

# Lab1 Setup

# Eclipse Tutorial

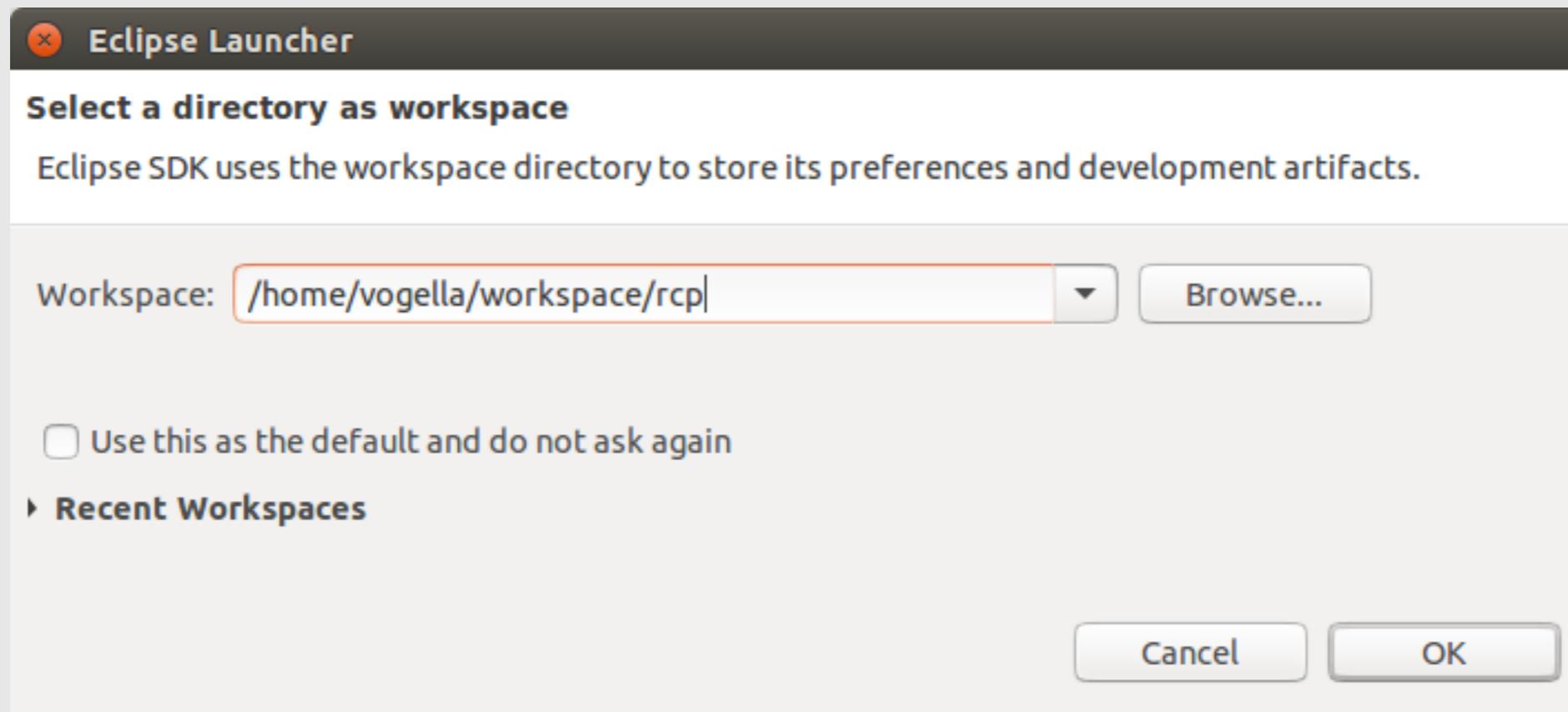
- Eclipse is a Java IDE.
- It's open source.
- Has many platforms and technologies.
- For this class, we will use Eclipse IDE for Java developers:
- Download link: <https://www.eclipse.org/downloads/packages/release/2019-12/r/eclipse-ide-javascript-developers>

# Java Development Tools (JDT)

- Eclipse has programming libraries called Java Development Tools (JDT).
- State of the art Java development environment
- Built atop Eclipse Platform
  - Implemented as Eclipse plug-ins
  - Using Eclipse Platform APIs and extension points
- Included in Eclipse Project releases
  - Available as separately installable feature
  - Part of Eclipse SDK drops

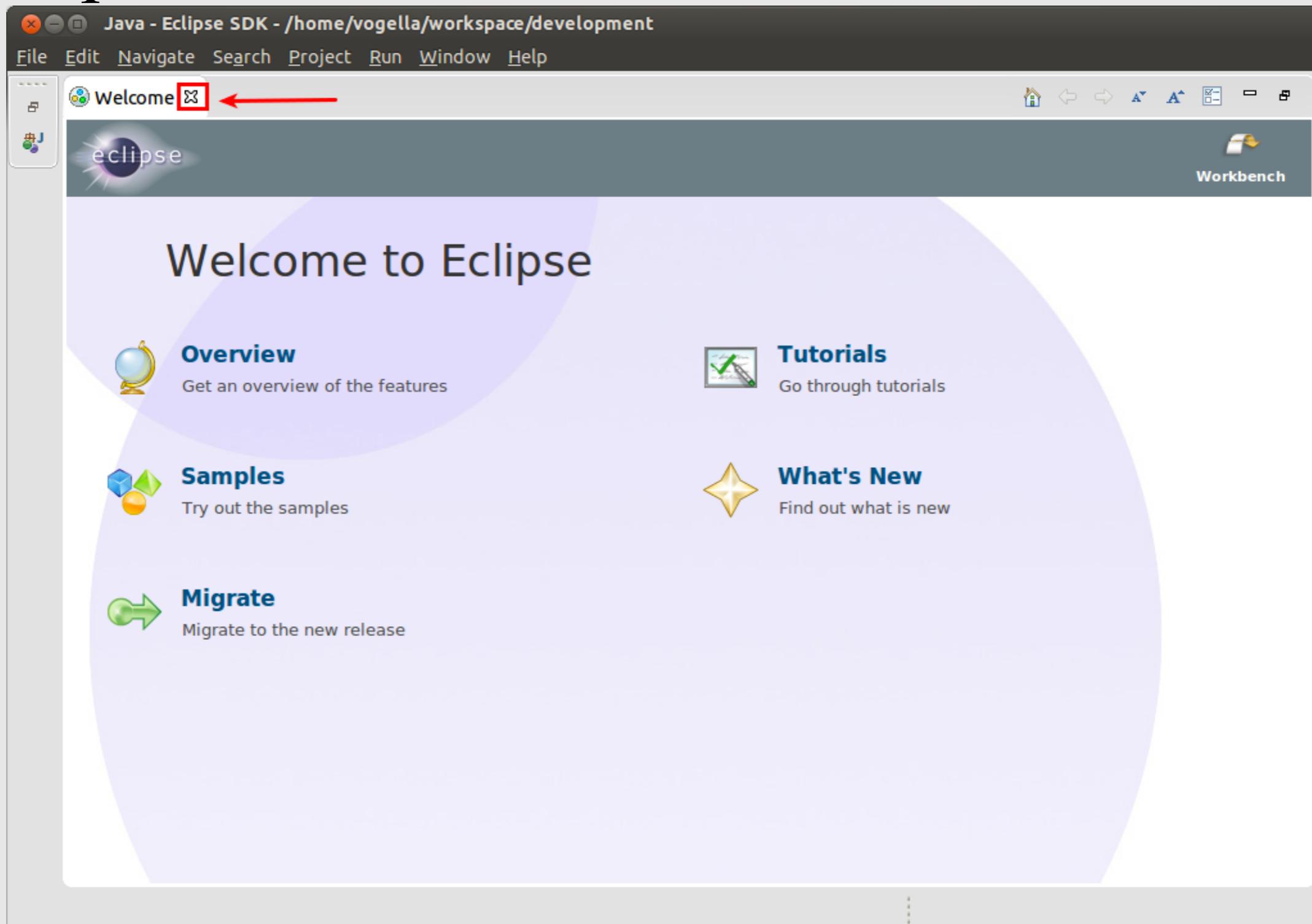
Source: [www.eclipse.org](http://www.eclipse.org)

# Eclipse Tutorial



When you open Eclipse, it will show a dialog to select your workspace directory. Workspace is the place where your codes will be saved. All Java projects will be saved in this workspace. Select a directory for your workspace.

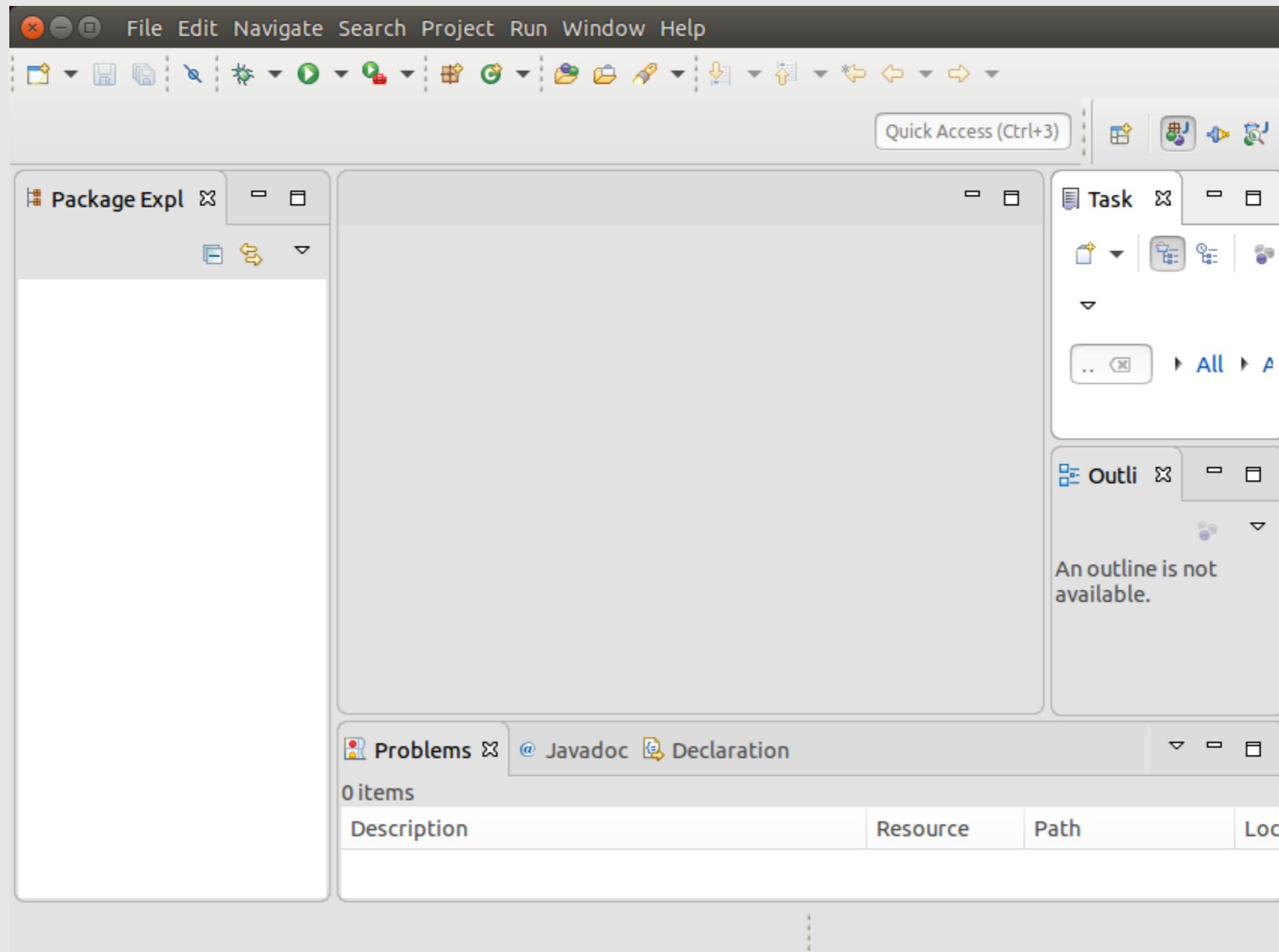
# Eclipse Tutorial



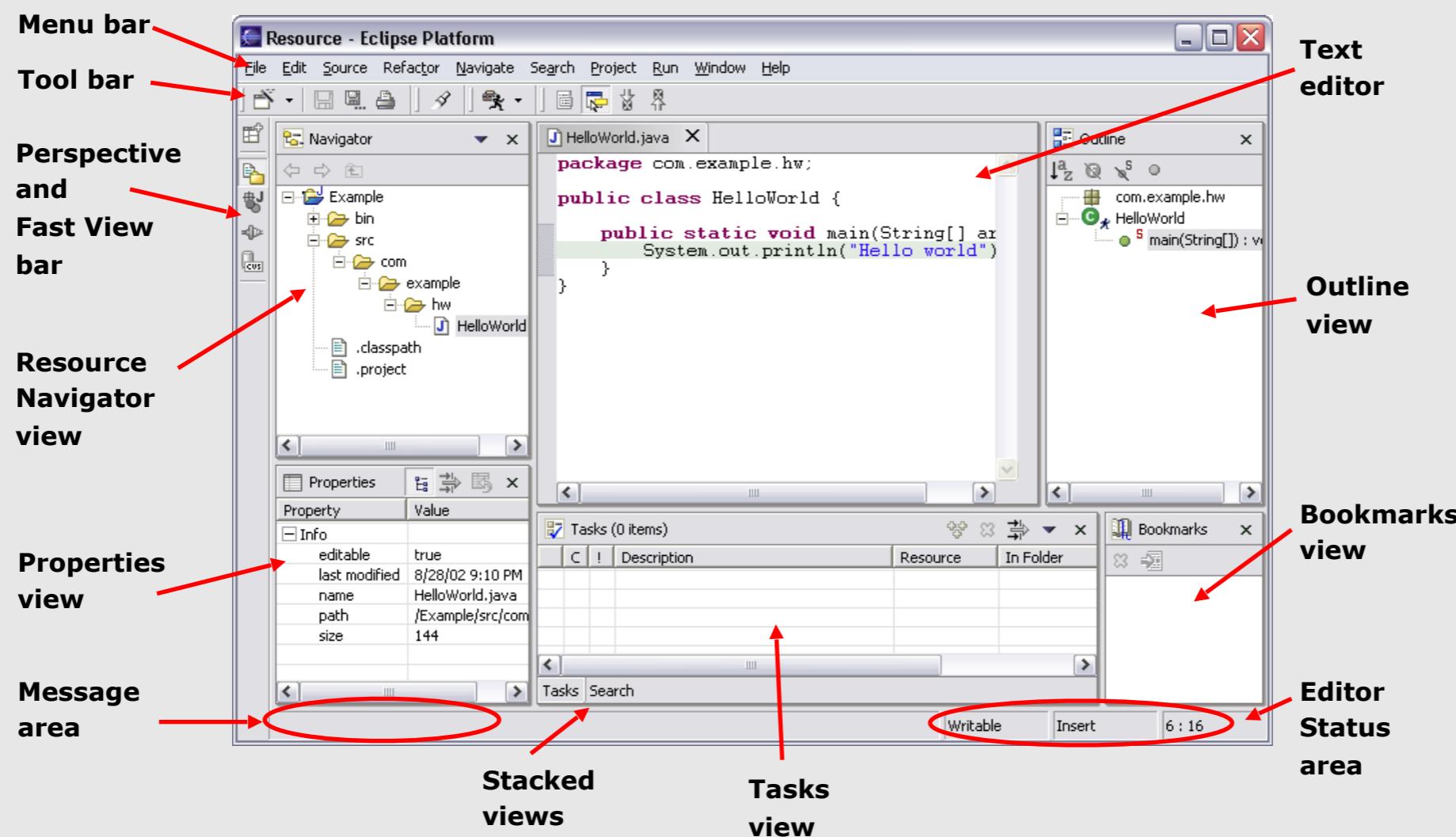
Close this window.

# Eclipse Tutorial

You will something like this..



# Workbench Terminology

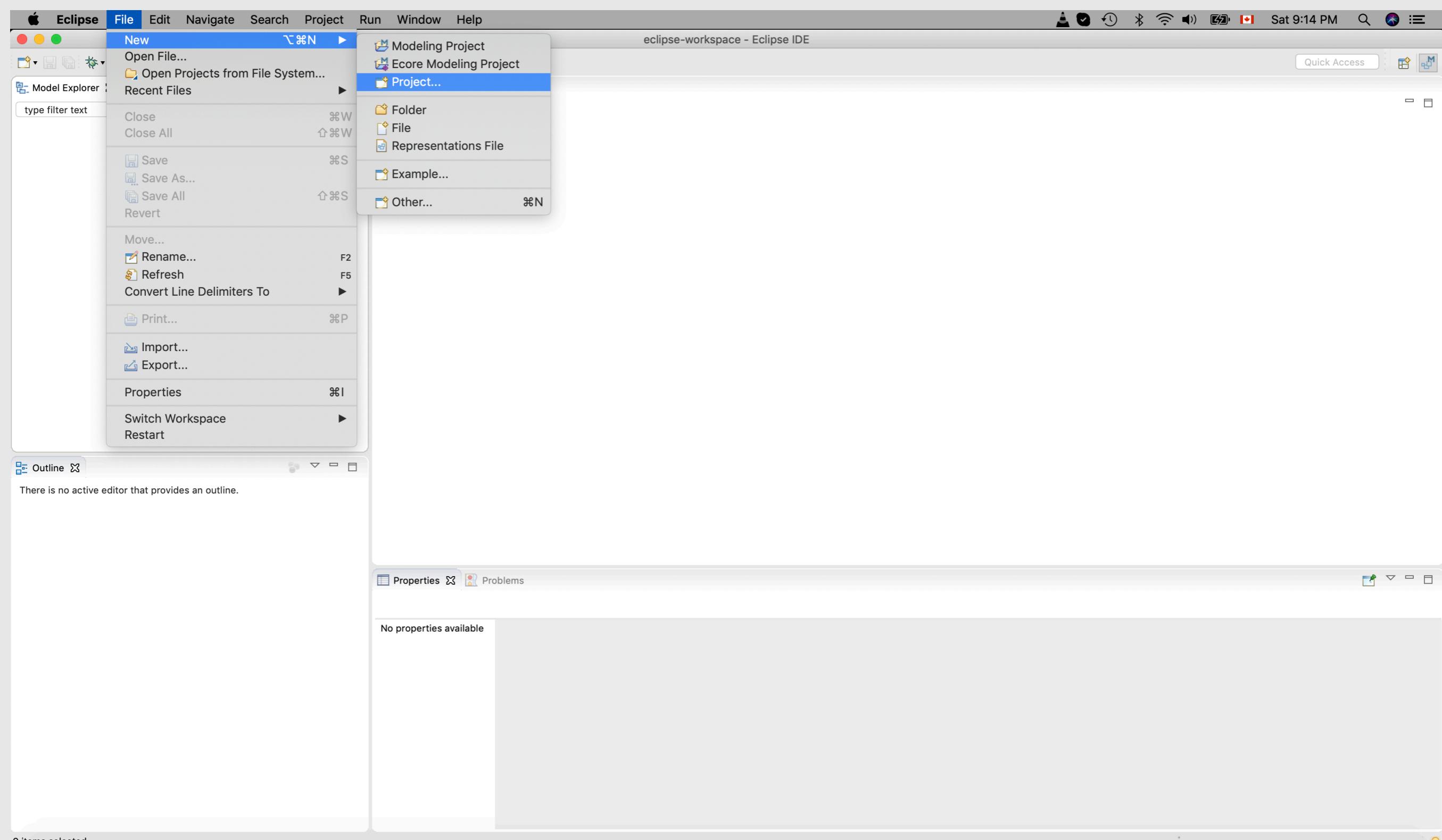


Source: [www.eclipse.org](http://www.eclipse.org)

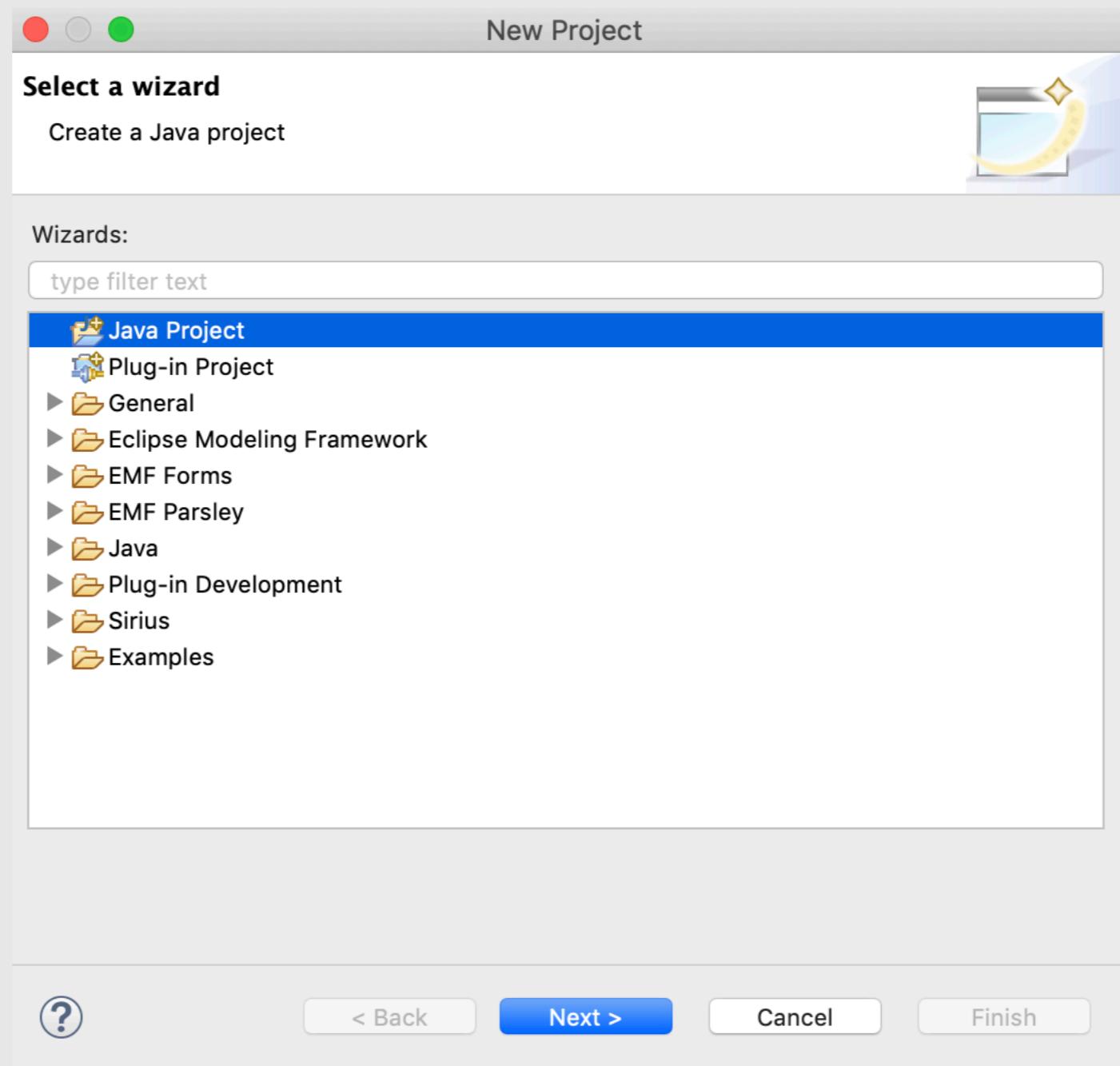
# Eclipse Projects

- Eclipse has concept of project.
- You have to create a project for each of your java program.
- There are different types of projects. For this class, we will only focus on Java projects.

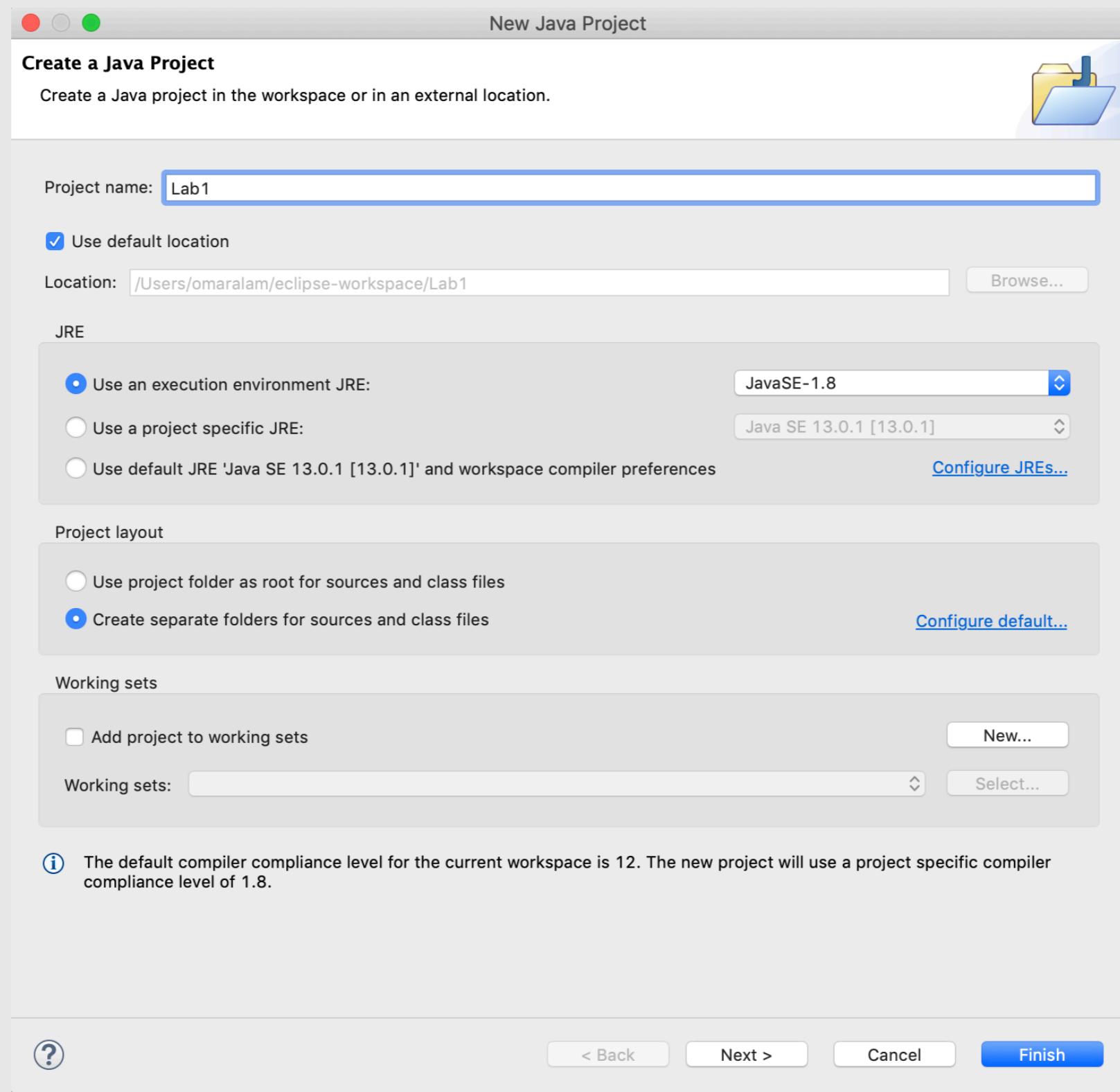
# Open Eclipse and click on File>New>Project



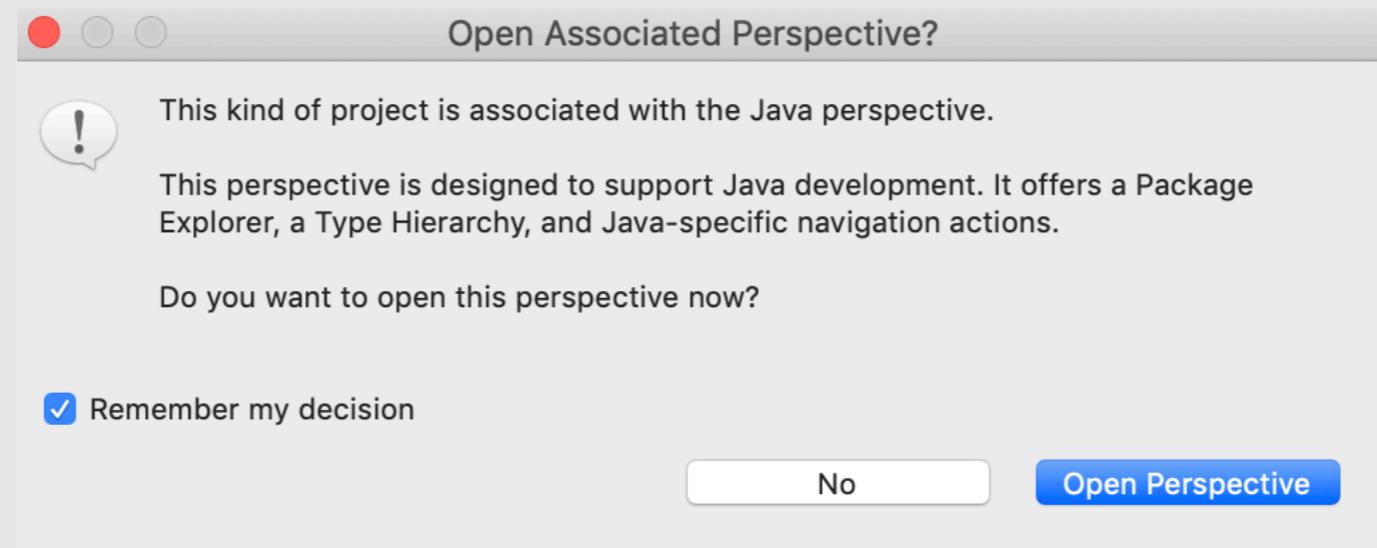
# From the wizard select Java Project



# Type your project name and click Finish



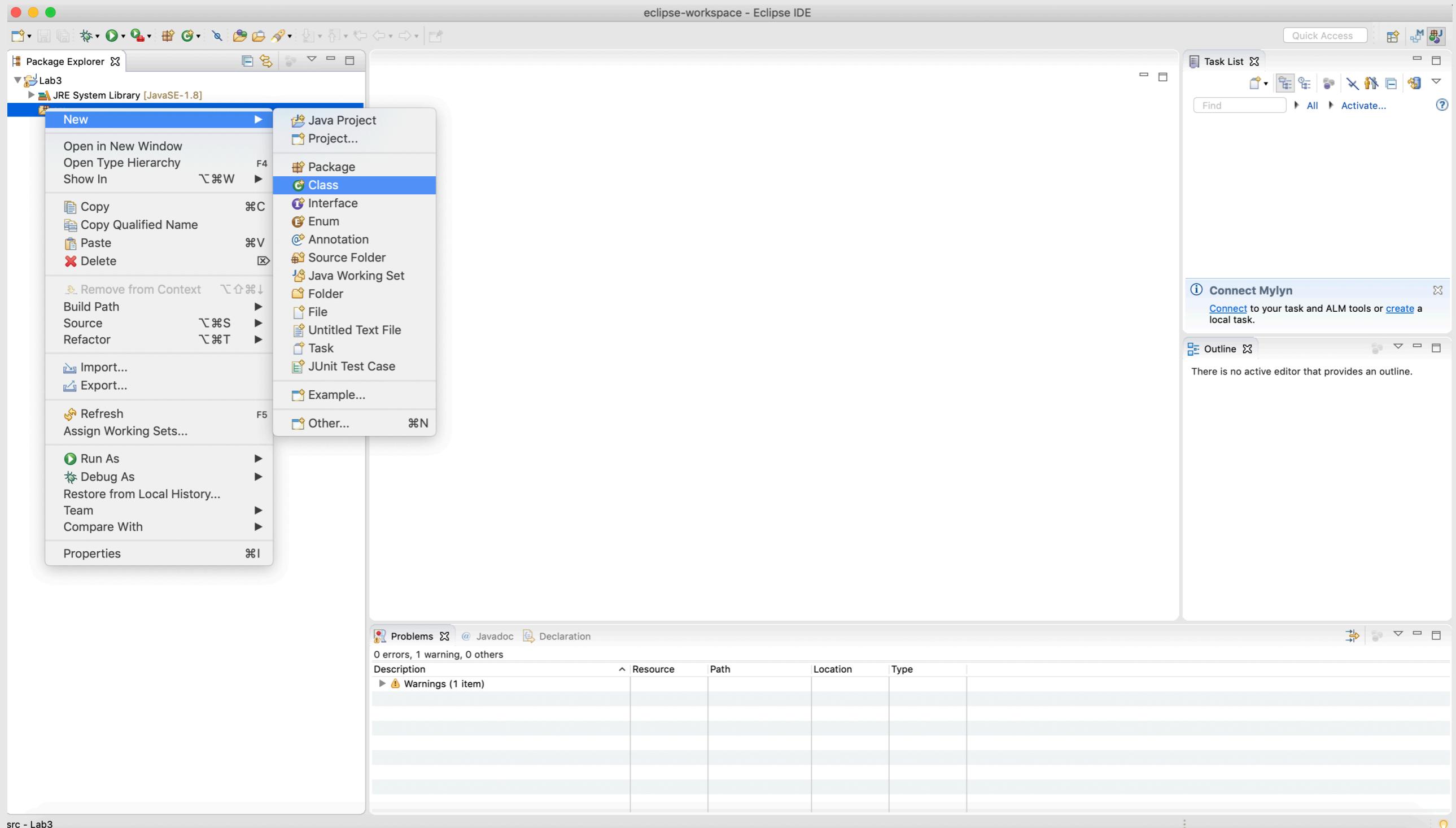
# Click on Open Perspective



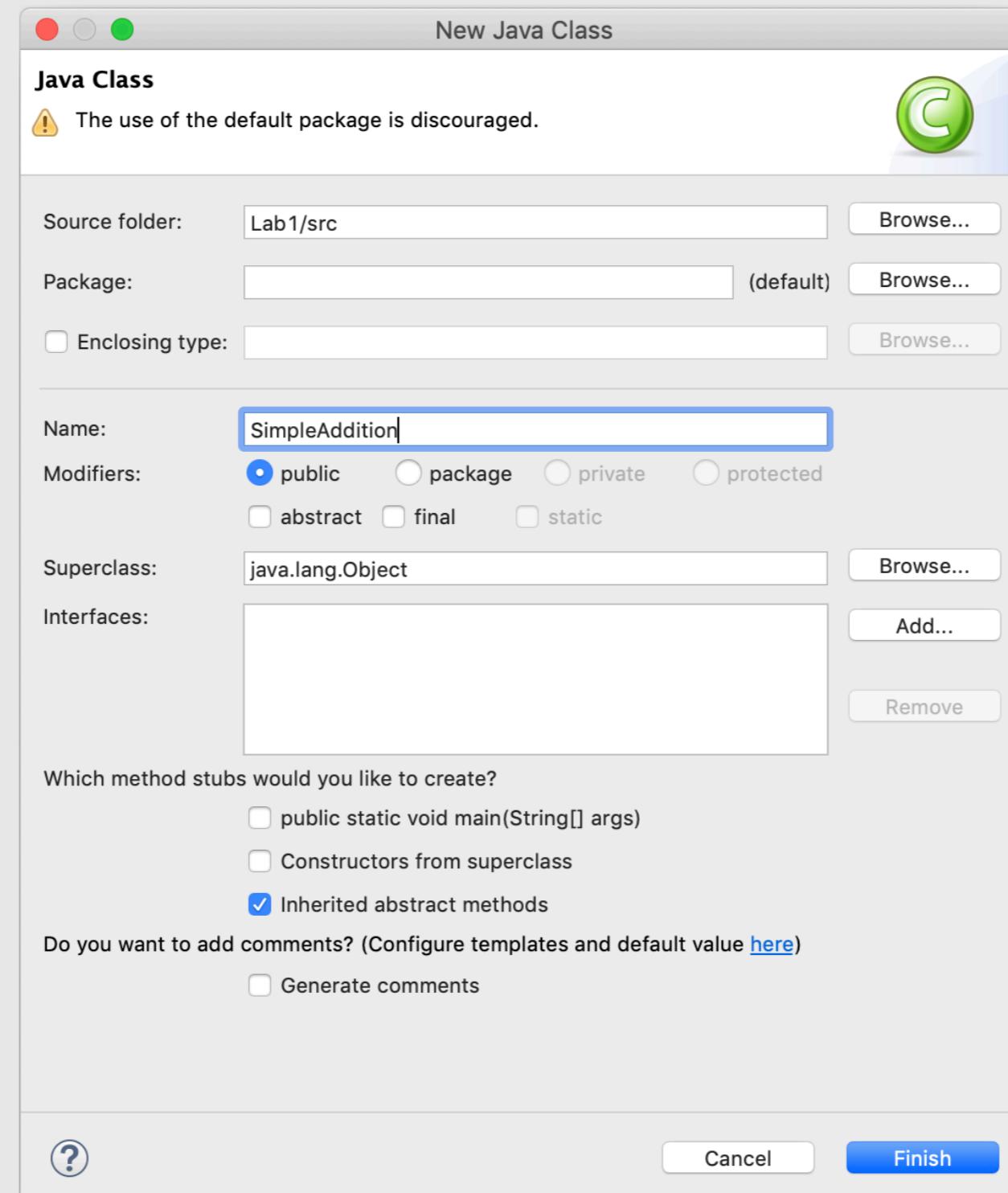
# Eclipse Projects

- Each Java class has to belong to a package.
- When you right-click on source (as shown in the next slide), you can add a Java class.
- In this case, Eclipse will add a default package for you and add the class to that project. You can change the name of the package later on.
- Alternatively, you can start by adding a package and then a class.

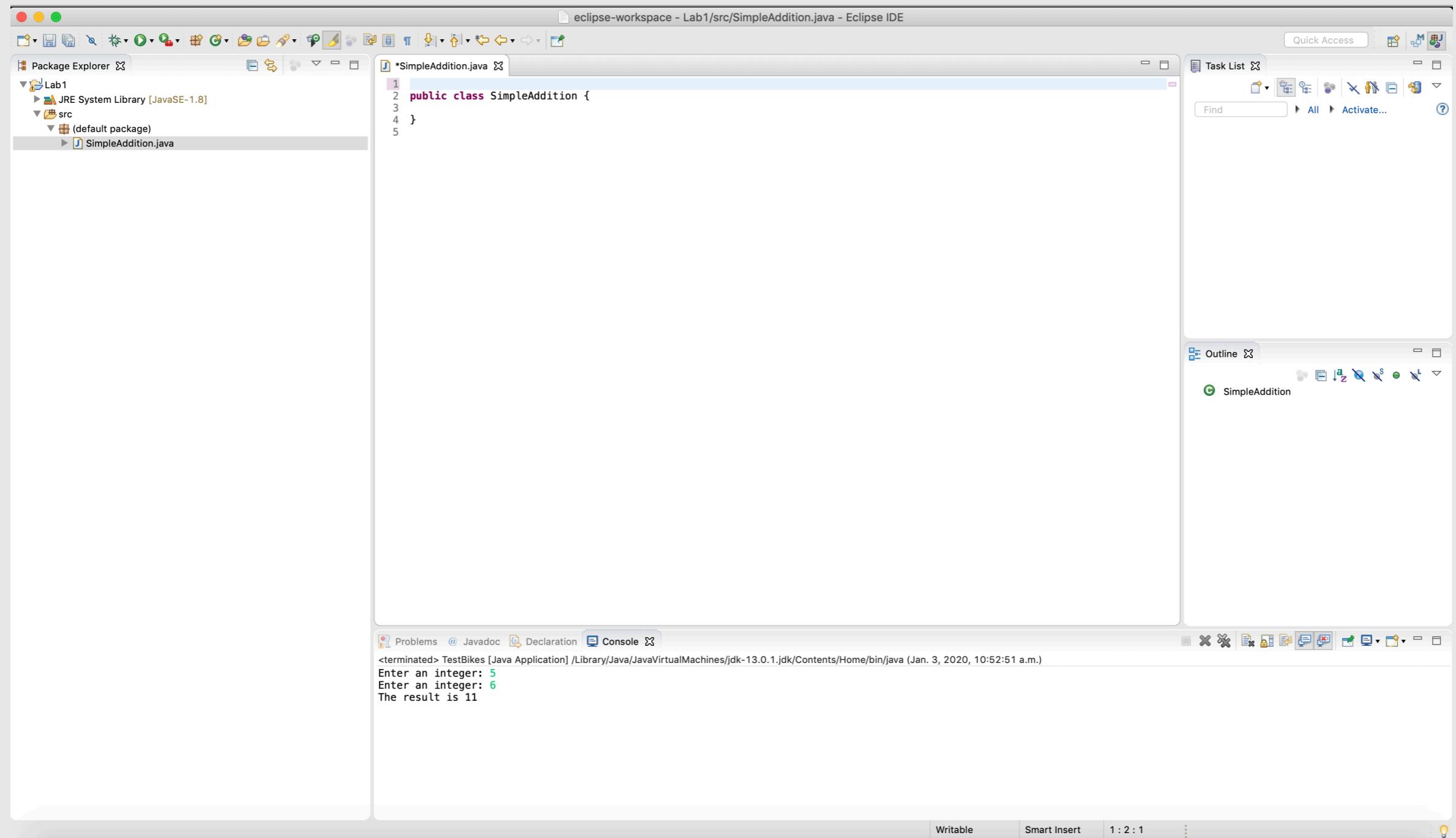
# Right-click on src, then select New>Class



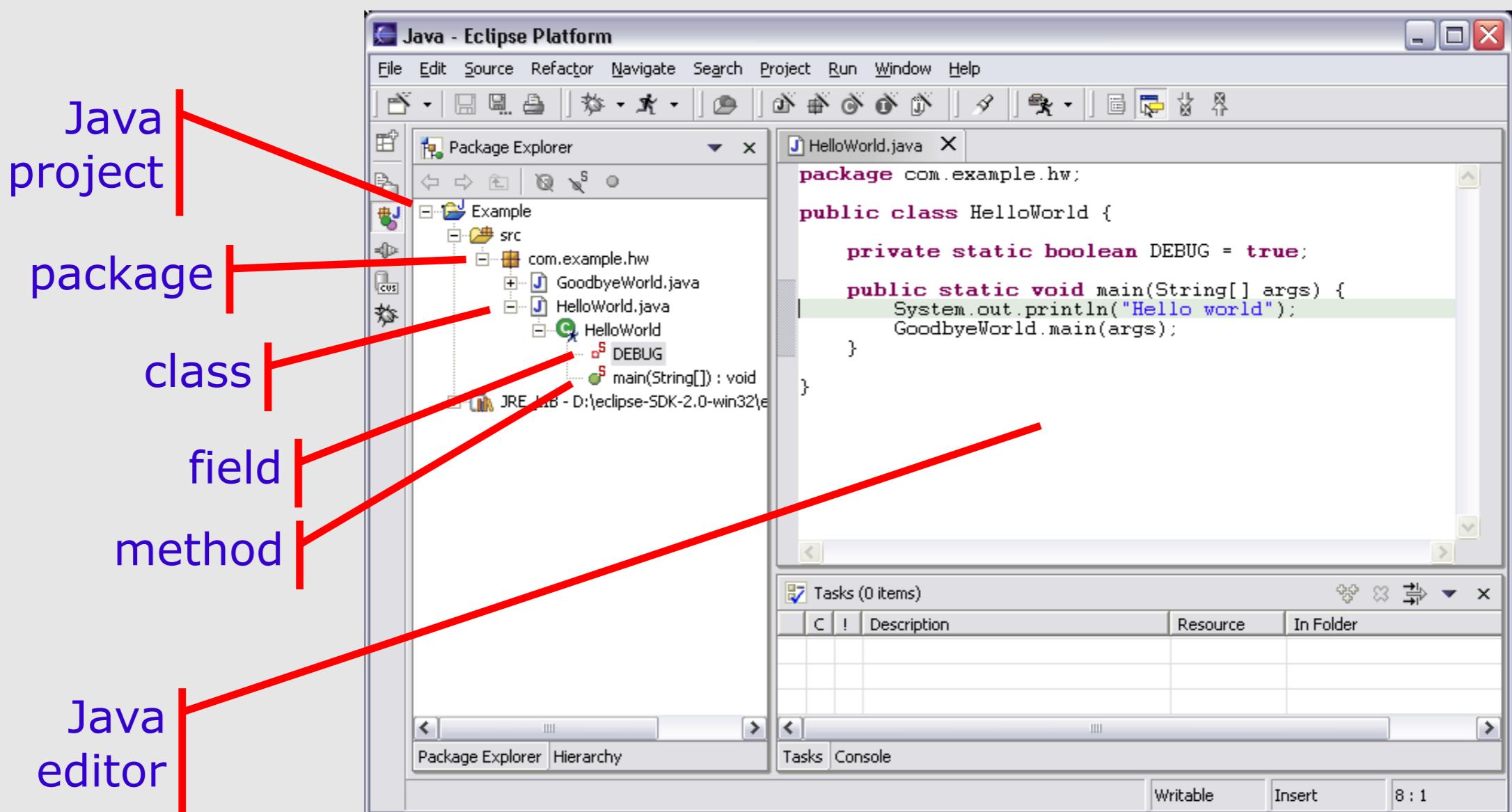
# Give your class a name and then click Finish



You can see your class, now you can write your code.. Write your first Hello world close in Java.



# Java Perspective



Write a simple code that takes two numbers and outputs the addition of them in Java. Ask for help if you need. You need to write a main function. The first input is shown to you here.

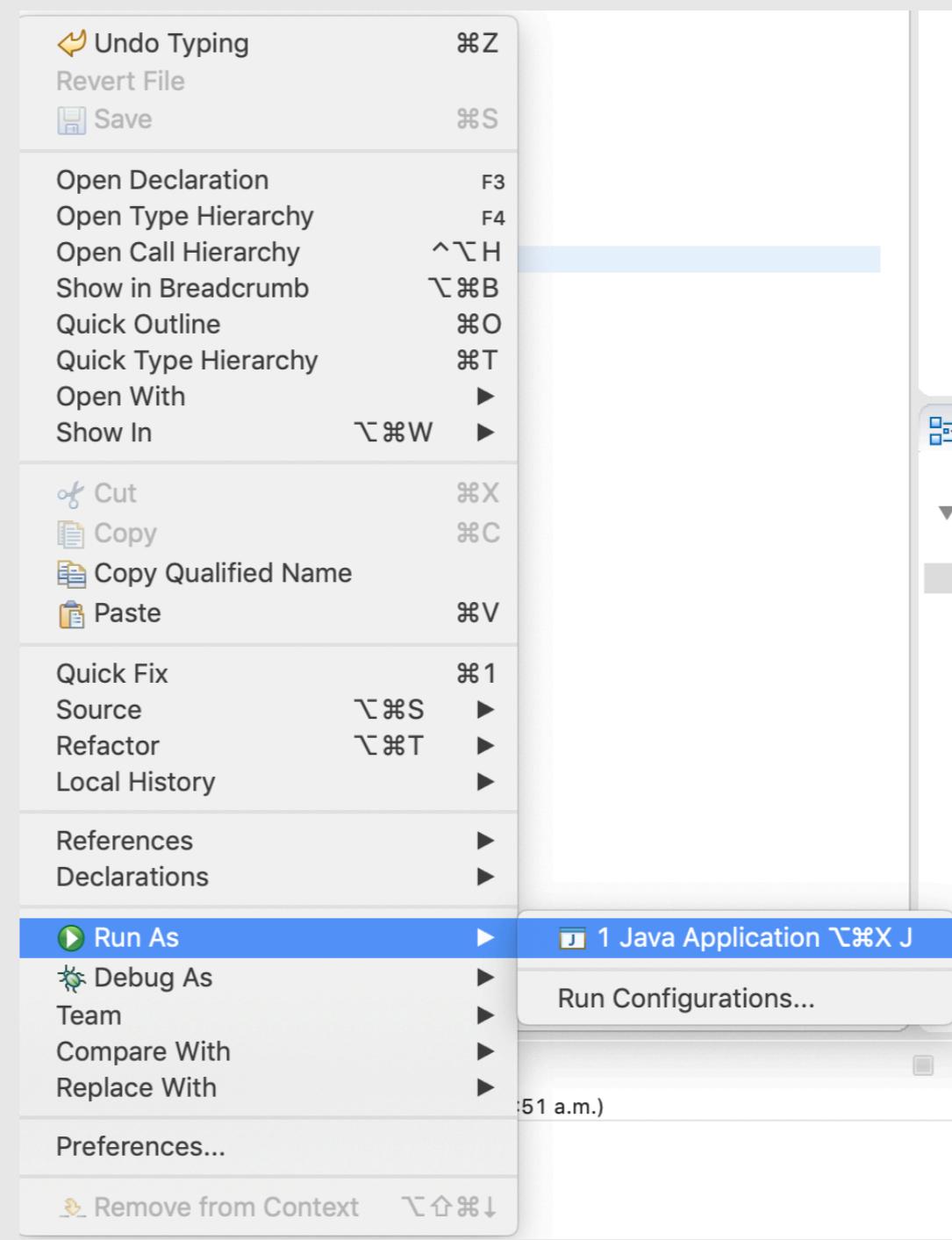
```
import java.util.Scanner;
class SimpleAddition {
    private static Scanner input;

    public static void main(String[] args){

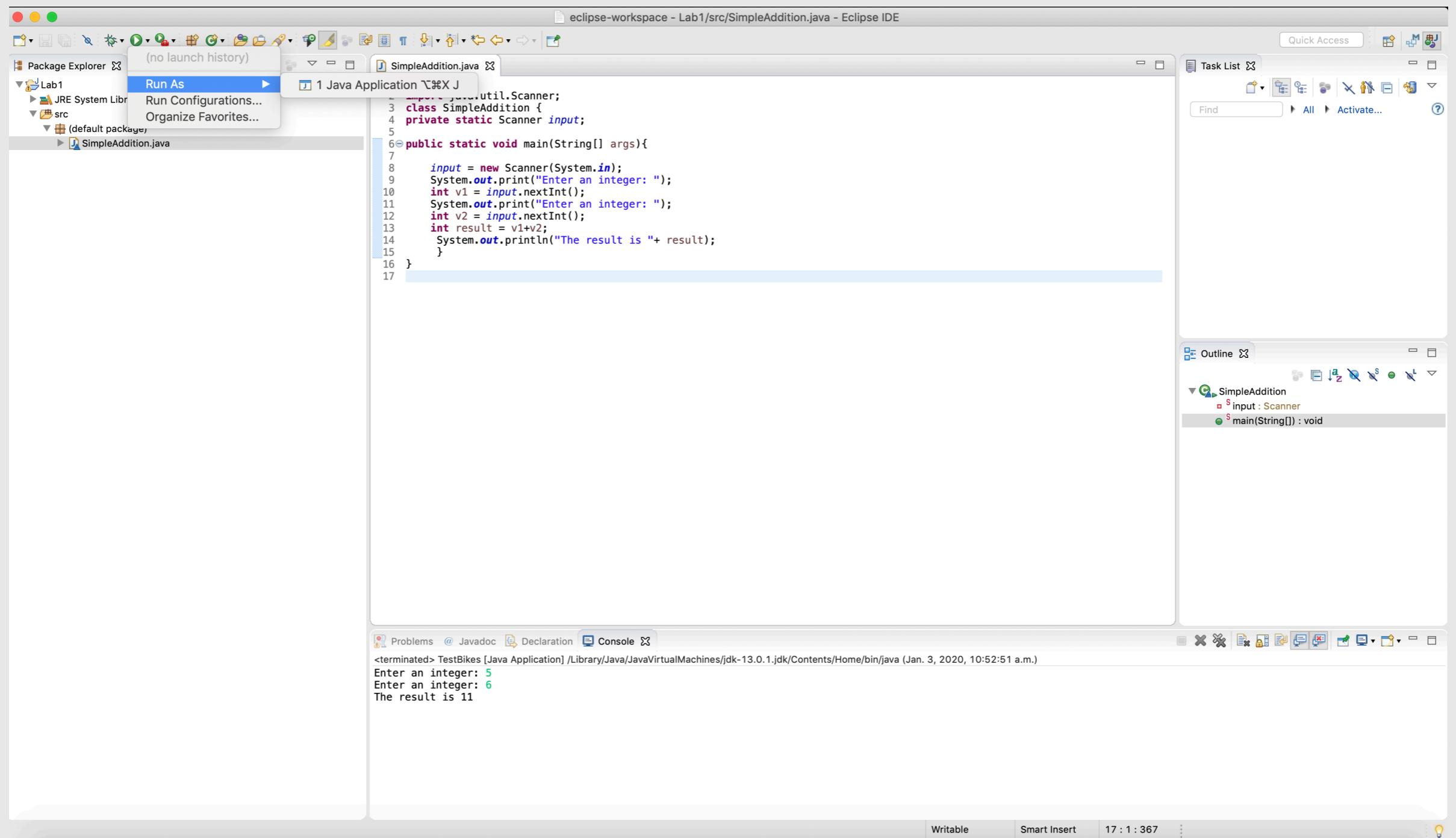
        input = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int v1 = input.nextInt();
        System.out.print("Enter an integer: ");
        int v2 = input.nextInt();
        v1++;
        int result = v1+v2;
        System.out.println("The result is "+ result);
    }
}
```

Then run your code..

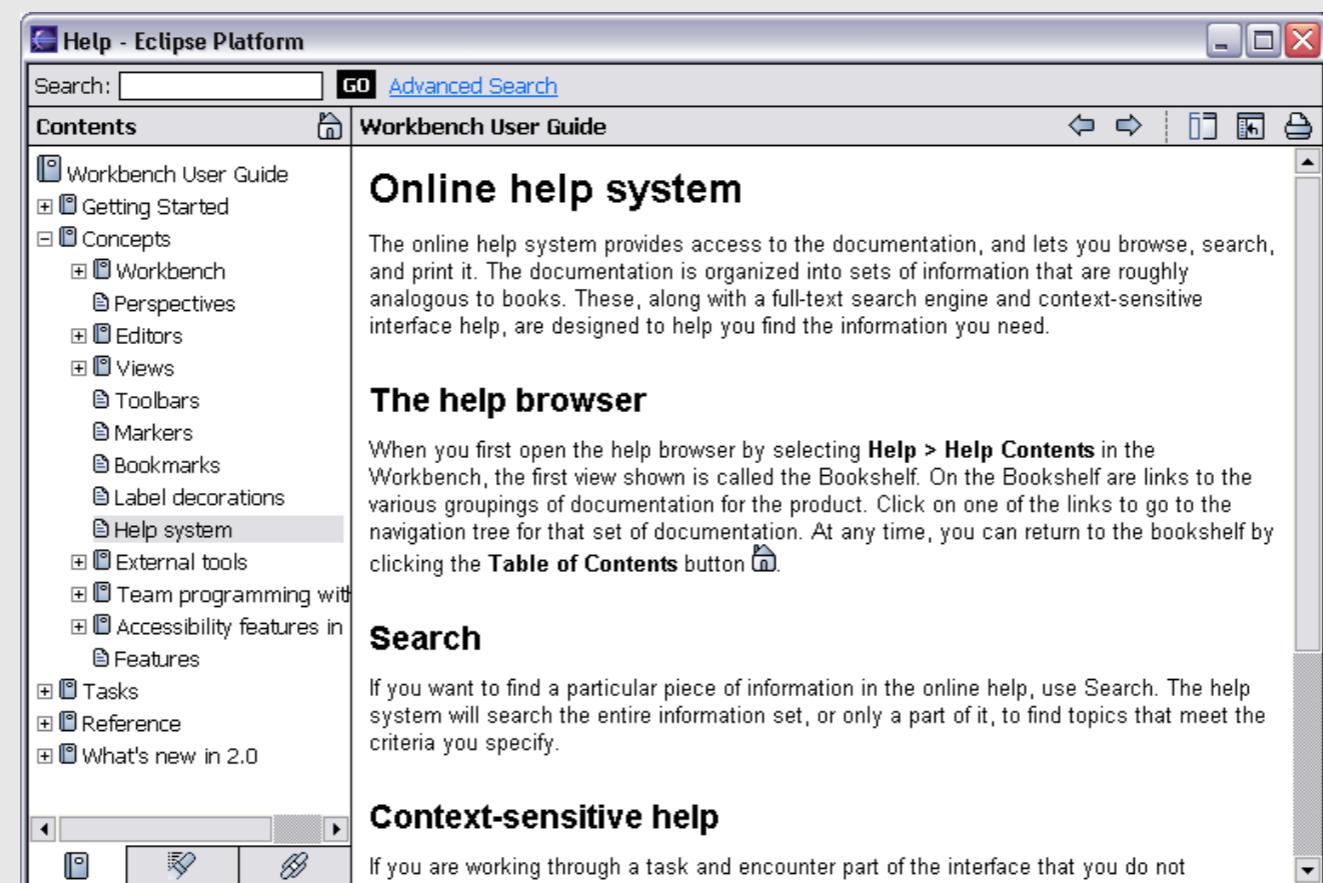
To run your code, right-click on your code and then select Run As->Java Application.



You can do that by clicking on the green button on file menu->Run As->Java Application

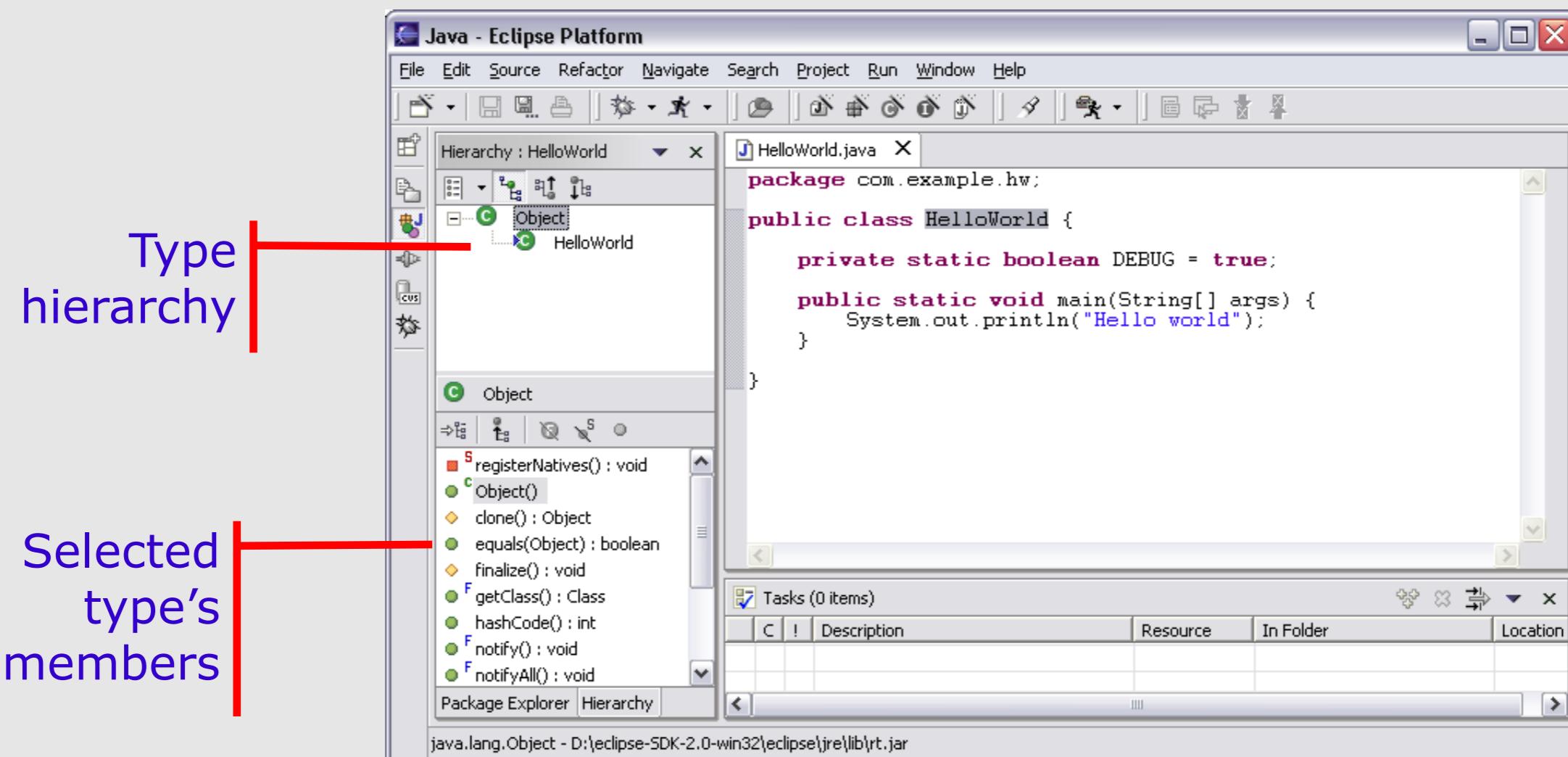


# Eclipse Help



Source: [www.eclipse.org](http://www.eclipse.org)

# Java Perspective



A screenshot of a Java code editor window titled "HelloWorld.java". The code is as follows:

```
package com.example.hw;

public class HelloWorld implements Cloneable {

    private static boolean DEBUG = true;

    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

The word "System" is highlighted in the code. A tooltip box is displayed, containing the following Javadoc text:

**java.lang.System**

The System class contains several useful class fields and methods. It cannot be instantiated. Among the facilities provided by the System class are standard input, standard output, and error output streams; access to externally defined "properties"; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.

At the bottom of the editor window, there are buttons for "Writable", "Insert", and a status bar showing "8 : 10".

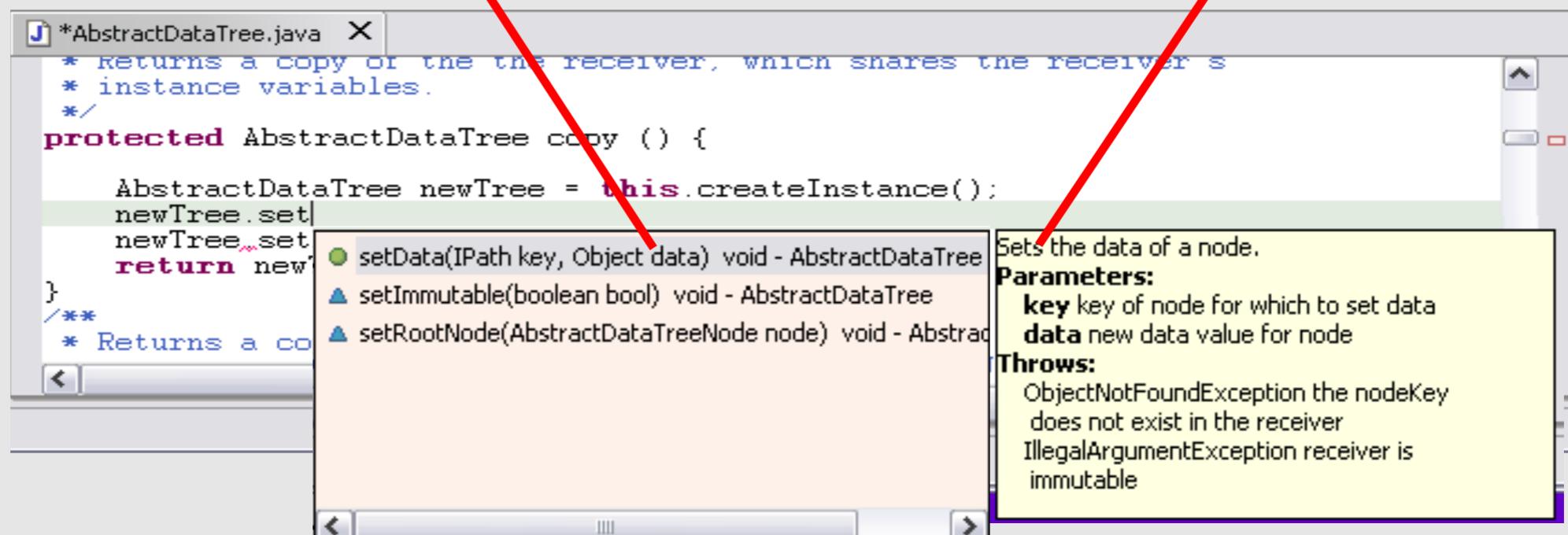
Hovering over identifier shows Javadoc spec

# Java Editor

- Method completion in Java editor

List of plausible methods

Doc for method



# Eclipse Debugger

- Eclipse Debugger allows you to inspect your code and see the values of the variables step-by-step.
- Usually, when developers suspect that their code is wrong, or they want to test that their code is correct, they use the debugger.
- To debug your code, you have to use breakpoints at the lines of code that you want to inspect

# Eclipse Debugger

A screenshot of the Eclipse IDE interface. A context menu is open over a Java code editor. The code shown is:

```
1 import java.util.Scanner;
2 class SimpleAddition {
3     private static Scanner input;
4
5     public static void main(String[] args){
6         input = new Scanner(System.in);
7         System.out.print("Enter an integer: ");
8         int v1 = input.nextInt();
9         System.out.print("Enter an integer: ");
10        int v2 = input.nextInt();
11        result = v1 + v2;
12    }
13 }
```

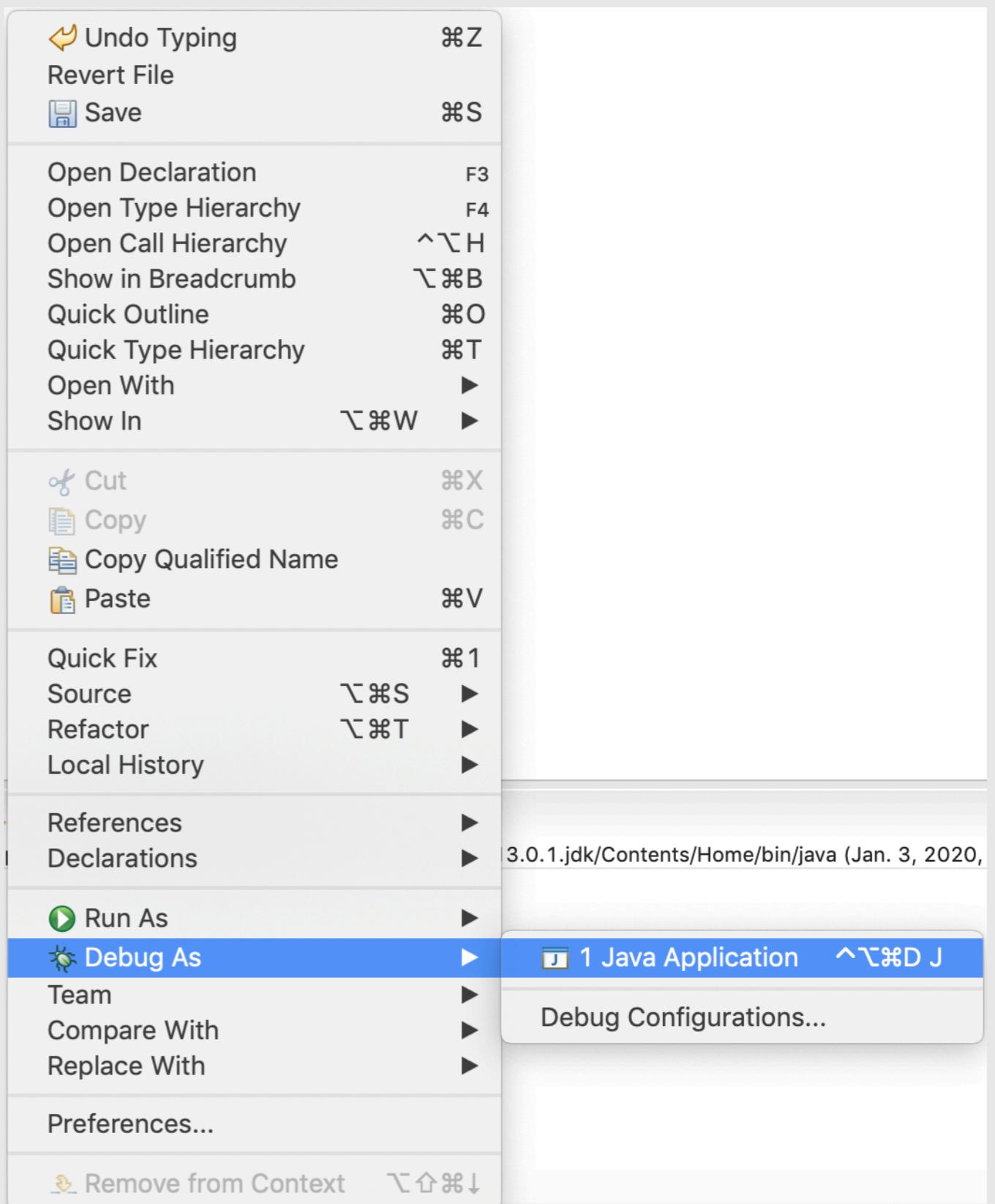
The line `int v1 = input.nextInt();` is highlighted with a light blue background. The context menu options visible are:

- Toggle Breakpoint (⌘B)
- Disable Breakpoint (Double Click)
- Run to Line (Click)
- Go to Annotation (⌘1)
- Add Bookmark...
- Add Task...
- Show Quick Diff (⇧⌘Q)
- Show Line Numbers
- Folding (▶)
- Preferences...
- Breakpoint Properties... (⌘Double Click)

- Toggle breakpoint by right-clicking on the line `v++;`

# Eclipse Debugger

Right click, then select  
Debug As->Java Application



# Eclipse Debugger

A small blue-dot will show up

```
1 import java.util.Scanner;
2 class SimpleAddition {
3     private static Scanner input;
4
5     public static void main(String[] args){
6         input = new Scanner(System.in);
7         System.out.print("Enter an integer: ");
8         int v1 = input.nextInt();
9         System.out.print("Enter an integer: ");
10        int v2 = input.nextInt();
11        v1++;
12        int result = v1+v2;
13        System.out.println("The result is "+ result);
14    }
15 }
```

# Eclipse Debugger

When the debugger is running, Eclipse will expect that you are going to Navigate the code line-by-line. There are two kinds of “stepping”:  
1-Step into: which will go to the details of each line, starting with the toggled line.

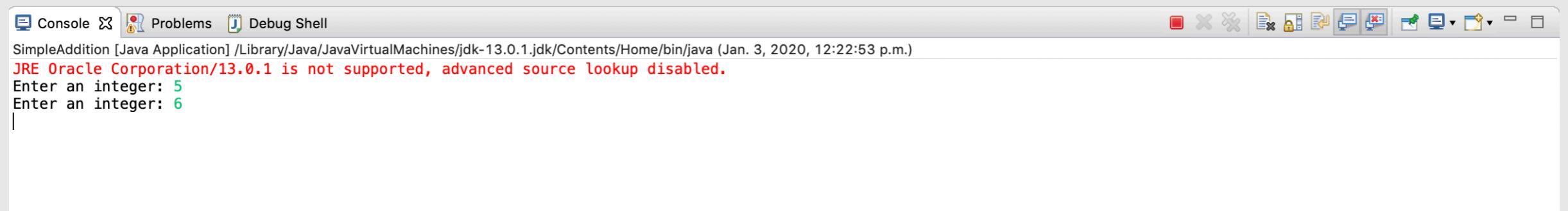
2-Step over: which will not go to the details of the line.

The first arrow from the right is step over, the next one is step into



# Eclipse Debugger

After you enter the two numbers in the console, the debugger will be triggered by the toggled line.



The screenshot shows the Eclipse IDE's Console view. The title bar includes tabs for 'Console', 'Problems', and 'Debug Shell'. The main area displays the output of a Java application named 'SimpleAddition' running on Java 13.0.1. The log message indicates the application was run on Jan. 3, 2020, at 12:22:53 p.m. A red warning message states: 'JRE Oracle Corporation/13.0.1 is not supported, advanced source lookup disabled.' Below this, the application prompts the user to enter integers. The first input '5' is in blue, and the second input '6' is in green, likely indicating they were entered while the debugger was active.

```
SimpleAddition [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java (Jan. 3, 2020, 12:22:53 p.m.)
JRE Oracle Corporation/13.0.1 is not supported, advanced source lookup disabled.
Enter an integer: 5
Enter an integer: 6
```

# Eclipse Debugger

You can follow the changes to the values of variables to your right:

The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - Lab1/src/SimpleAddition.java - Eclipse IDE". The left side features the Project Explorer and Debug perspectives. The center contains the code editor with SimpleAddition.java open, showing a breakpoint at line 13. The right side displays the Variables view, which lists the current values of variables: args (String[0] (id=18)), v1 (5), and v2 (6). The code in SimpleAddition.java is as follows:

```
1 import java.util.Scanner;
2 class SimpleAddition {
3     private static Scanner input;
4     public static void main(String[] args){
5         input = new Scanner(System.in);
6         System.out.print("Enter an integer: ");
7         int v1 = input.nextInt();
8         System.out.print("Enter an integer: ");
9         int v2 = input.nextInt();
10        v1++;
11        int result = v1+v2;
12        System.out.println("The result is "+ result);
13    }
14 }
```