Eclipse IDE Tutorial

1. What is Eclipse? Why is it important?

https://www.eclipse.org/home/whatis/

Eclipse IDE is one of the most popular Integrated Development Environments (IDEs) for Java programming. Launched in 2001, Eclipse has grown to become a versatile tool for Java development and other languages such as C++, Python, and PHP, thanks to its extensive plugin ecosystem. It's an open-source platform, which means it is freely available and constantly being improved by a global community of developers.

Key Features of Eclipse:

- **Code Suggestions**: Eclipse helps to speed up development by offering intelligent code suggestions and auto-completion, reducing the time spent typing and searching for correct syntax.
- **Project Management**: It provides tools to organize code into projects, manage multiple files, and keep track of dependencies, making it easier to manage large codebases.
- **Powerful Debugging**: The integrated debugger allows developers to set breakpoints, step through code line by line, and inspect variables, which is essential for identifying and fixing bugs.

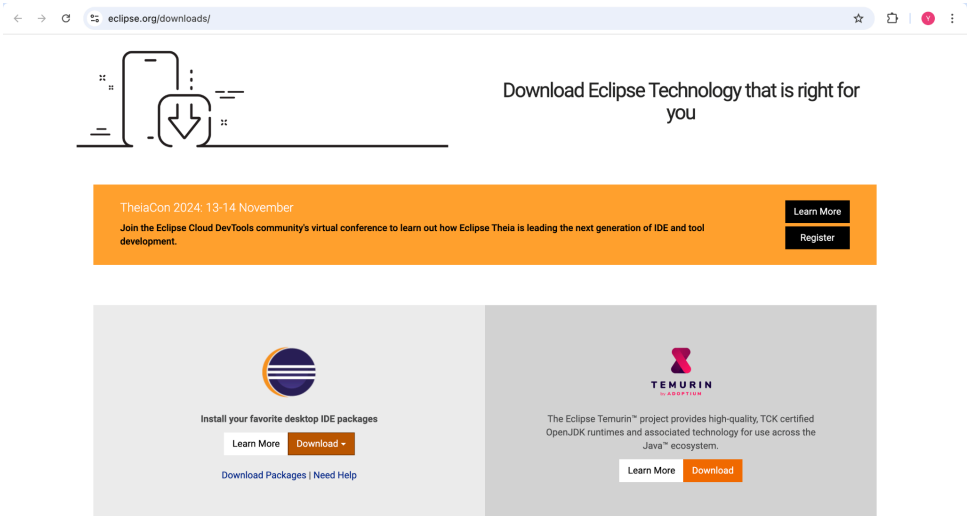
Why is Eclipse Important?

- **Versatility and Extensibility**: Eclipse's plugin-based architecture allows developers to customize and extend the IDE's functionality. This means you can tailor the environment to meet your specific development needs, whether you are working with Java, web development, or even data science.
- **Wide Adoption and Community Support**: With **over 15 million downloads annually**, Eclipse is one of the most widely used IDEs globally. This widespread use ensures strong community support, which means a wealth of online tutorials, forums, and resources to help you troubleshoot issues and improve your skills.
- **Industry Standard**: While there are other popular Java IDEs, such as IntelliJ IDEA, Eclipse remains a top choice for many developers, especially in educational settings and enterprise environments. Its familiarity and open-source nature make it an essential tool for students and professionals alike. Learning Eclipse equips you with skills that are highly transferable to real-world projects and job environments.

2. Setting Up Eclipse for Java Development

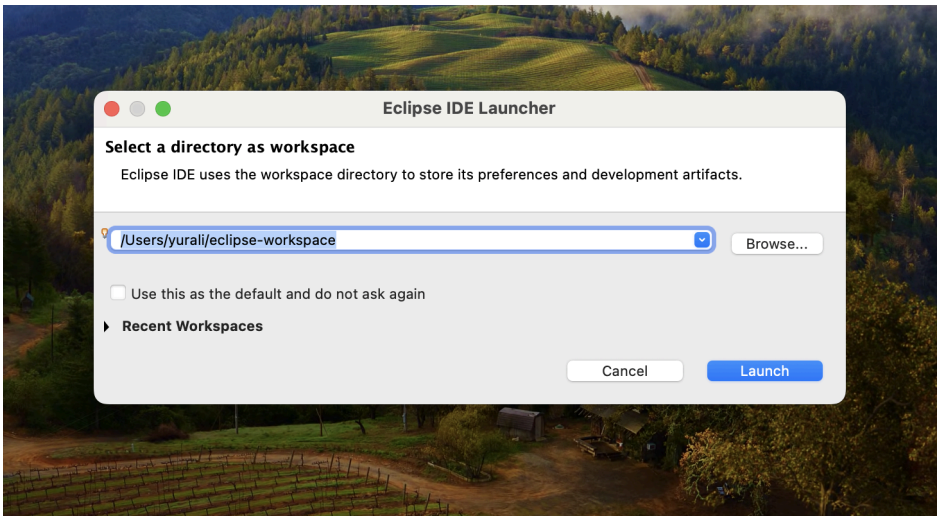
I. Download Eclipse

Go to the [Eclipse official website] and download the installer for your operating system. Follow the instructions to install.



II. Open Eclipse

After installation, launch Eclipse and select a workspace folder. The workspace is where all your projects will be stored.



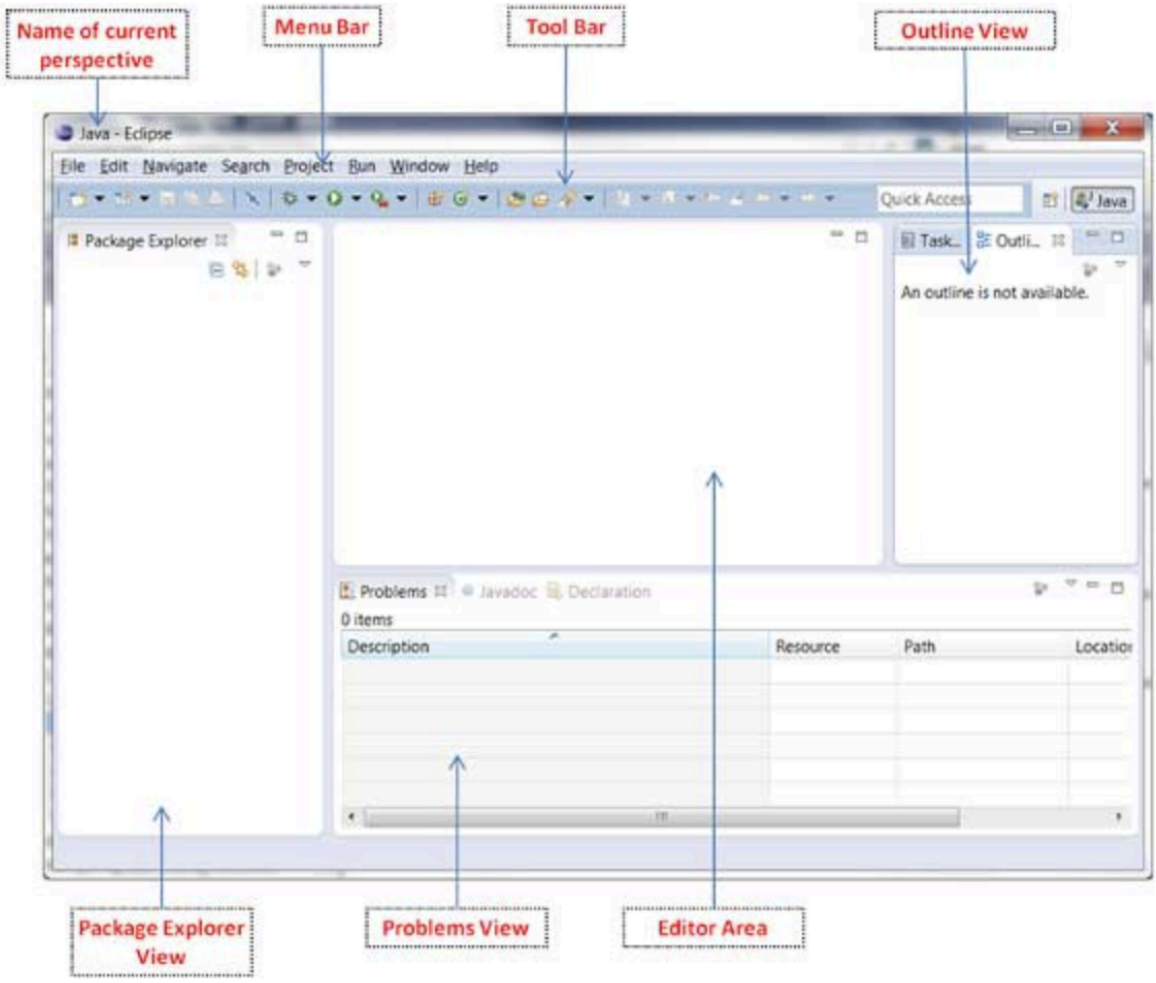
When Eclipse starts up for the first time it prompts you for the location of the workspace folder. All your data will be stored in the workspace folder. You can accept the default or choose a new location.

If you select "Use this as the default and do not ask again", this dialog box will not come up again. You can change this preference using the Workspaces Preference Page.

III. Overview of the Eclipse Interface

Familiarize yourself with the key areas:

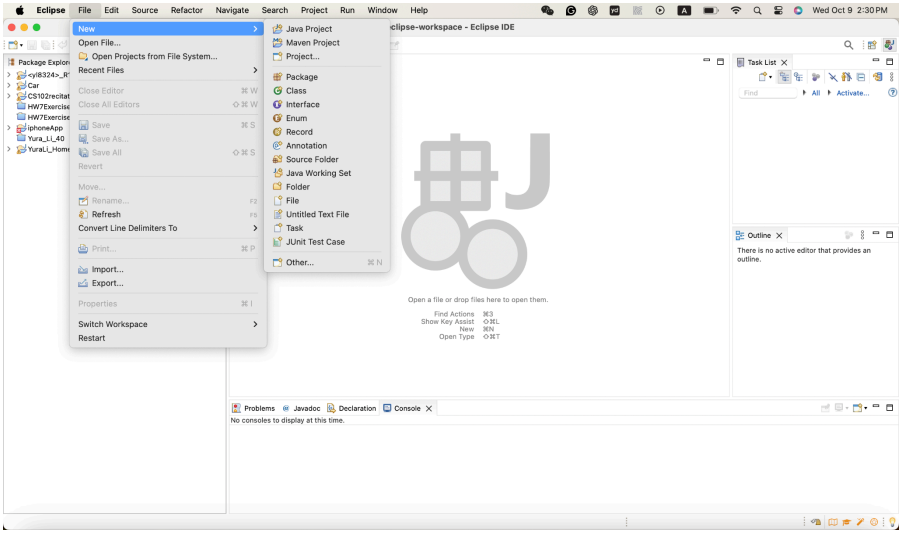
- Views
- Editors
- Menu Bar
- Tool Bar



3. Creating a New Java Project (With a Package)

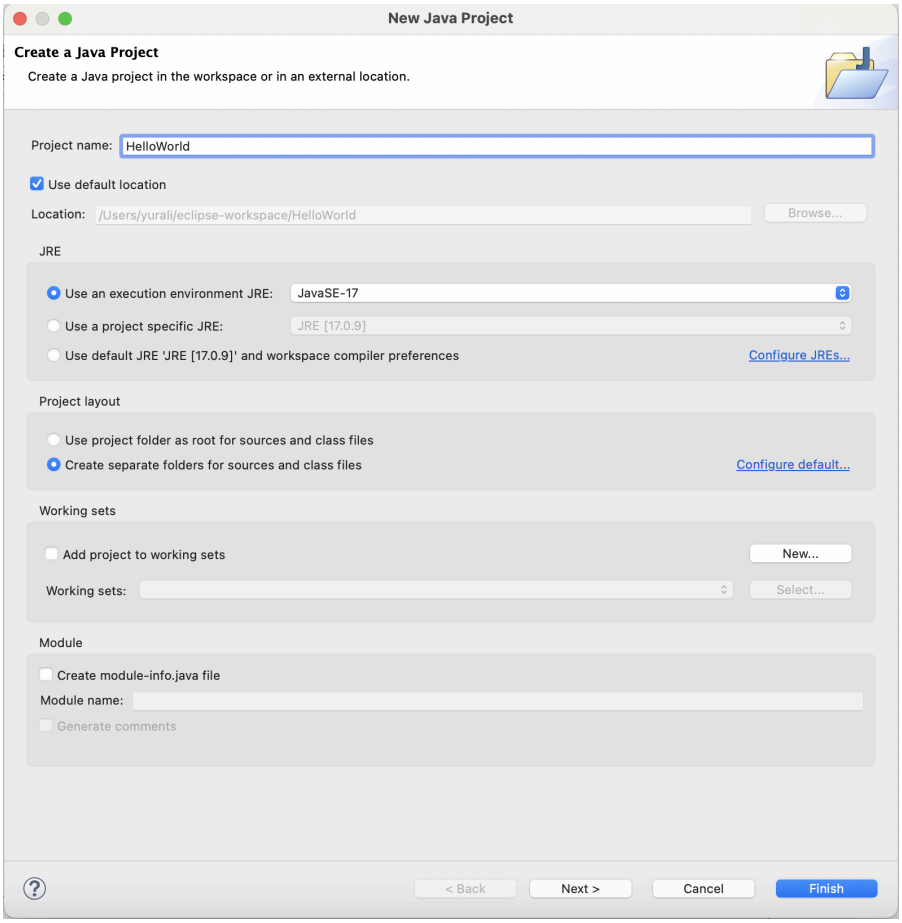
I. Create a Project

- Go to File → New → Java Project.

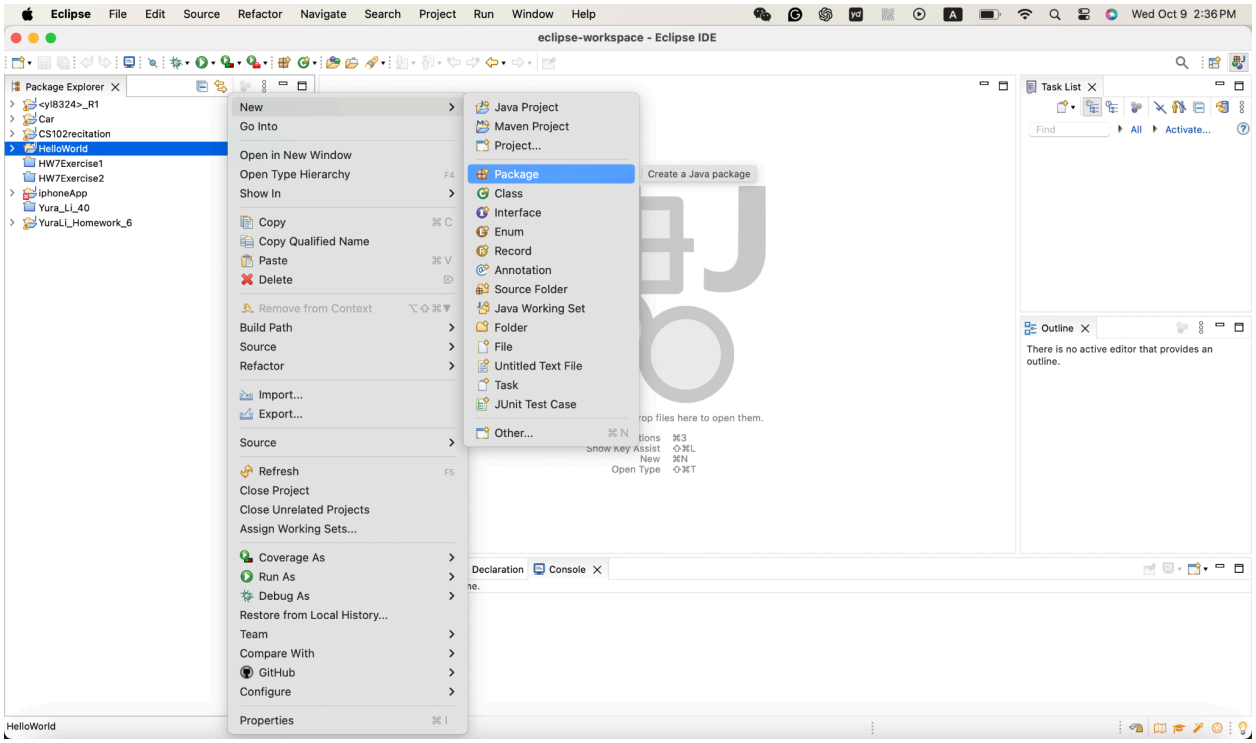


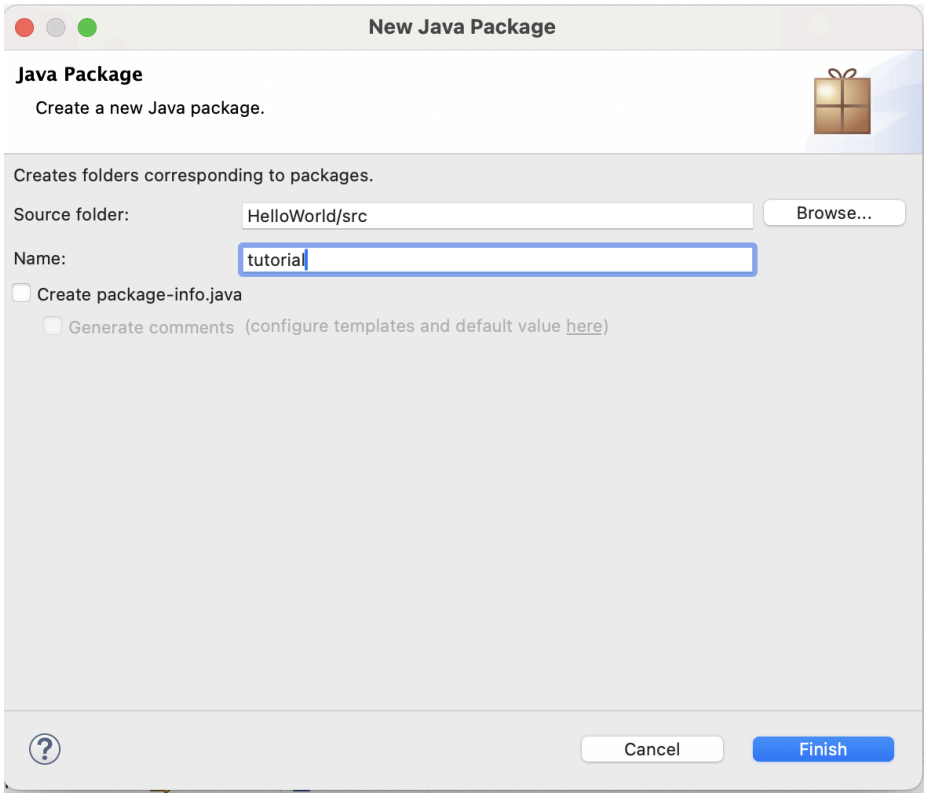
- Name your project (e.g., “HelloWorld”).

II. Creating Packages



- Right-click on src in the Project Explorer, select New → Package.
- Name the package.

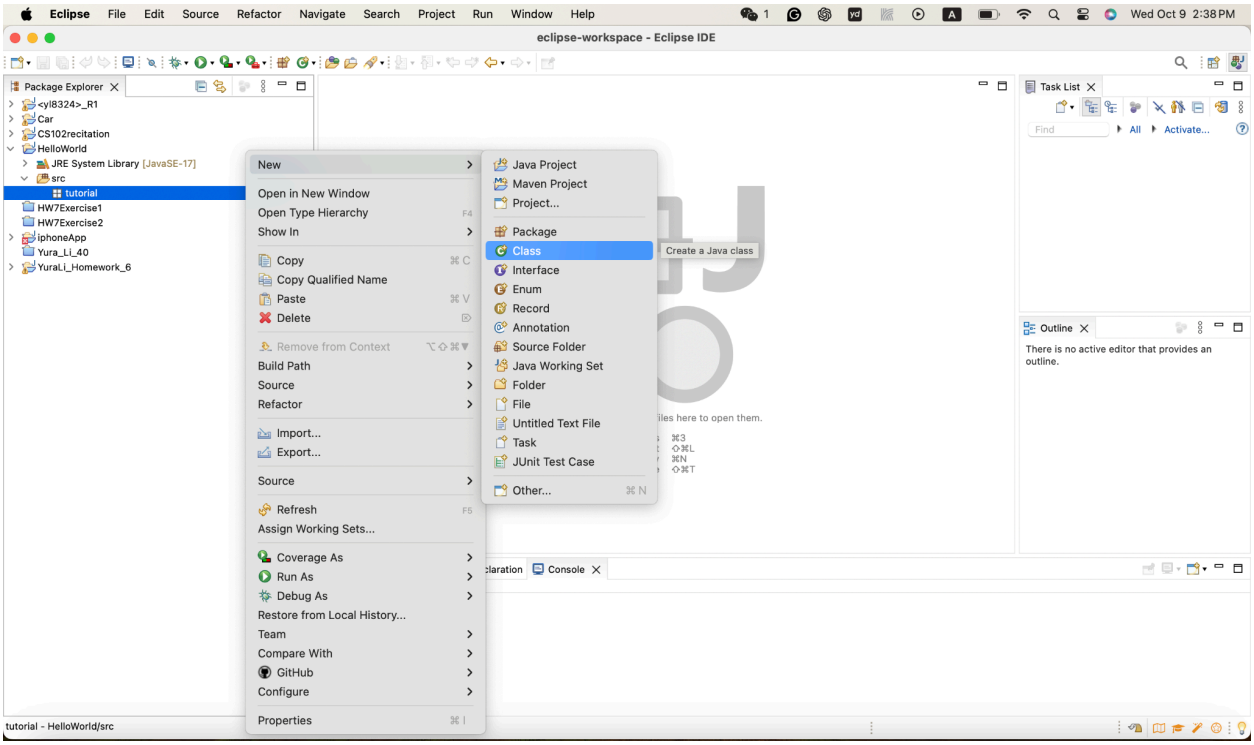




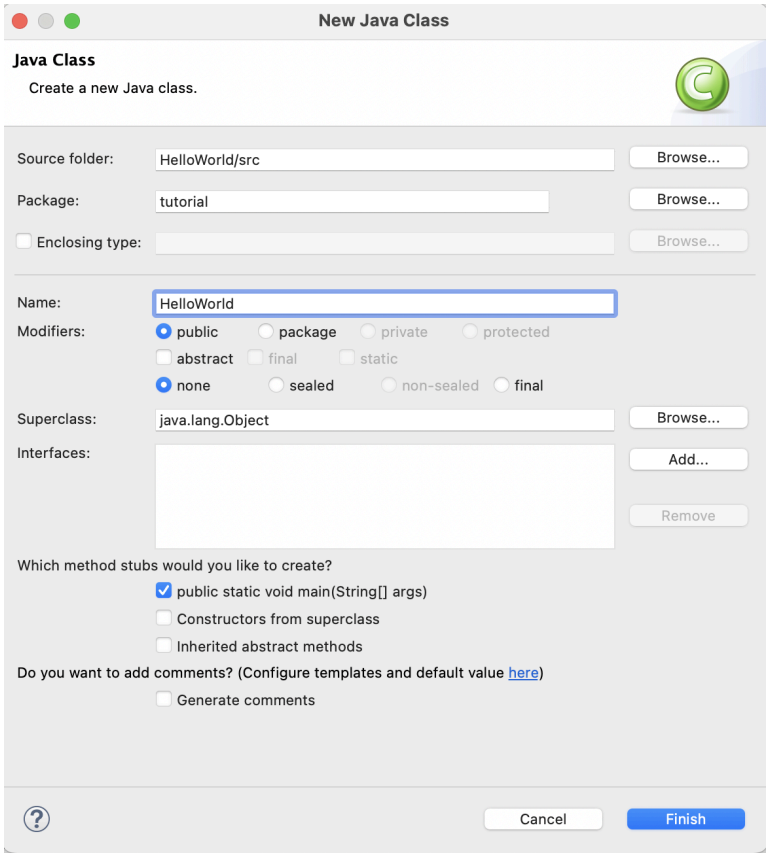
4. Writing and Running a Simple Java Program

I. Creating a New Class

- Right-click on the package you created, select New → Class.

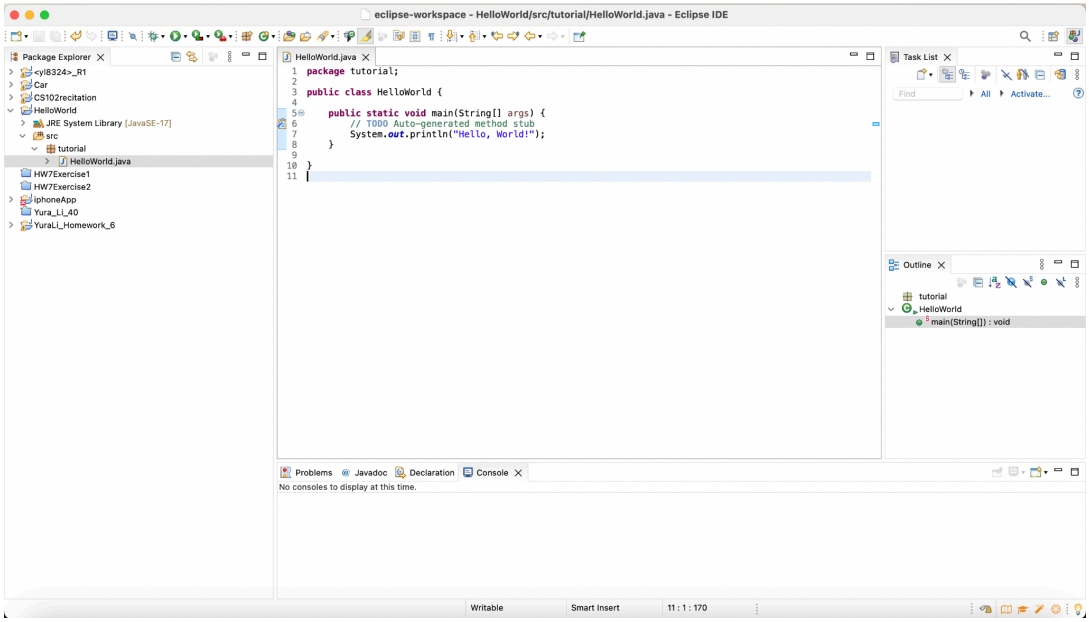


- Name it HelloWorld, check the box for public static void main(String[] args).



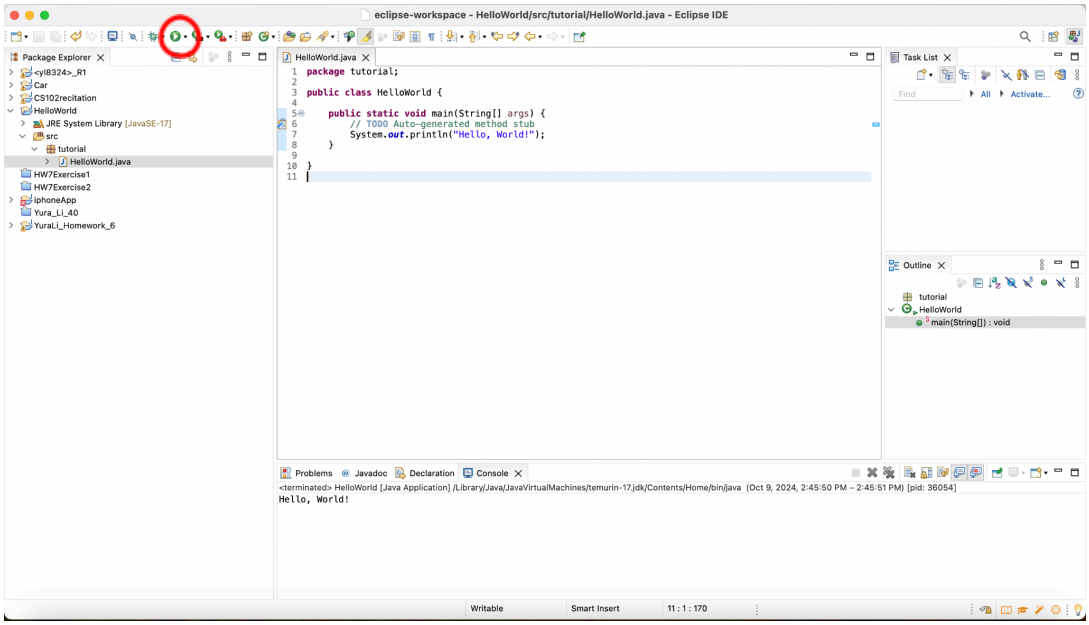
II. Writing the Code

Replace the default content with the following code (Remember always save your code before executing):

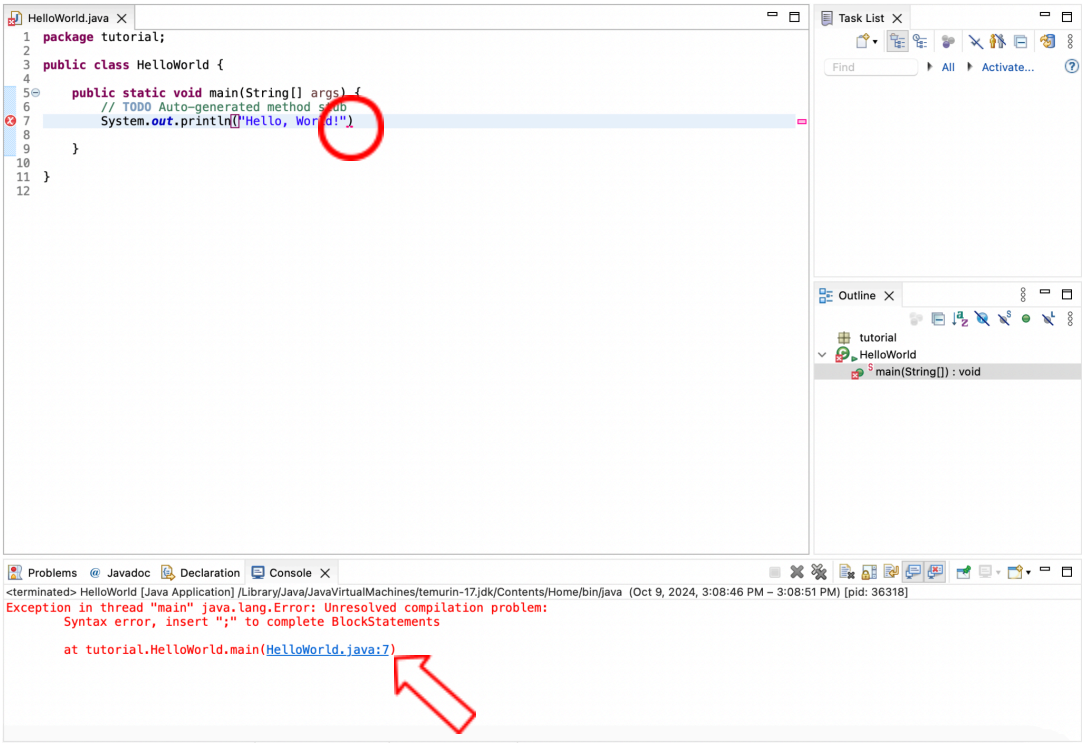


III. Running the Program

- Click the green play button on the toolbar or right-click on the class → Run As → Java Application.



IV. Check the error and debug

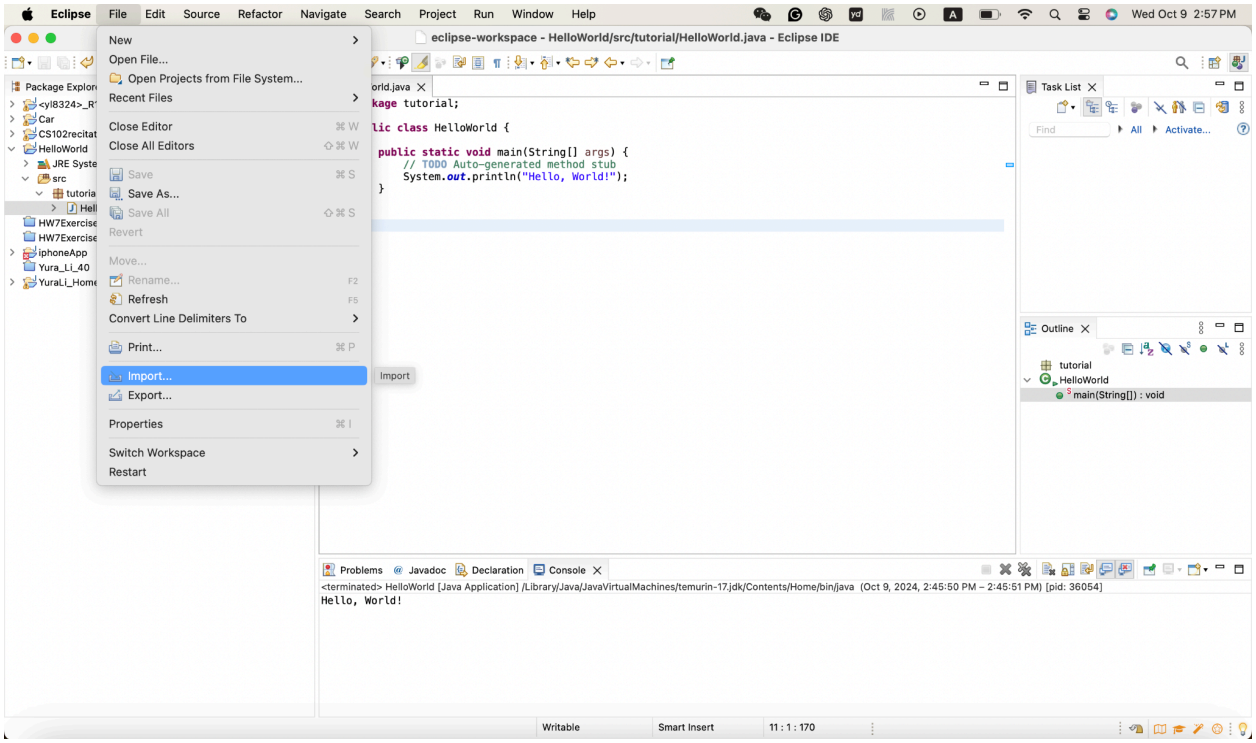


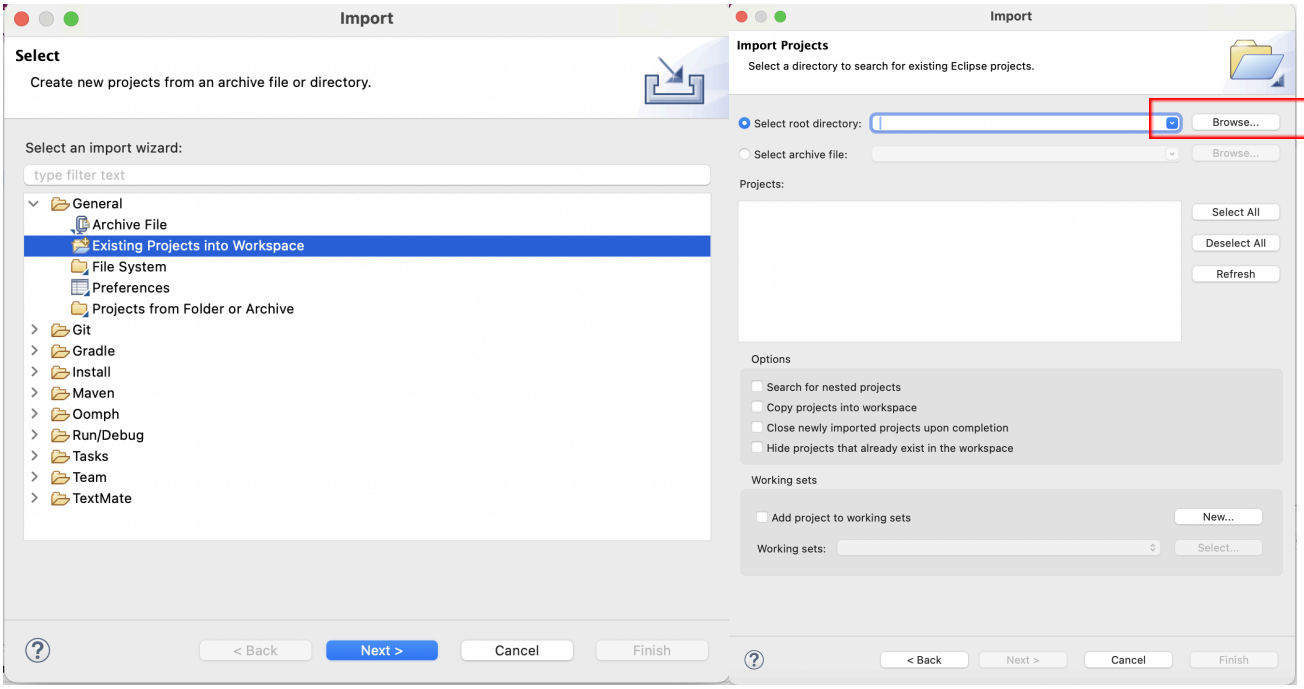
Eclipse will underline syntax errors such as the underlined above. However, runtime error/ logic error is unable to be detected unless the program is executed. You will have to target the line (listed in the console) where the error occurs and fix it.

5. Importing the Project in Eclipse

Before beginning component development, you have to import the component project you just created into Eclipse.

Within Eclipse, choose **File** > **Import** > **General** > **Existing Projects into Workspace**.

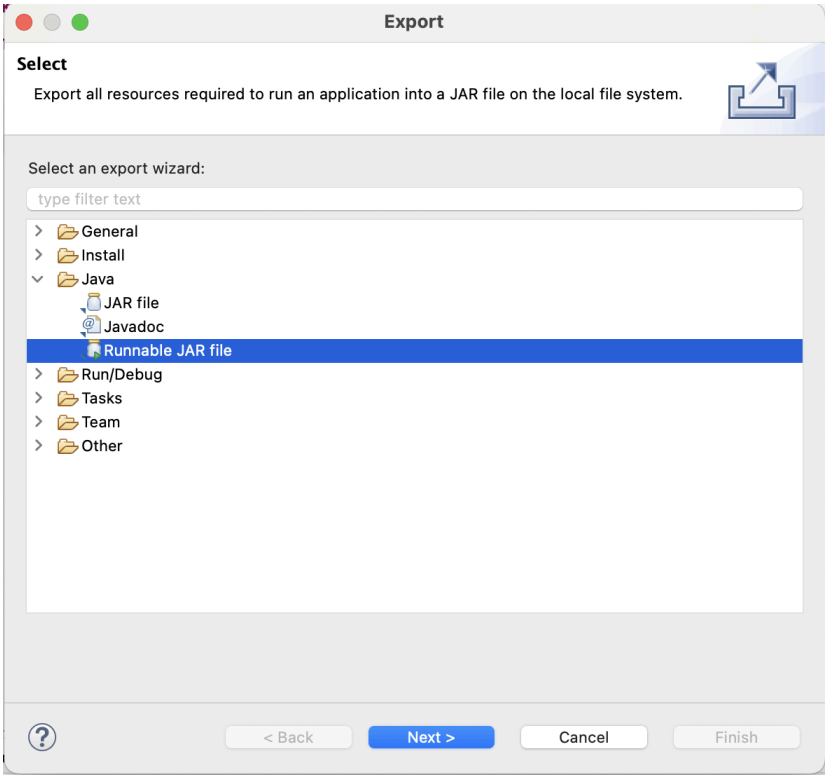




6. Exporting and Sharing Java Projects

I. Exporting a Project

- Right-click the project → Export → Java → Runnable JAR File.



- Select the Launch Configuration, choose the export destination, and finish.

II. Sharing Projects

- You can also zip the project folder and share it via email or upload it to your Learning Management System (LMS).

7. Troubleshooting Common Issues

I. Windows

- Eclipse Crashes or Won't Start:
  - Ensure you are using the correct version of Java (e.g., Eclipse 64-bit with Java 64-bit). https://help.eclipse.org/latest/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Ftasks%2Ftask-add_new_jre.htm
  - Update your Java Runtime Environment (JRE) and ensure the JAVA_HOME environment variable is set correctly. https://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/#:~:text=Set%20JAVA_HOME:,Program%20Files%5CJava%5Cjdk1.

- Java Build Path Errors:
  - Right-click on the project → Build Path → Configure Build Path. Check if all libraries are correctly listed under Libraries.
  - Clean the project: Project → Clean.

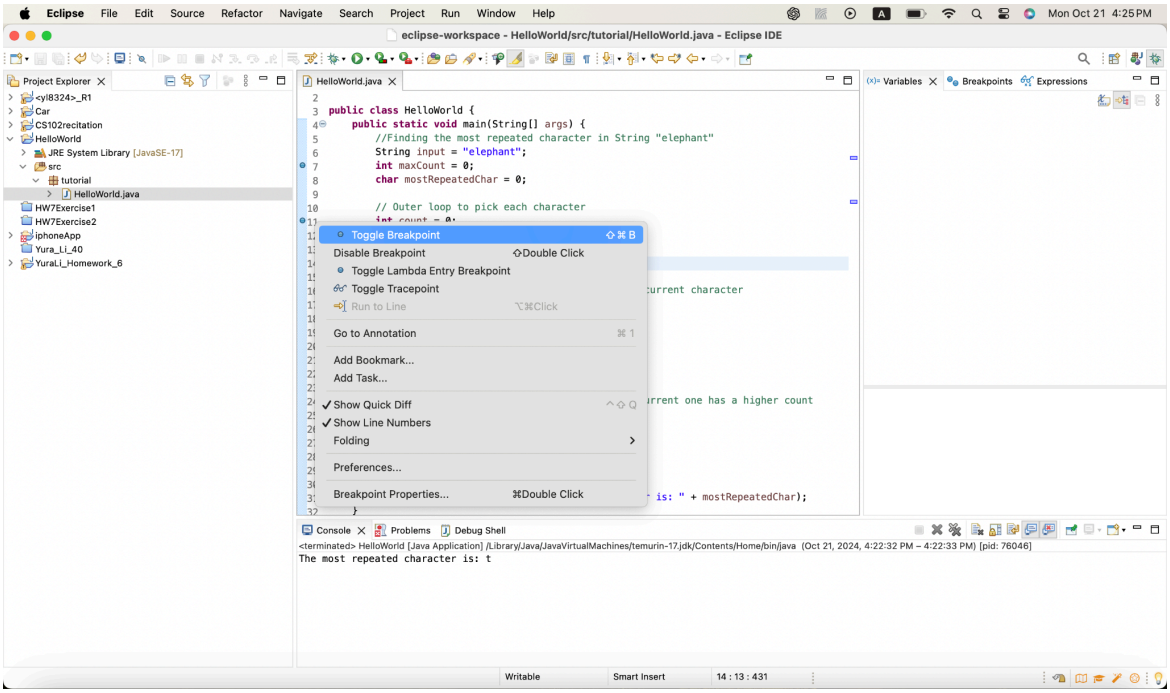
II. Mac

- Eclipse Doesn't Recognize Installed JDK:
  - Go to Eclipse → Preferences → Java → Installed JREs and manually add the JDK location. https://stackoverflow.com/questions/13635563/setting-jdk-in-eclipse
- Permission Issues:
  - Ensure Eclipse has proper permissions to access the workspace directory. You may need to adjust folder permissions. https://www.eclipse.org/forums/index.php/t/1085642/

8. Debugging in Eclipse

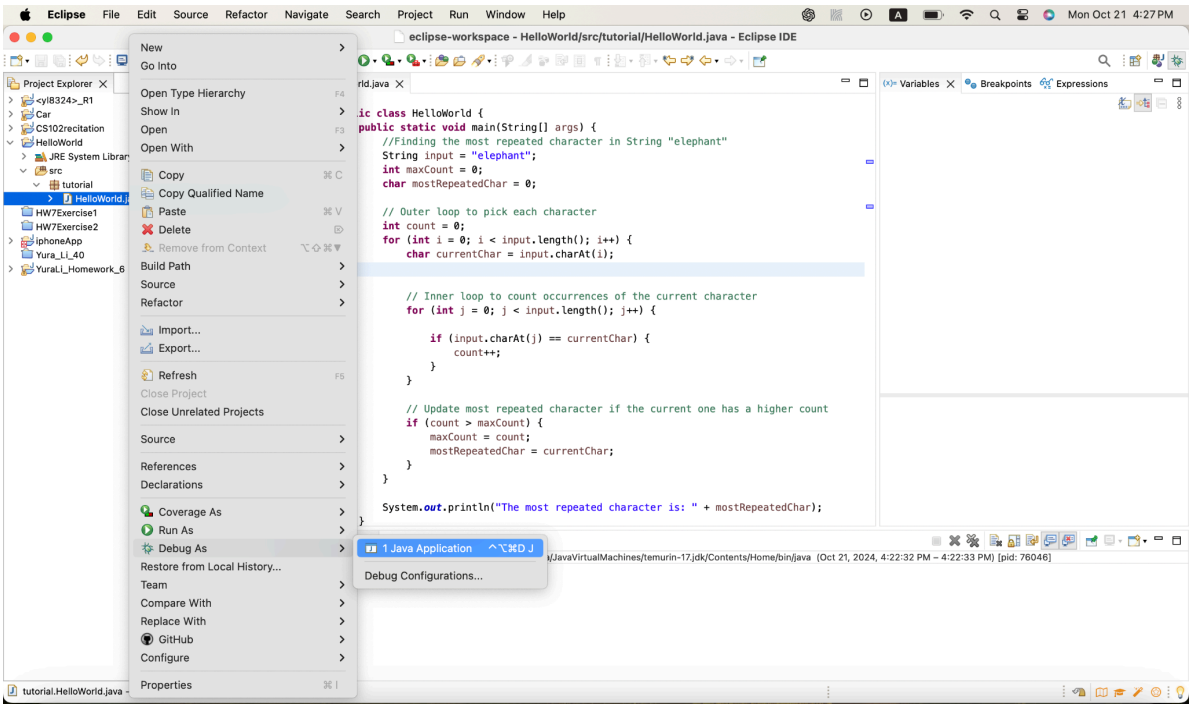
1. Setting Breakpoints

Click in the left margin next to a line number to set a breakpoint (e.g., In the example below, the most repeated character in “elephant” should be “e” but our output is “t”).



2. Debug Mode

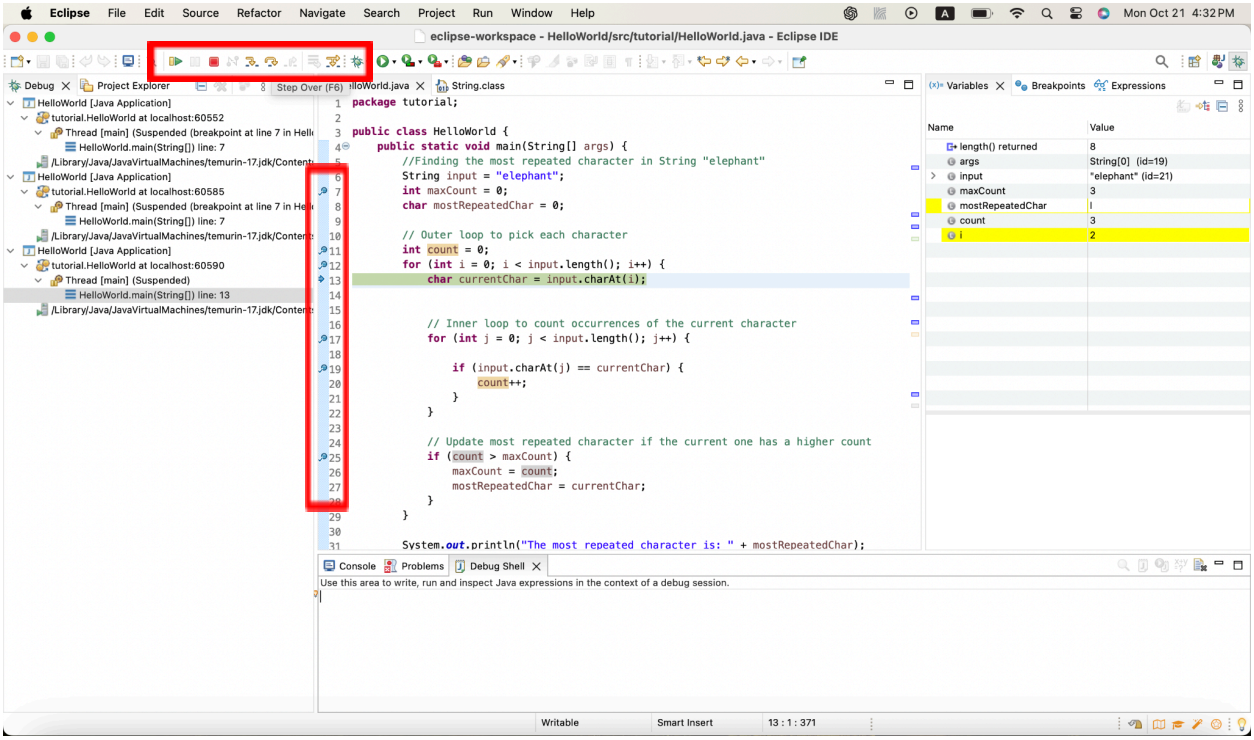
Click on the bug icon or right-click on the class → Debug As → Java Application.
Use the controls to step through the code and inspect variables.



3. **Viewing Variables and Call Stack**

In the Debug Perspective, watch how variable values change as you step through the program.

The Call Stack will help you trace how the code is being executed.



By adding several breakpoints, we are able to trace all the values of variables through each iteration. We found out that for the outer loop, our “count” variable didn’t reset resulting in continuous incrementing which leads to the wrong output.