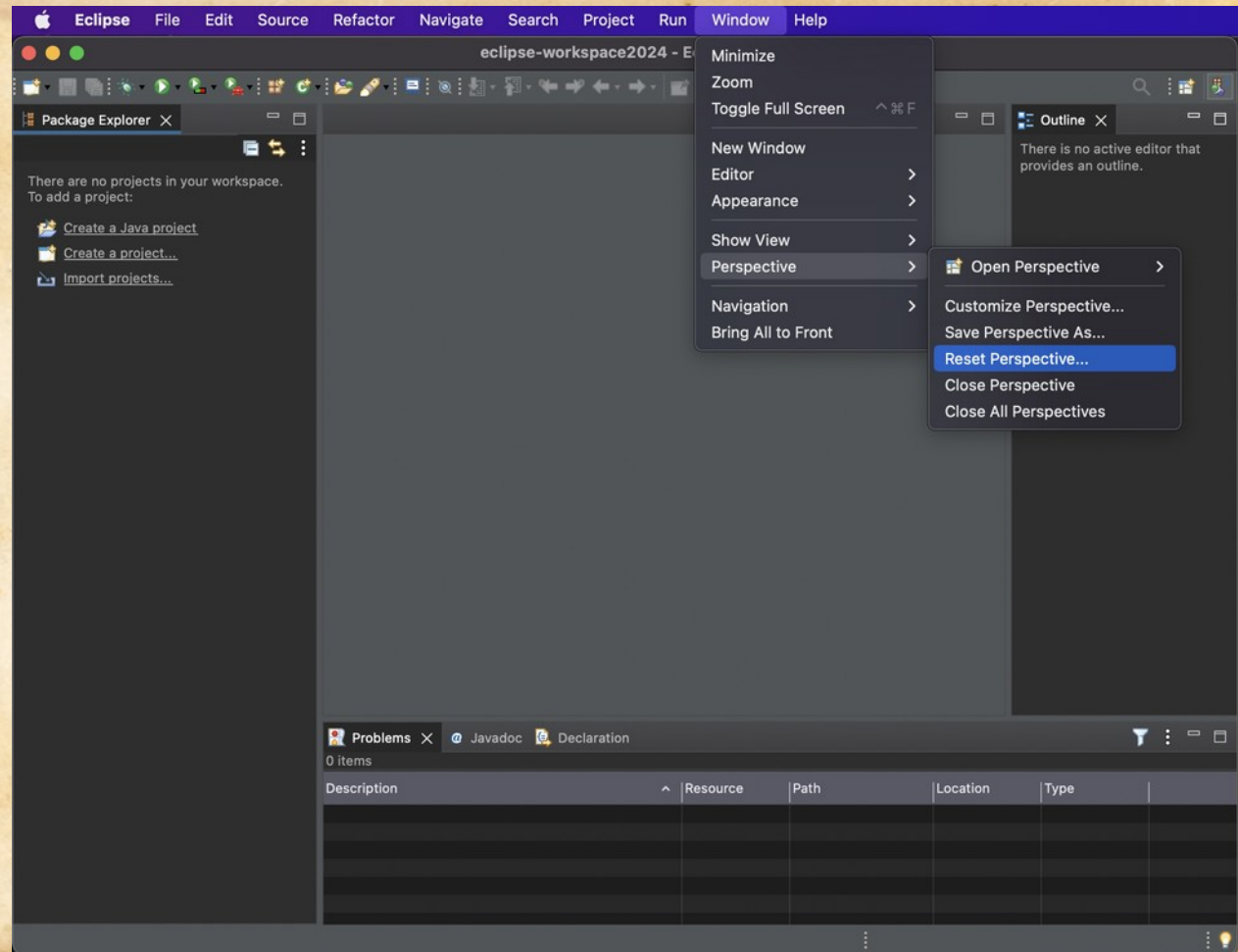


Congratulations!

- Next step is writing your first Java program!

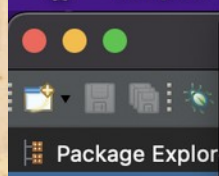


- *Hint:* If you ever lose your “**Package Explorer**” or other tabs, “**Reset Perspective...**” might restore the interface to the default settings...



Eclipse Project

- “Java Project” is an Eclipse thing, not a Java thing.
 - Simply a directory to organize your files.
- One project is probably sufficient for this entire course, so name your project for this course something like one of these:
 - APCScA
 - AP_CompSci
 - iGCSE
 - iGCSE_CS
- The next page shows where to find the menu option to create a new Java project.

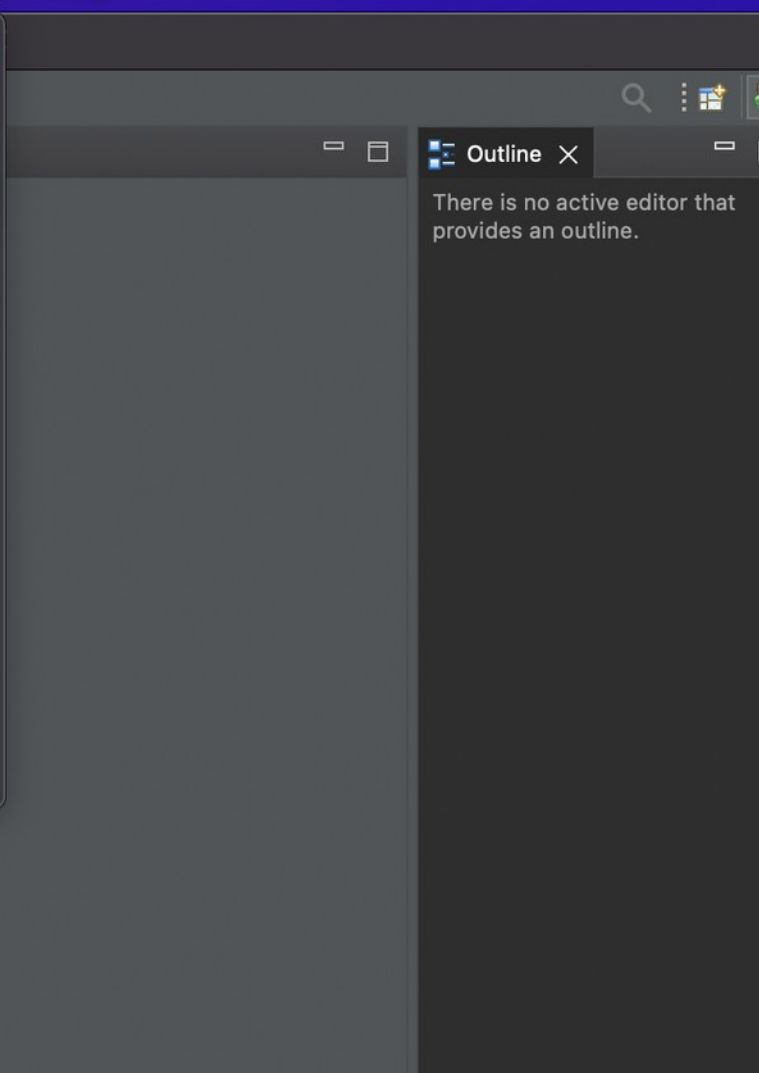


There are no projects in the workspace.
To add a project, click one of the following links:

- Create a Java Project
- Create a project from existing source code
- Import an existing project into the workspace

- New ⌘ N
- Open File...
- Open Projects from File System...
- Recent Files
- Close Editor ⌘ W
- Close All Editors ⇧ ⌘ W
- Save ⌘ S
- Save As...
- Save All ⇧ ⌘ S
- Revert
- Move...
- Rename... F2
- Refresh F5
- Convert Line Delimiters To
- Print... ⌘ P
- Import...
- Export...
- Properties ⌘ I
- Switch Workspace
- Restart

- Java Project
- Project...
- Package
- Class
- Interface
- Enum
- Record
- Annotation
- Source Folder
- Java Working Set
- Folder
- File
- Untitled Text File
- JUnit Test Case
- Example...
- Other... ⌘ N



Select an appropriate
project name;
no spaces

Probably one project
for all the code you'll
write for this entire
course.

Uncheck this box

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location

Location: [Browse...](#)

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE 'JRE [17.0.4]' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

Module

☐ Create module-info.java file

i The default compiler compliance level for the current workspace is 17. The new project will use a project specific compiler compliance

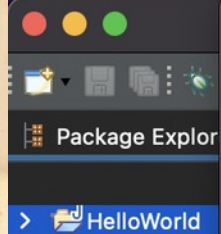
[?](#) [< Back](#) [Next >](#) [Cancel](#) [Finish](#)

Java Package

- This is an example Java package name

`com.nielsenedu.name.introjava`

- Java package names start with your domain name, reversed.
 - Code designed by a developer at “nielsenedu.com” will have Java package names starting with “com.nielsenedu”.
- The remainder of the package name should be a descriptive hierarchy. For students naming their packages, replace “*name*” with your name.
- The next page shows where to find the menu option to create a new Java package.



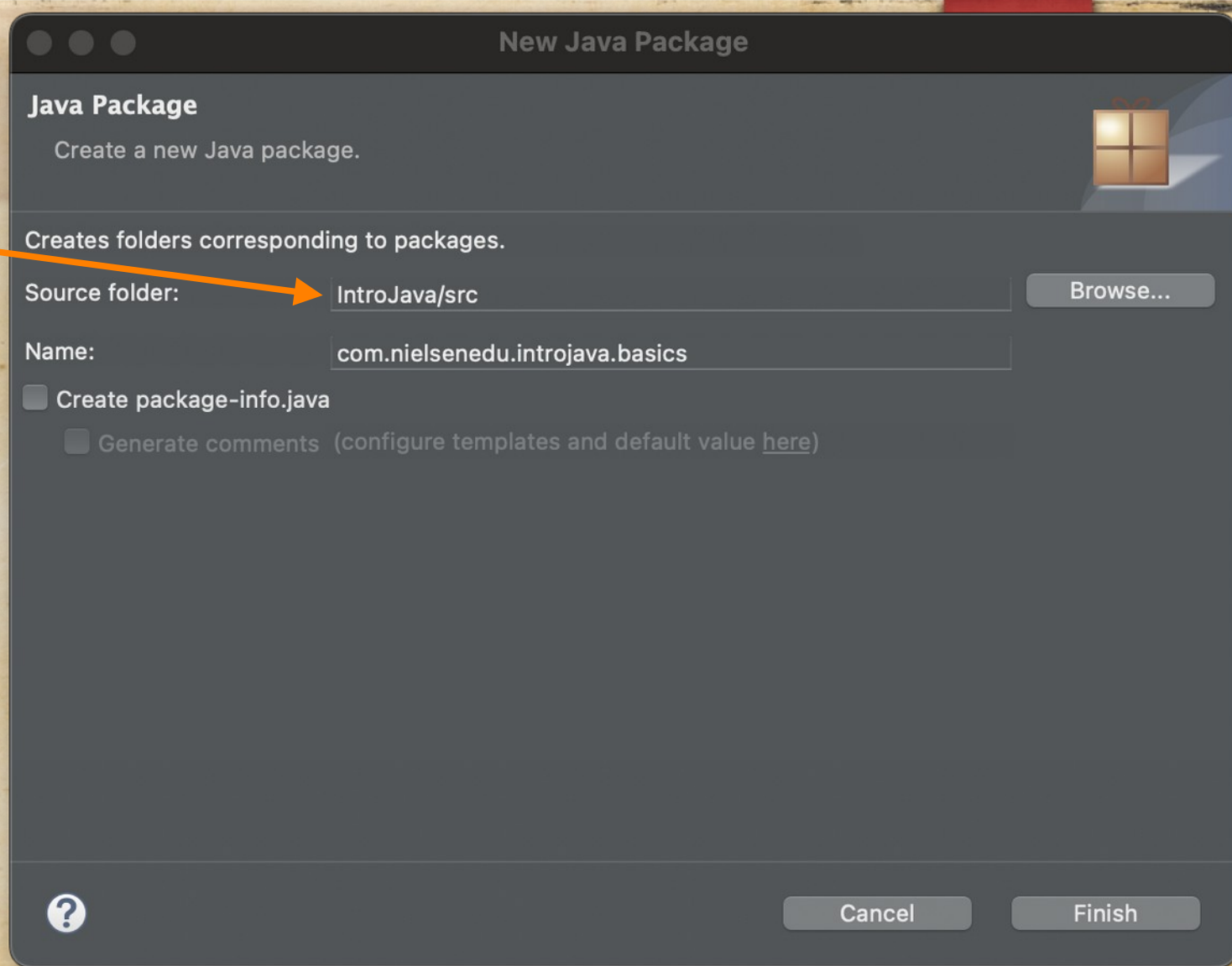
- New ⌘ N
- Open File...
- Open Projects from File System...
- Recent Files
- Close Editor ⌘ W
- Close All Editors ⇧ ⌘ W
- Save ⌘ S
- Save As...
- Save All ⇧ ⌘ S
- Revert
- Move...
- Rename... F2
- Refresh F5
- Convert Line Delimiters To
- Print... ⌘ P
- Import...
- Export...
- Properties ⌘ I
- Switch Workspace
- Restart

- Java Project
- Project...
- Package**
- Class
- Interface
- Enum
- Record
- Annotation
- Source Folder
- Java Working Set
- Folder
- File
- Untitled Text File
- JUnit Test Case
- Example...
- Other... ⌘ N

Outline X

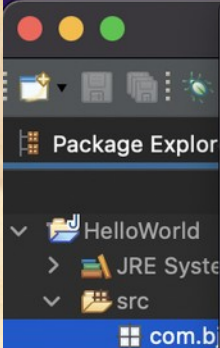
There is no active editor that provides an outline.

If this is not automatically filled, ensure the project name is highlighted before creating a new Java Package so Eclipse knows where you want to create the package.



Java Class

- Java programs are organized into *classes*
- Each Java file contains one class.
- Classes are named using camel case (no spaces and the first letter of each word is capitalized). for example:
 - HelloWorld
 - BasicOutput
 - Graphing
- The next pages show where to find the menu option to create a new Java Class, and naming a class HelloWorld.



New  ⌘ N >

Open File...

 Open Projects from File System...

Recent Files >

Close Editor ⌘ W

Close All Editors ⇧ ⌘ W

 Save ⌘ S

 Save As...

 Save All ⇧ ⌘ S

Revert

Move...

 Rename... F2

 Refresh F5

Convert Line Delimiters To >

 Print... ⌘ P

 Import...

 Export...

Properties ⌘ I

Switch Workspace >

Restart

 Java Project

 Project...

 Package

 **Class**

 Interface

 Enum

 Record

 Annotation

 Source Folder

 Java Working Set

 Folder

 File

 Untitled Text File

 JUnit Test Case

 Example...

 Other... ⌘ N

Outline ×

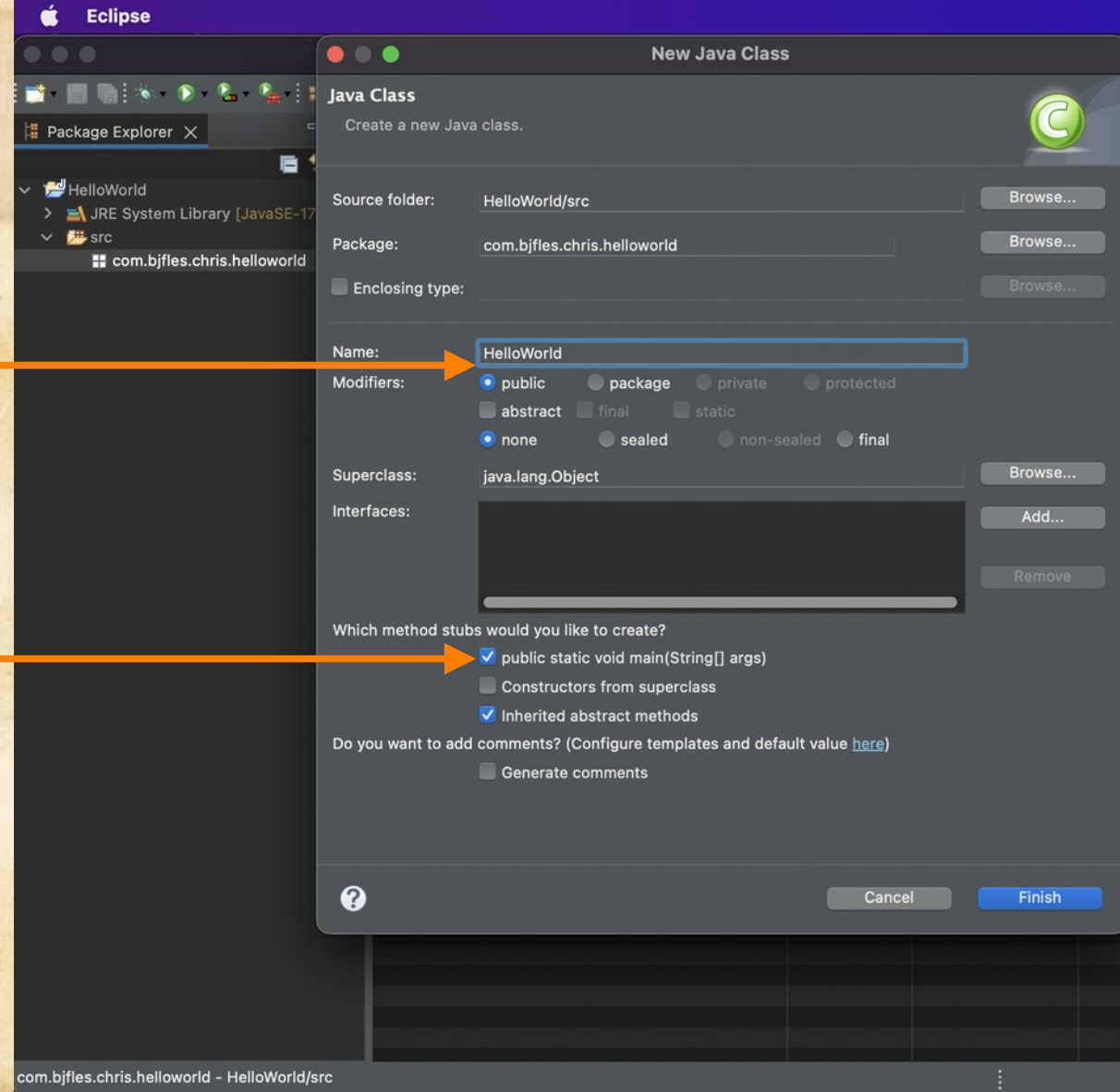
There is no active
provides an outlin

Java Class

The naming convention for a Java class is **Pascal case**, also known as **upper camel case**.

There are no spaces, and the first letter of each word is upper case

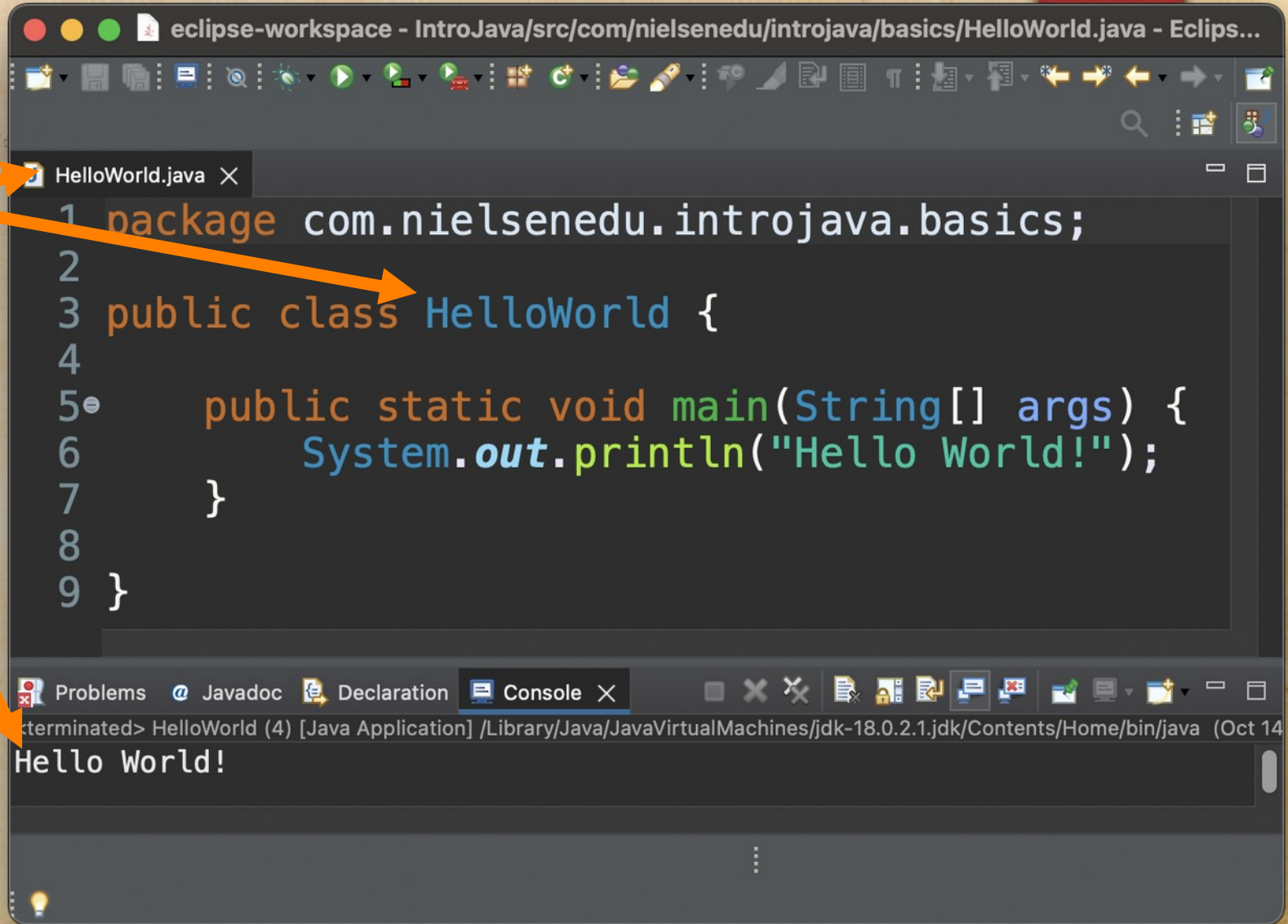
Selecting this will add template code for the **main** method, which is where the Java program will start running.



Java Class

Filename and
package name
match.

The program runs.



The screenshot shows the Eclipse IDE interface. The top toolbar contains various icons for file operations, editing, and running. The editor window displays the code for HelloWorld.java, which is highlighted in the top-left corner. The code is as follows:

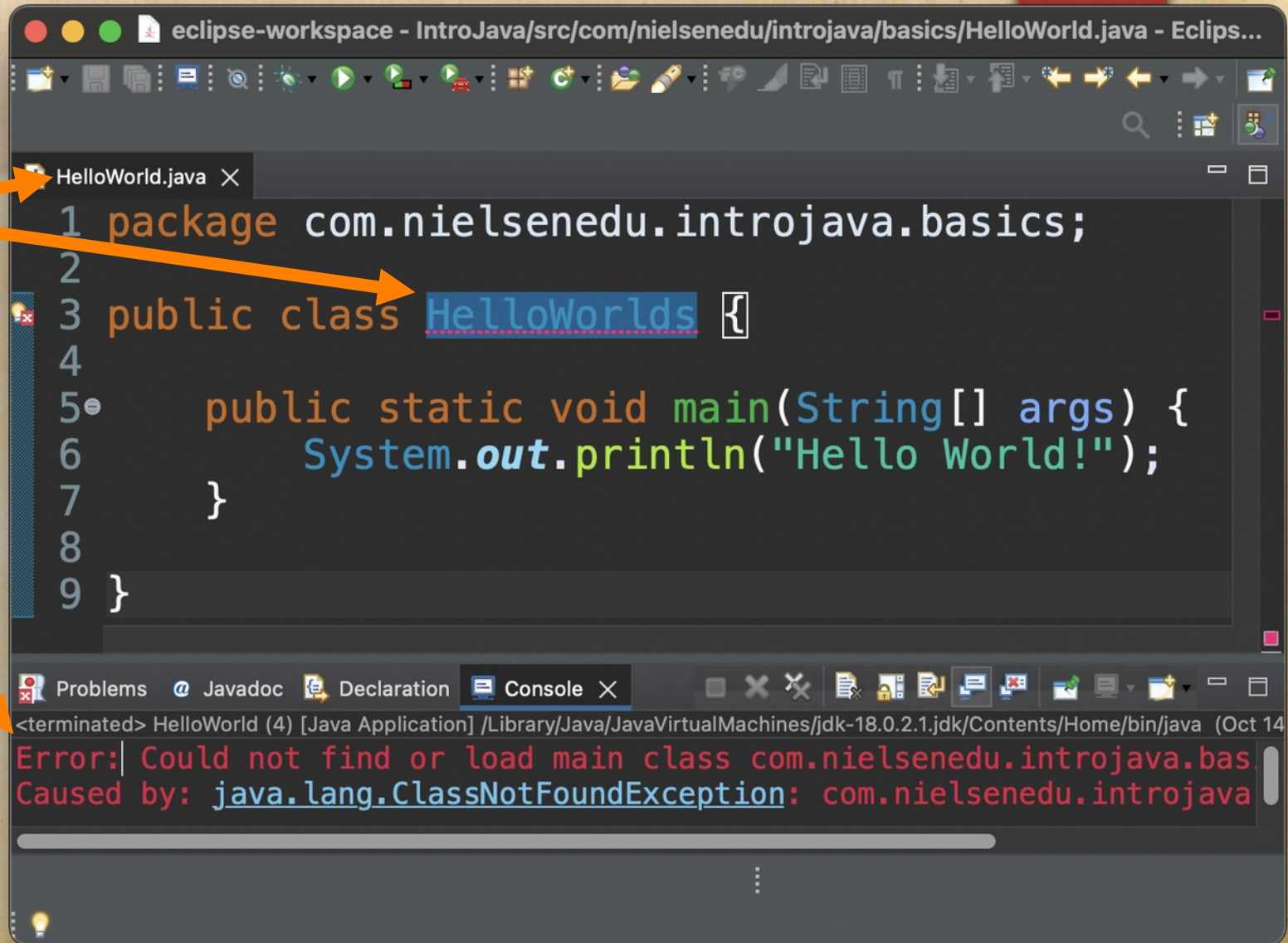
```
1 package com.nielsenedu.introjava.basics;  
2  
3 public class HelloWorld {  
4  
5     public static void main(String[] args) {  
6         System.out.println("Hello World!");  
7     }  
8  
9 }
```

Below the editor, the 'Console' tab is active, showing the output of the program: 'Hello World!'. The status bar at the bottom indicates the program was terminated successfully.

Java Class

Filename and
package name
do **NOT** match.

The program will
not run.



The screenshot shows the Eclipse IDE with a Java file named `HelloWorld.java`. The code defines a package `com.nielsenedu.introjava.basics` and a public class `HelloWorlds`. The `main` method prints "Hello World!". An orange arrow points from the text "Filename and package name do NOT match." to the package name and the class name. Another orange arrow points from "The program will not run." to the error message in the console.

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

Problems | Javadoc | Declaration | Console

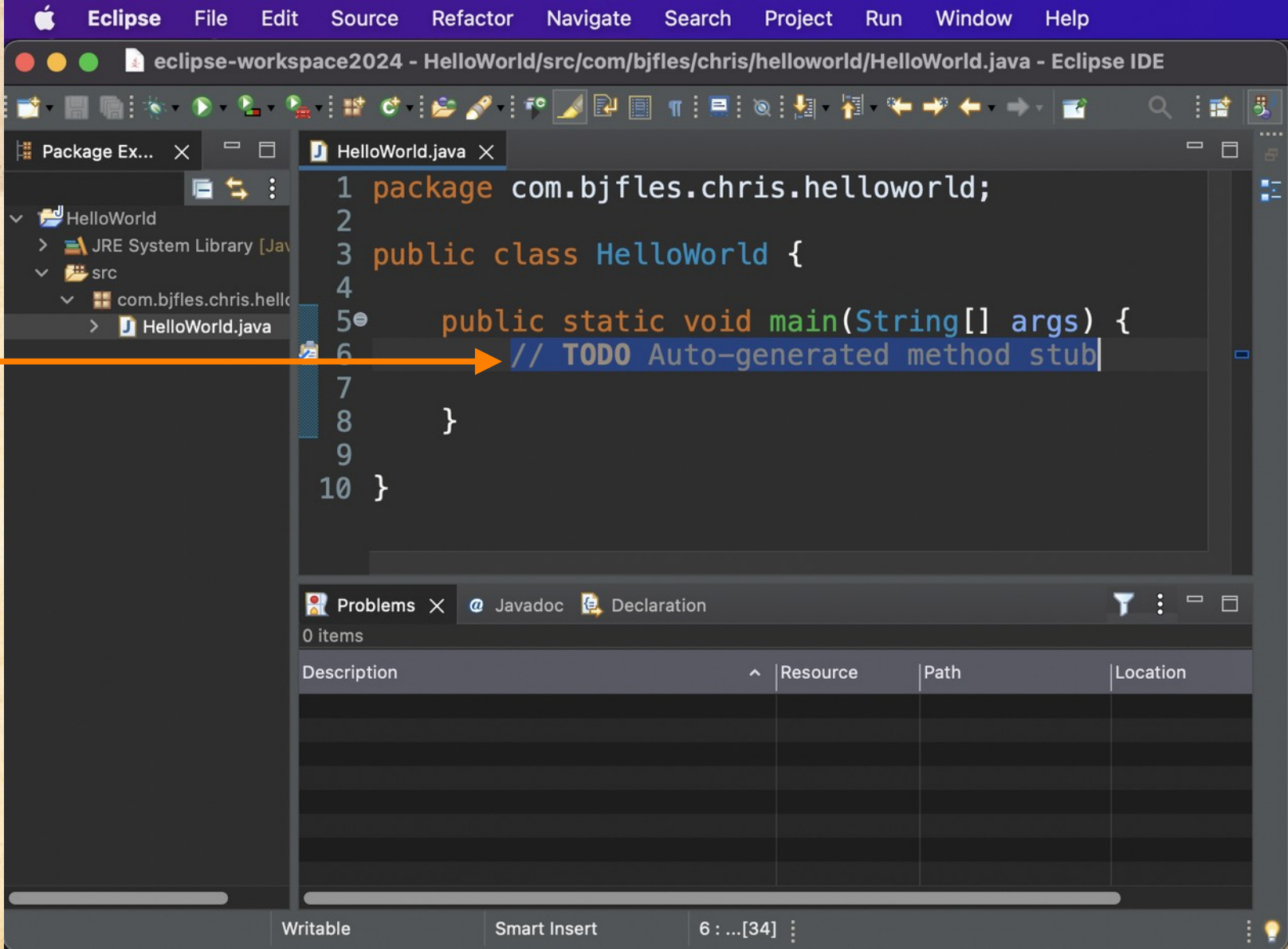
<terminated> HelloWorld (4) [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java (Oct 14)

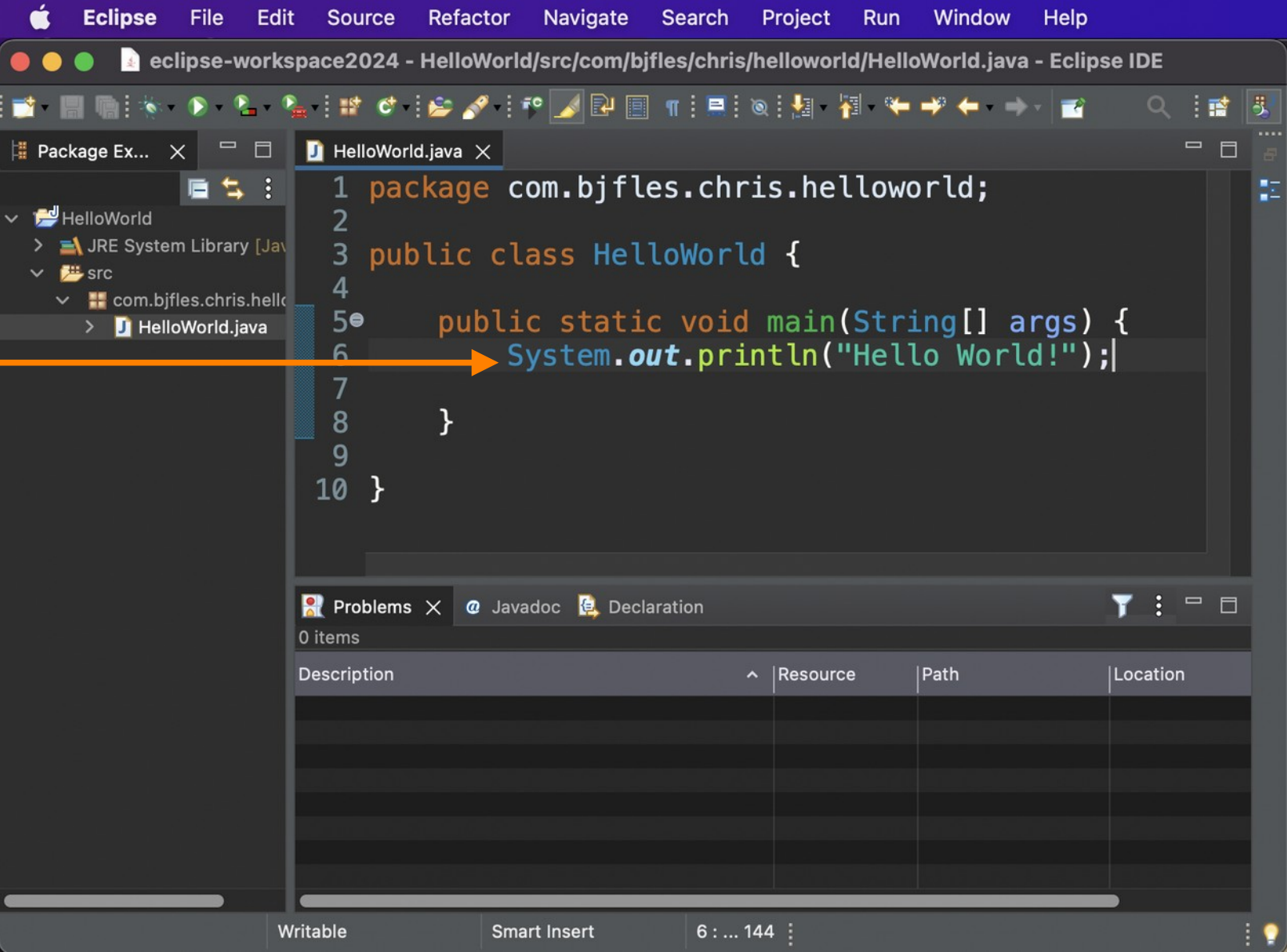
Error: Could not find or load main class com.nielsenedu.introjava.bas.
Caused by: java.lang.ClassNotFoundException: com.nielsenedu.introjava

Java Class

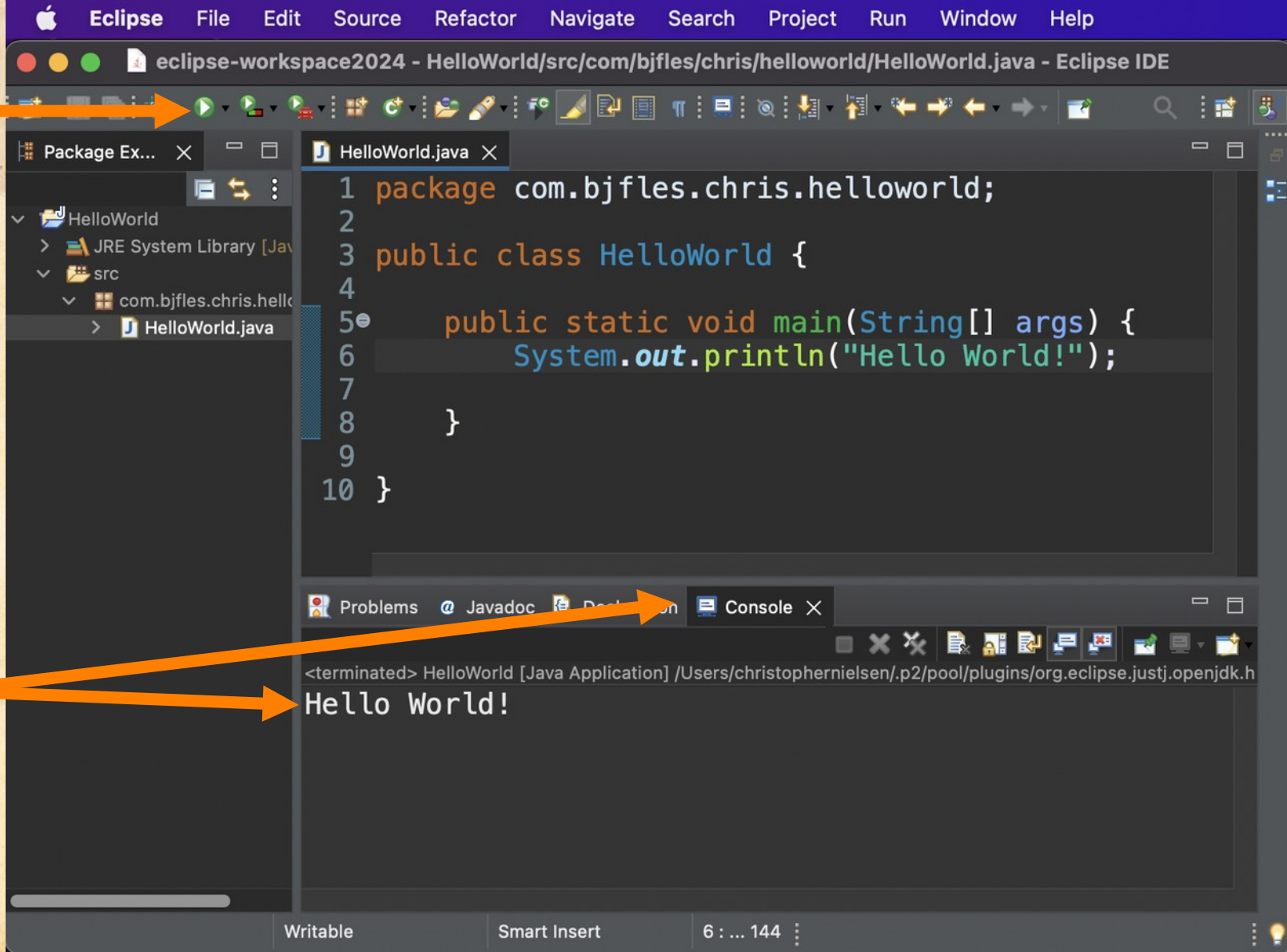
- Java programs are organized into *classes*
- Each Java file contains one class.
- Java classes are named using camel case; examples:
 - HelloWorld
 - BasicOutput.
- The filename must be the same as the class name, for example:
 - Inside HelloWorld.java there will be class HelloWorld

Replace
line 6...





Click here
to run.



Check the
output in the
console.

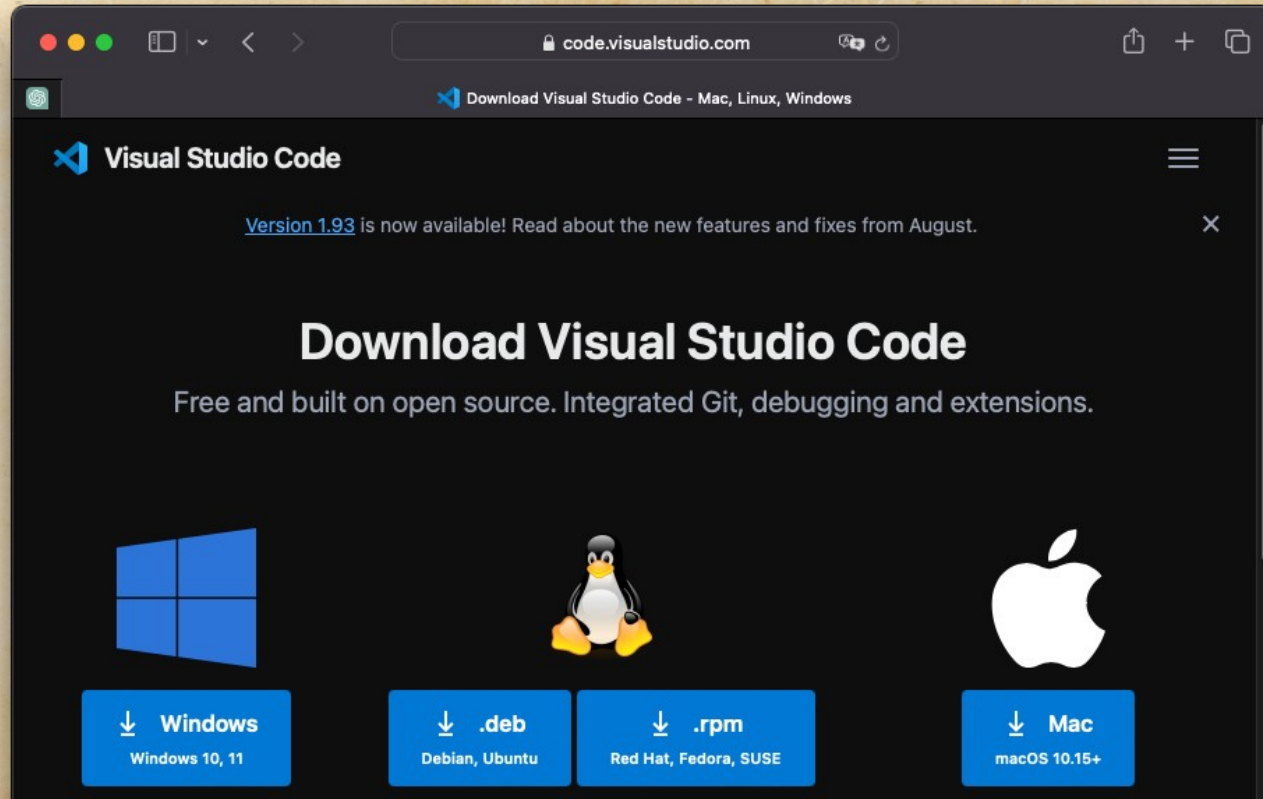
Lecture Contents

- Previously, on Installing Eclipse...
- Reset Perspective
- Hello World Program
 - Create a *project*
 - Create a *package*
 - Create a **Class**
 - `public static void main(String[] args)`
 - `System.out.println("...");`

- **What about VS Code?**

What about VS Code?

- If you downloaded VS Code: ***code.visualstudio.com***



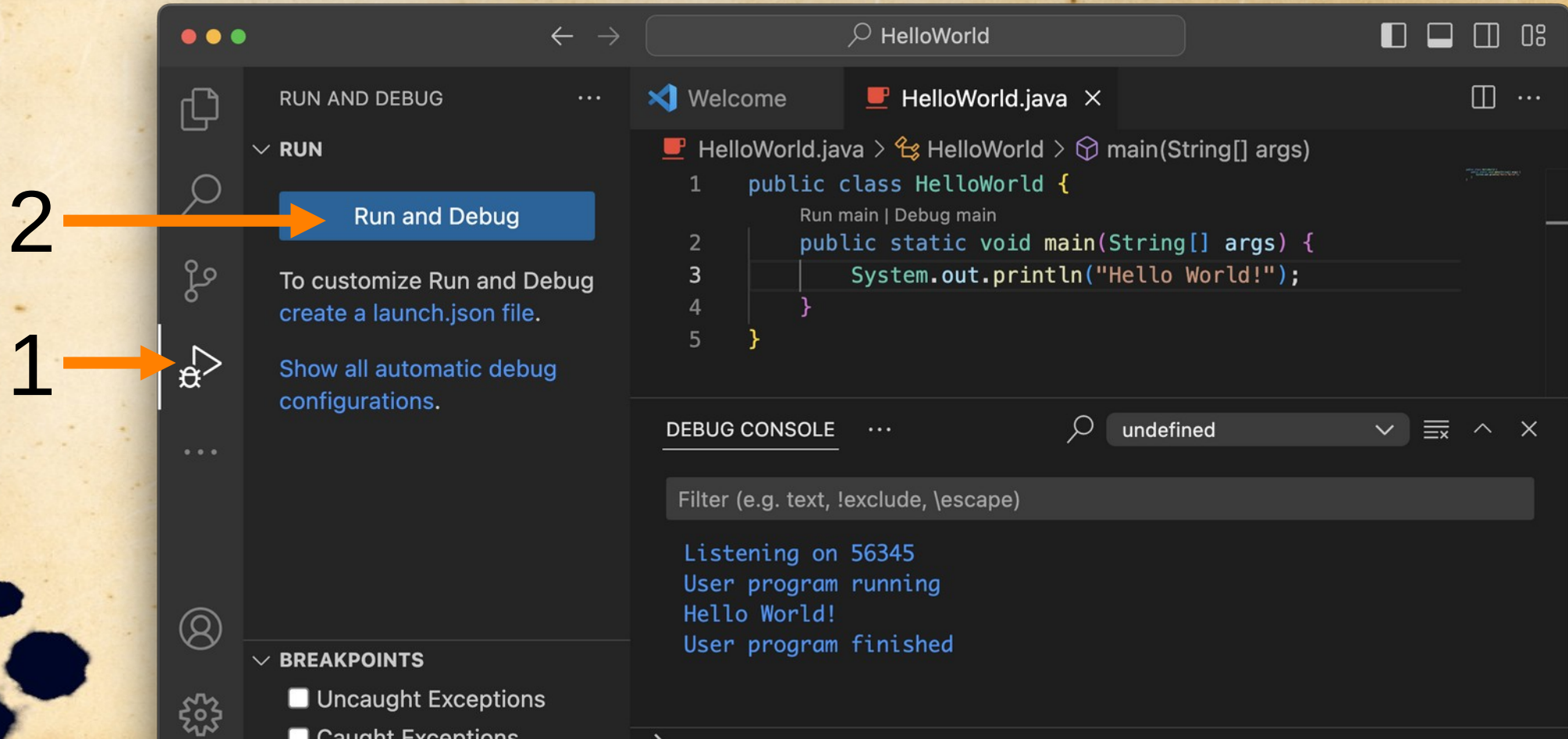
What about VS Code?

You will need to either install the JDK or an extension such as this one:



A screenshot of the Visual Studio Code interface. The top bar shows the search bar with "HelloWorld" and window controls. The left sidebar contains the "EXTENSIONS: MARKETPLACE" view with a search filter "@category:debugger". The main area displays the "Java" extension by Oracle Corporation, version v22.1.2, with a download count of 1,068,258. The extension description states it is the "Java Platform Extension for Visual Studio Code" and includes an "Install" button, an "Auto Update" checkbox, and a settings gear. Below the extension name are tabs for "DETAILS", "FEATURES", and "CHANGELOG". The "DETAILS" tab is active, showing the title "Java Platform Extension for Visual Studio Code" and a description: "Java Platform extension from Oracle brings full featured development support (edit-compile-debug & test cycle) to VS Code. It offers support for Maven and Gradle projects." On the right side, there is a "Categories" section with buttons for "Programming Languages", "Snippets", "Linters", "Debuggers", "Formatters", and "Testing". The bottom of the interface shows a scroll bar and a "Resources" section.

What about VS Code?





Introduction

*Integrated Development Environment (IDE):
Eclipse – Hello World*