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-java-developers

Java Development Tools (JDT)

- Eclipse has programming libraries called Java Development Tools (JDT).

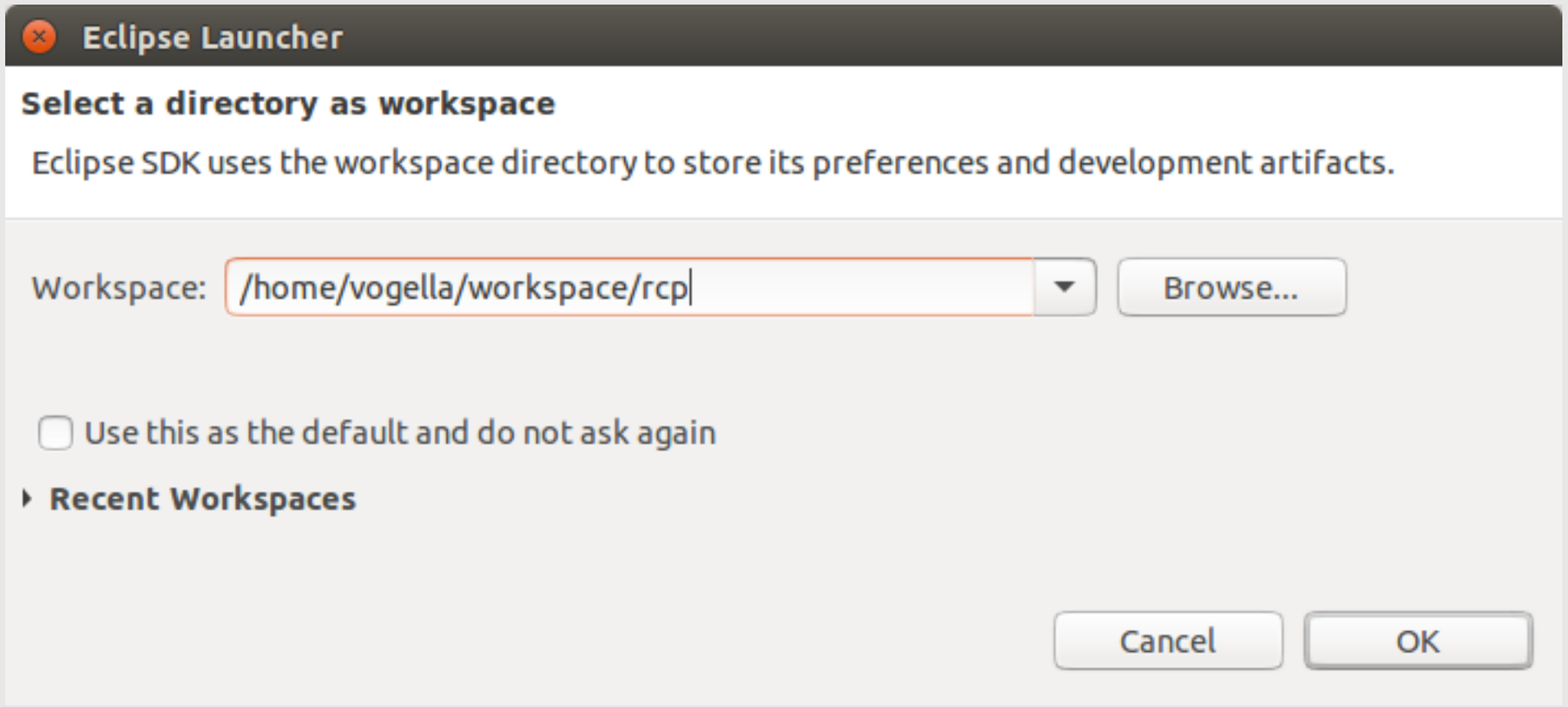
- State of the art Java development environment

- 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

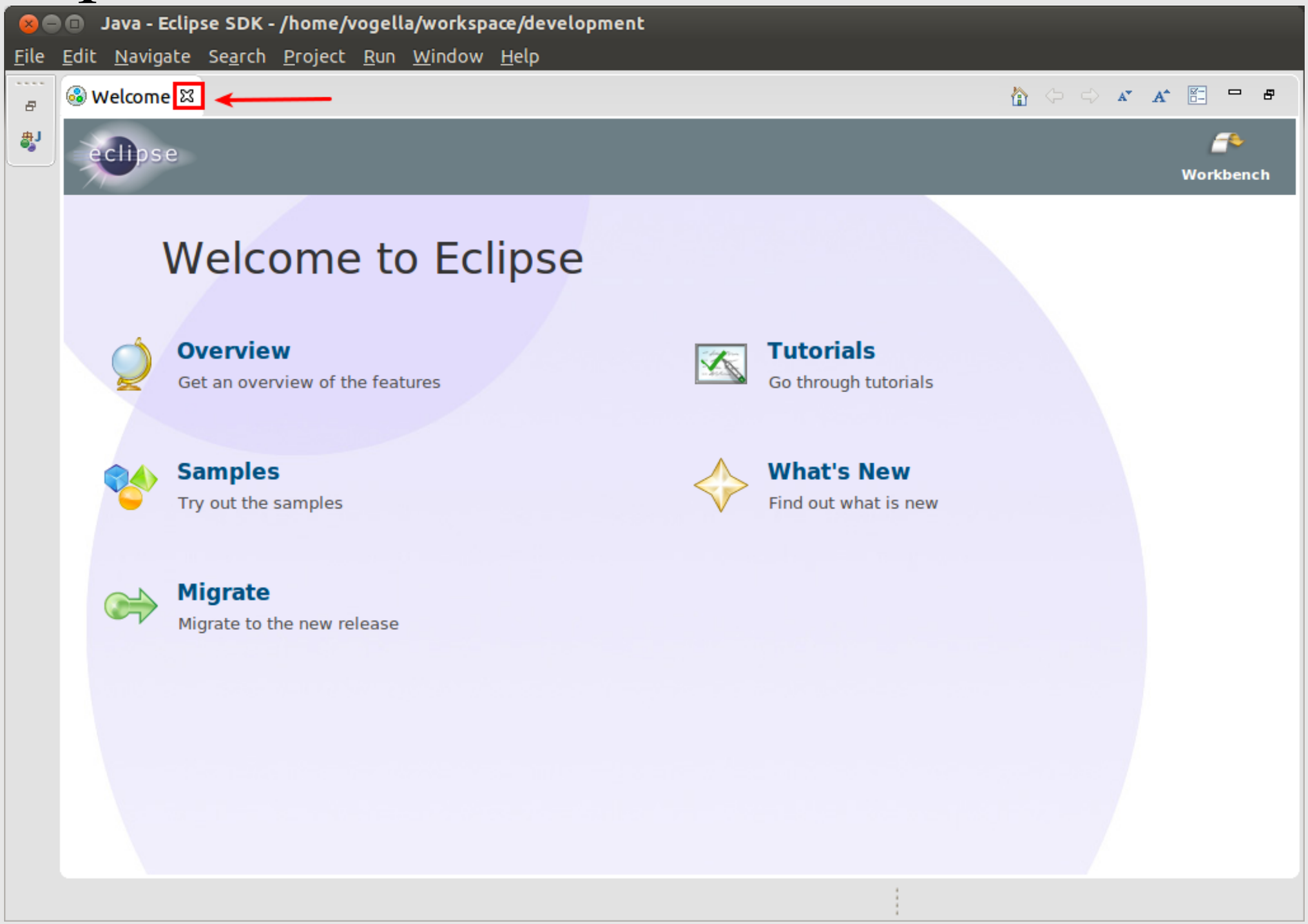
Source: www.eclipse.org

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.

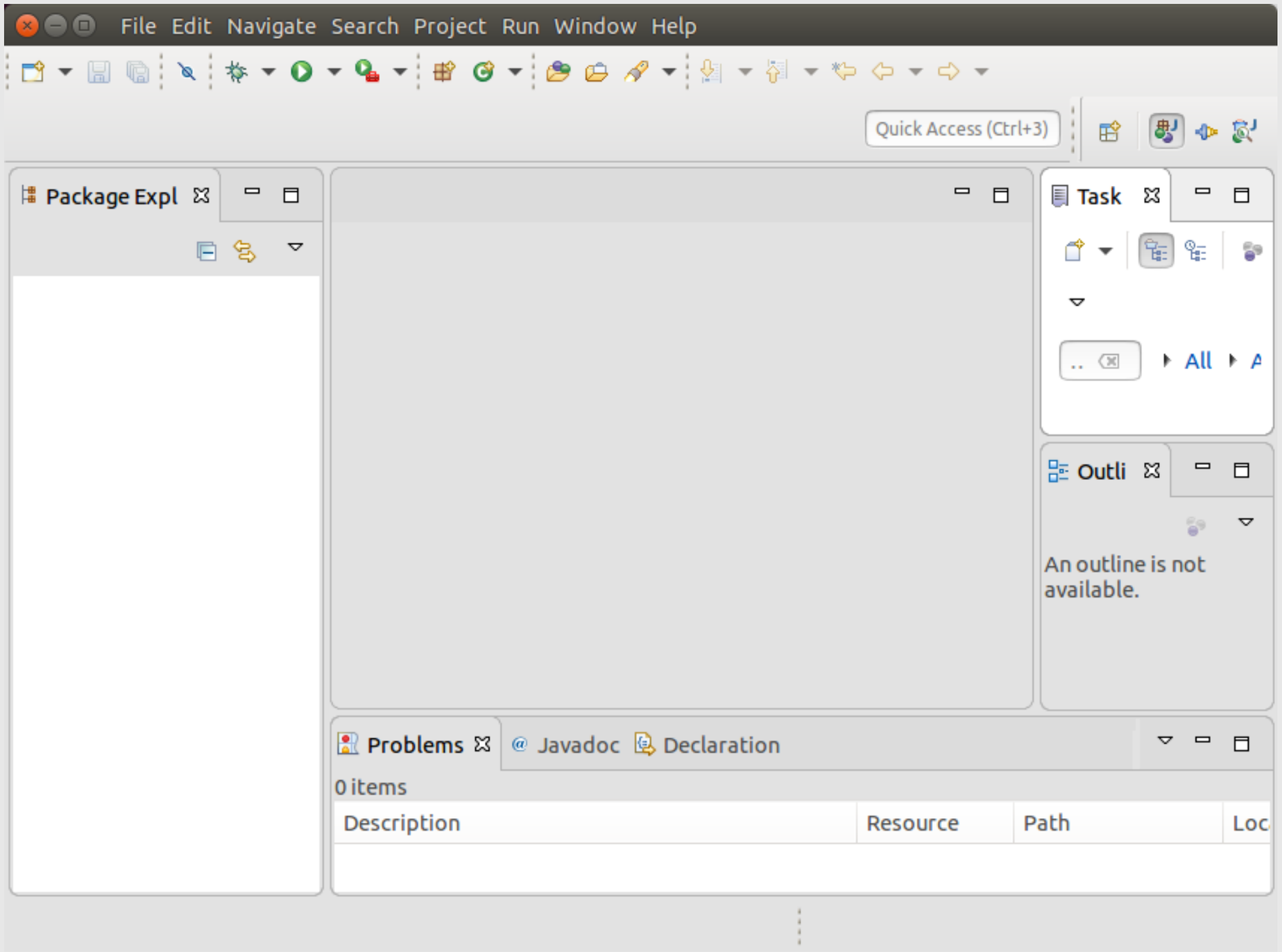
Eclipse Tutorial



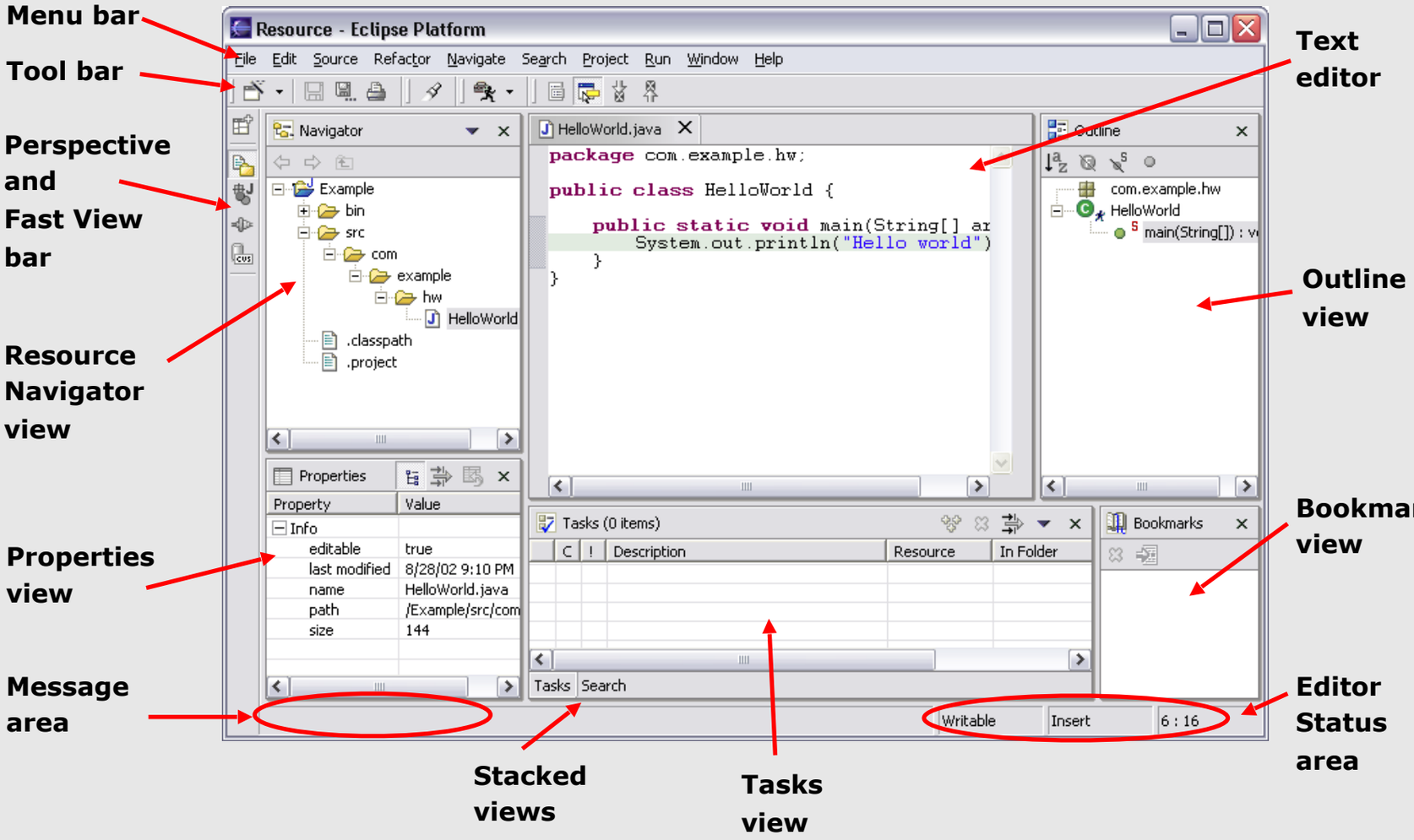
Close this window.

Eclipse Tutorial

You will something like this..



Workbench Terminology

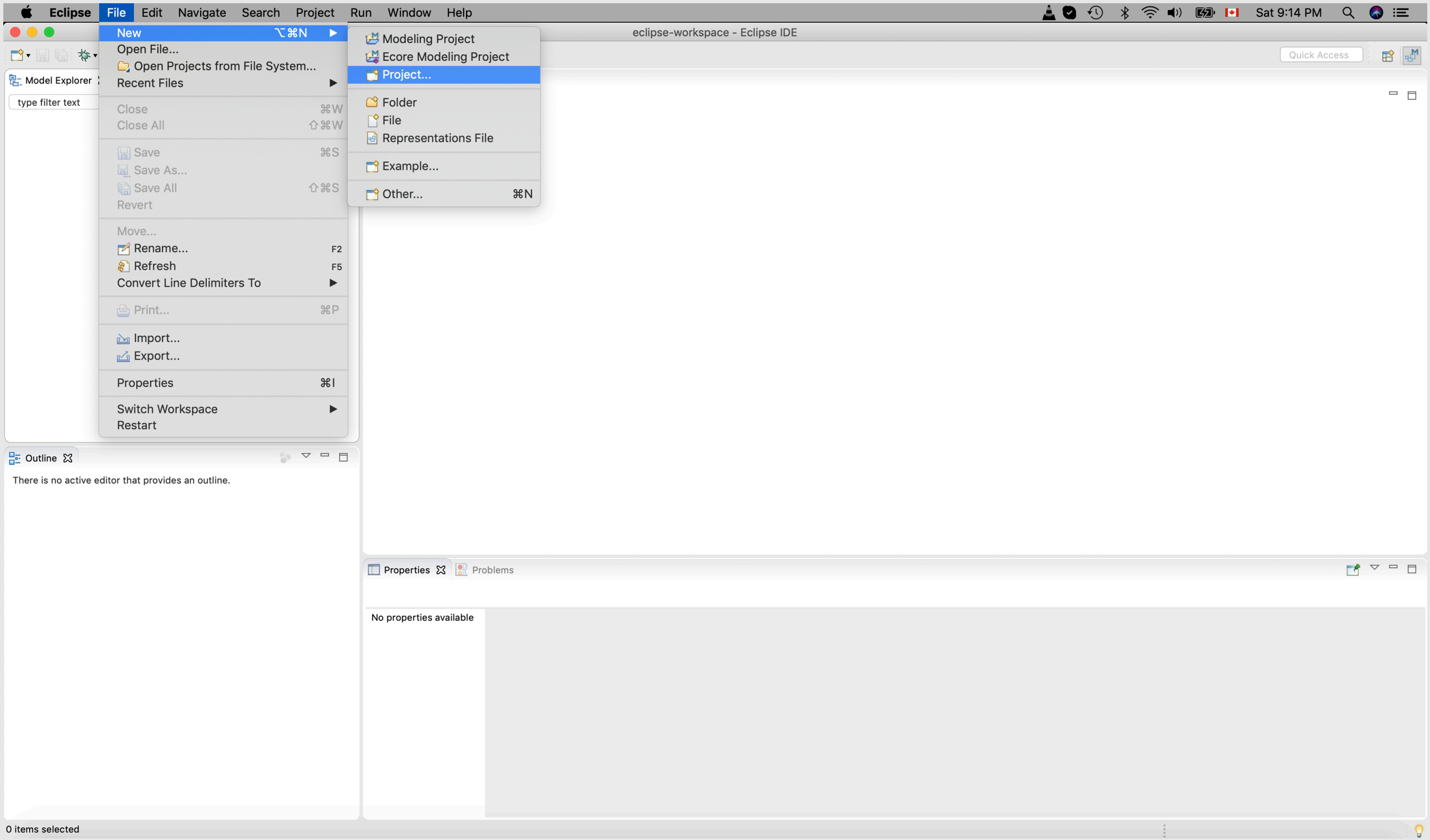


Source: www.eclipse.org

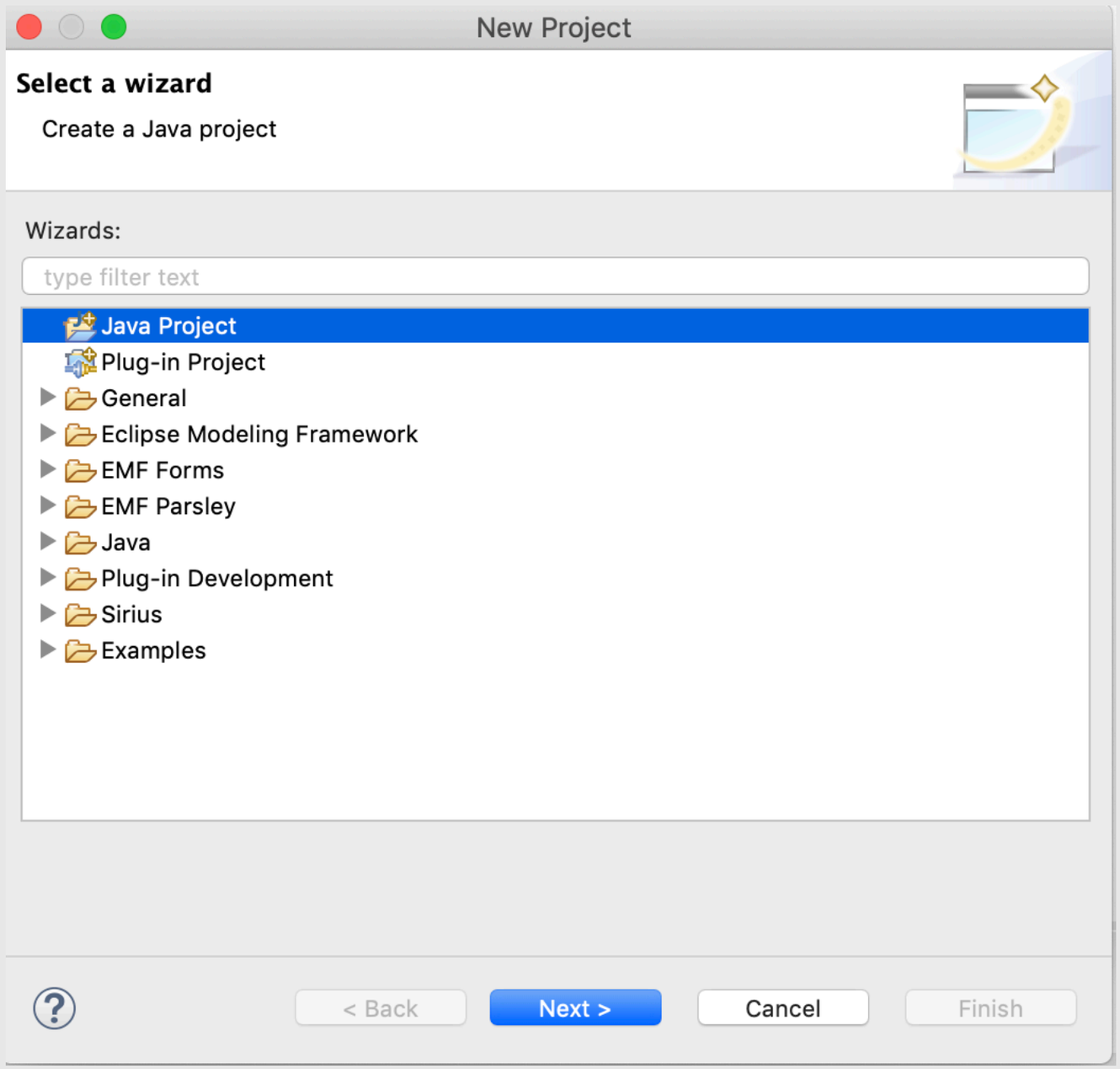
Eclipse Projects

- 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.

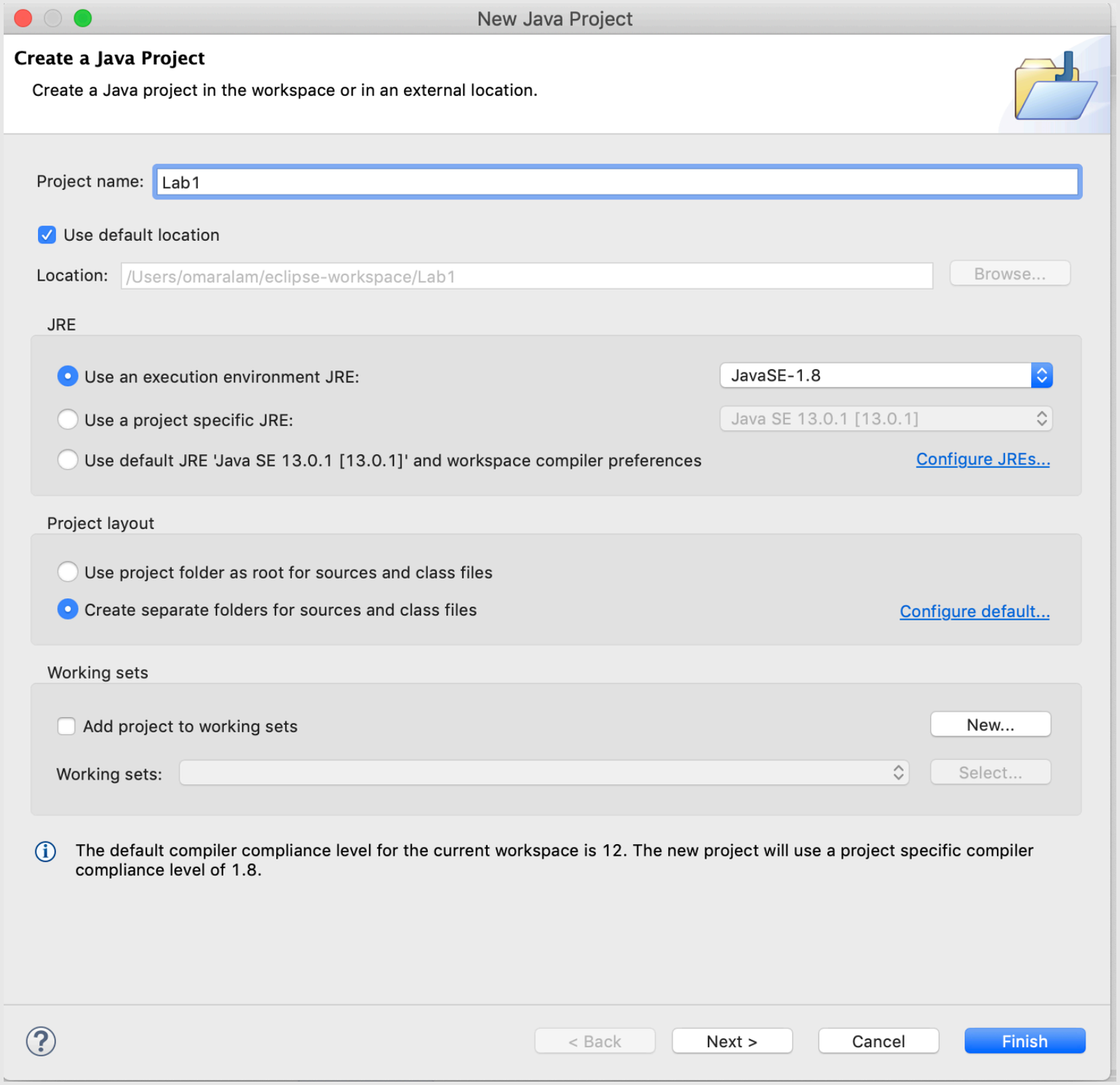
Open Eclipse and click on File>New>Project



From the wizard select Java Project



Type your project name and click Finish

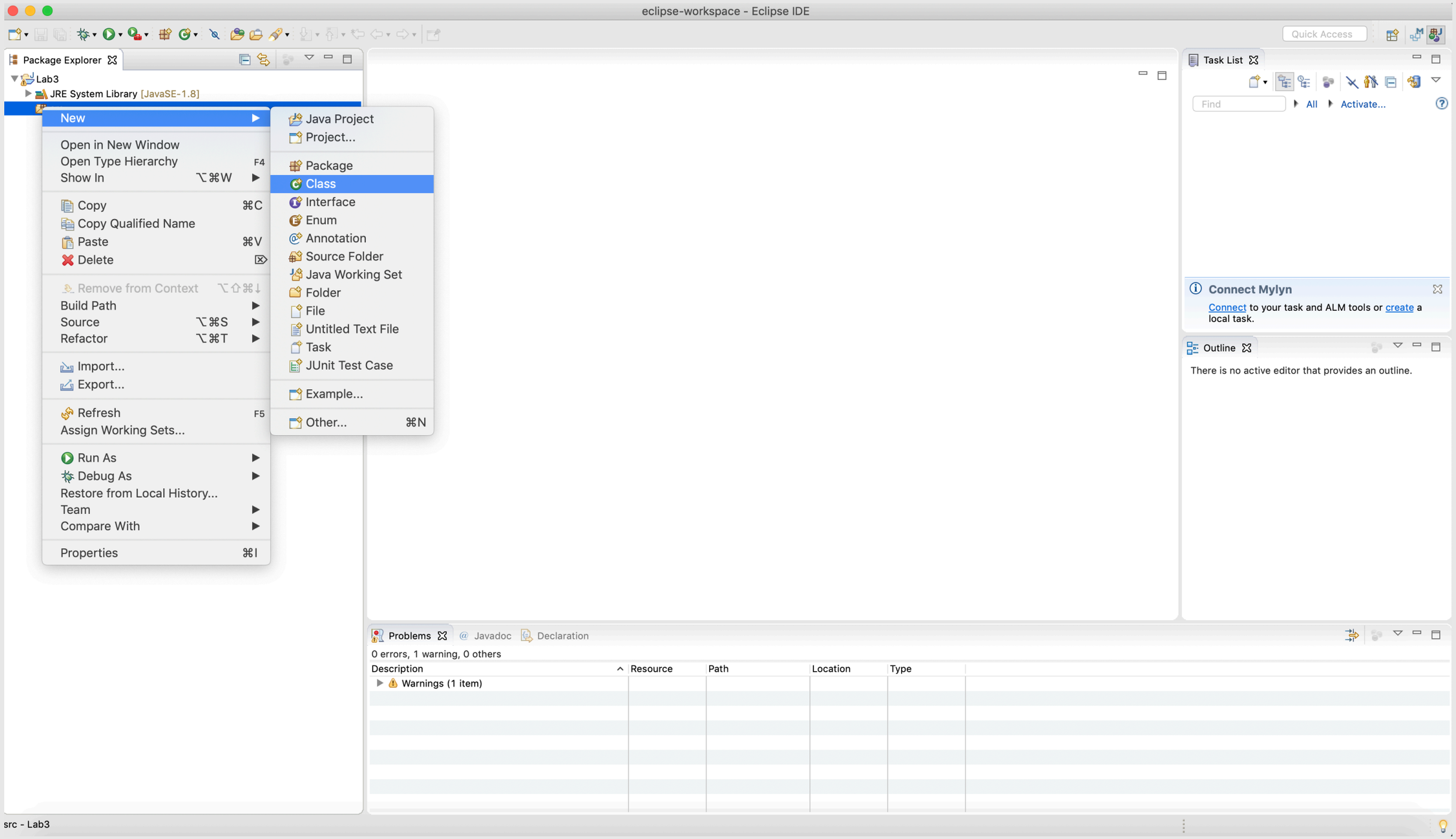


Click on Open Perspective

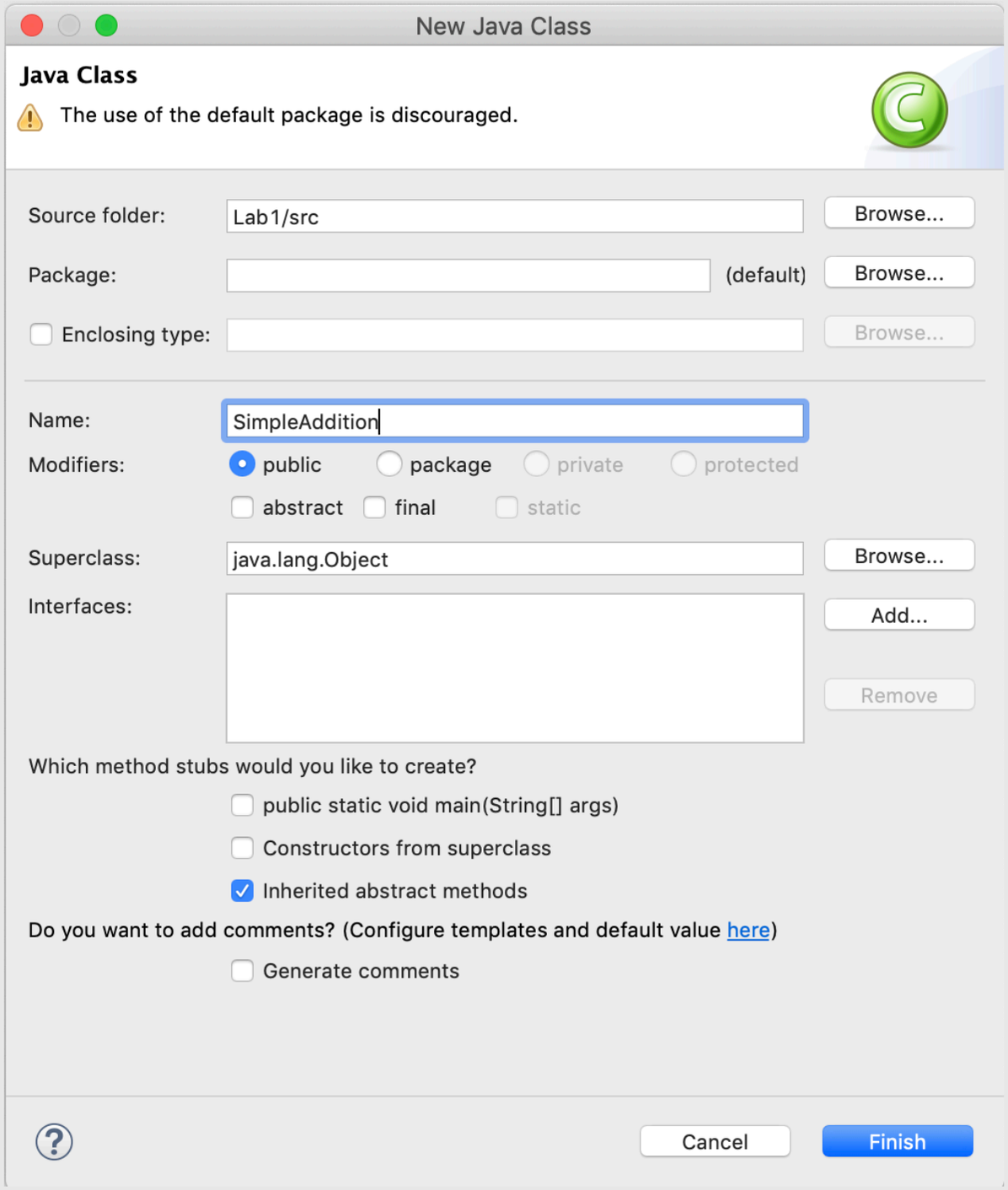
Eclipse Projects

- 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.

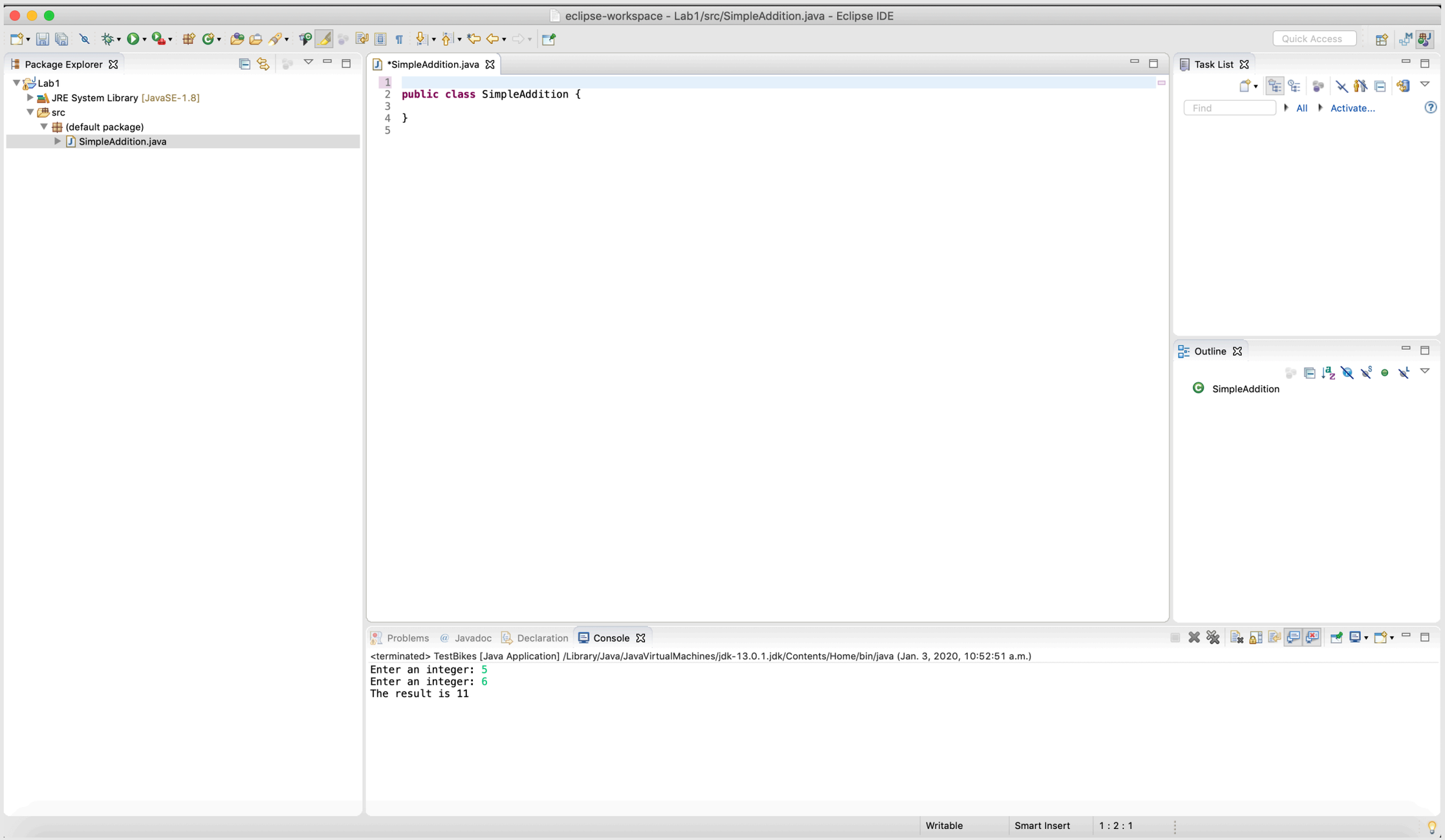
Right-click on src, then select New>Class



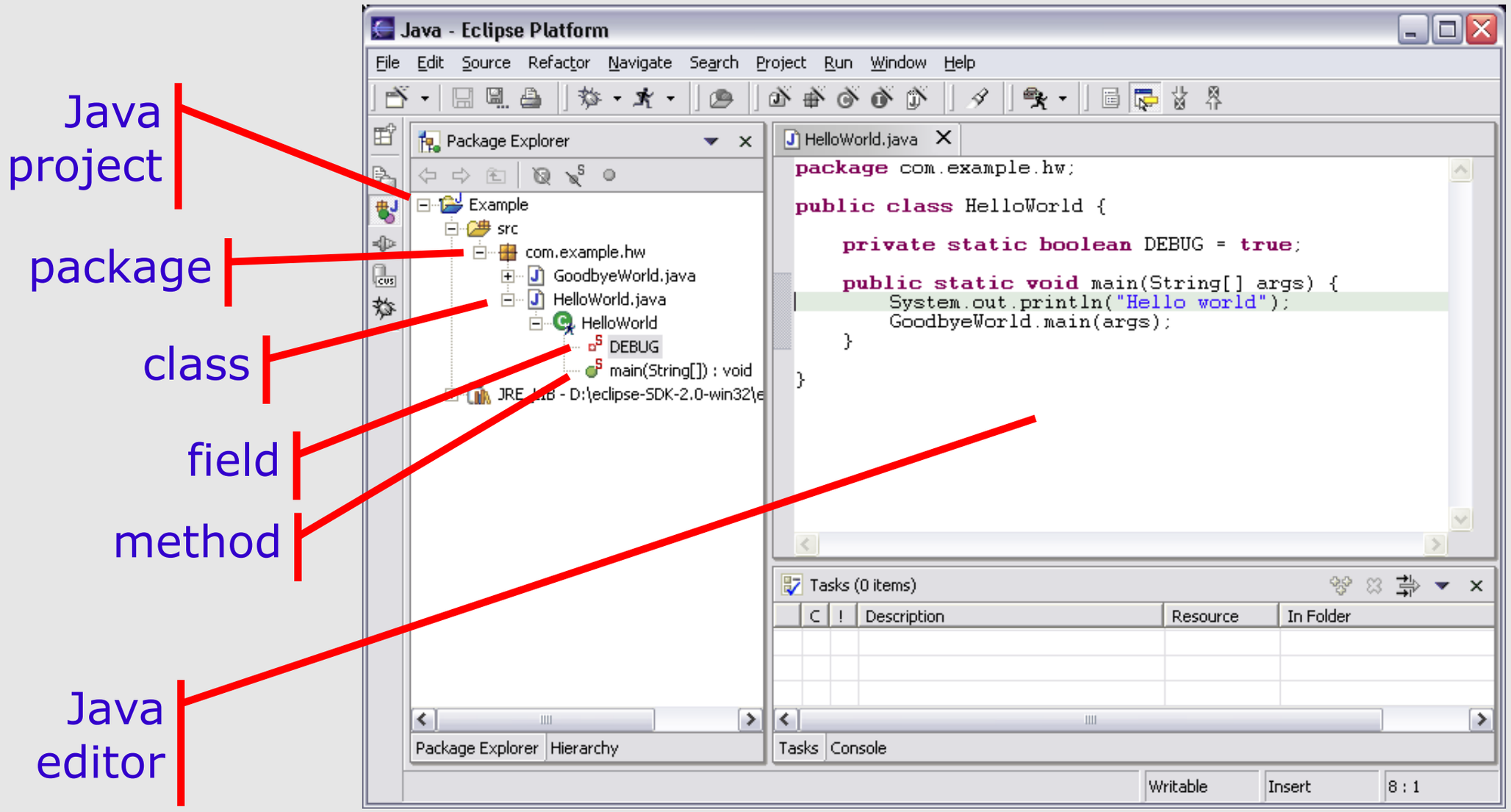
Give your class a name and then click Finish



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



Java Perspective

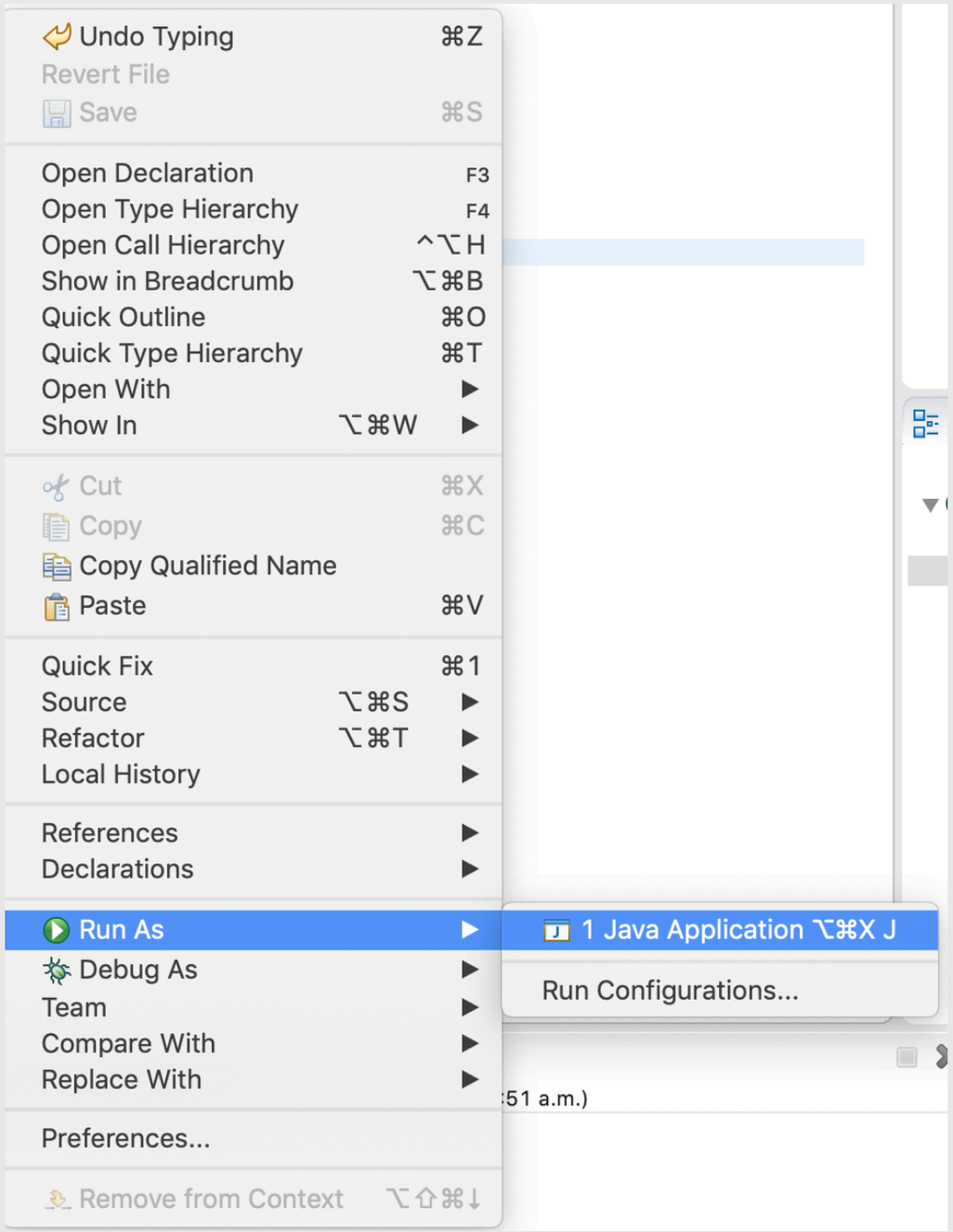


Write a simple code that takes two numbers and outputs the addition of them in Java. Ask for help if you need. You need to write a main function. The first input is shown to you here.

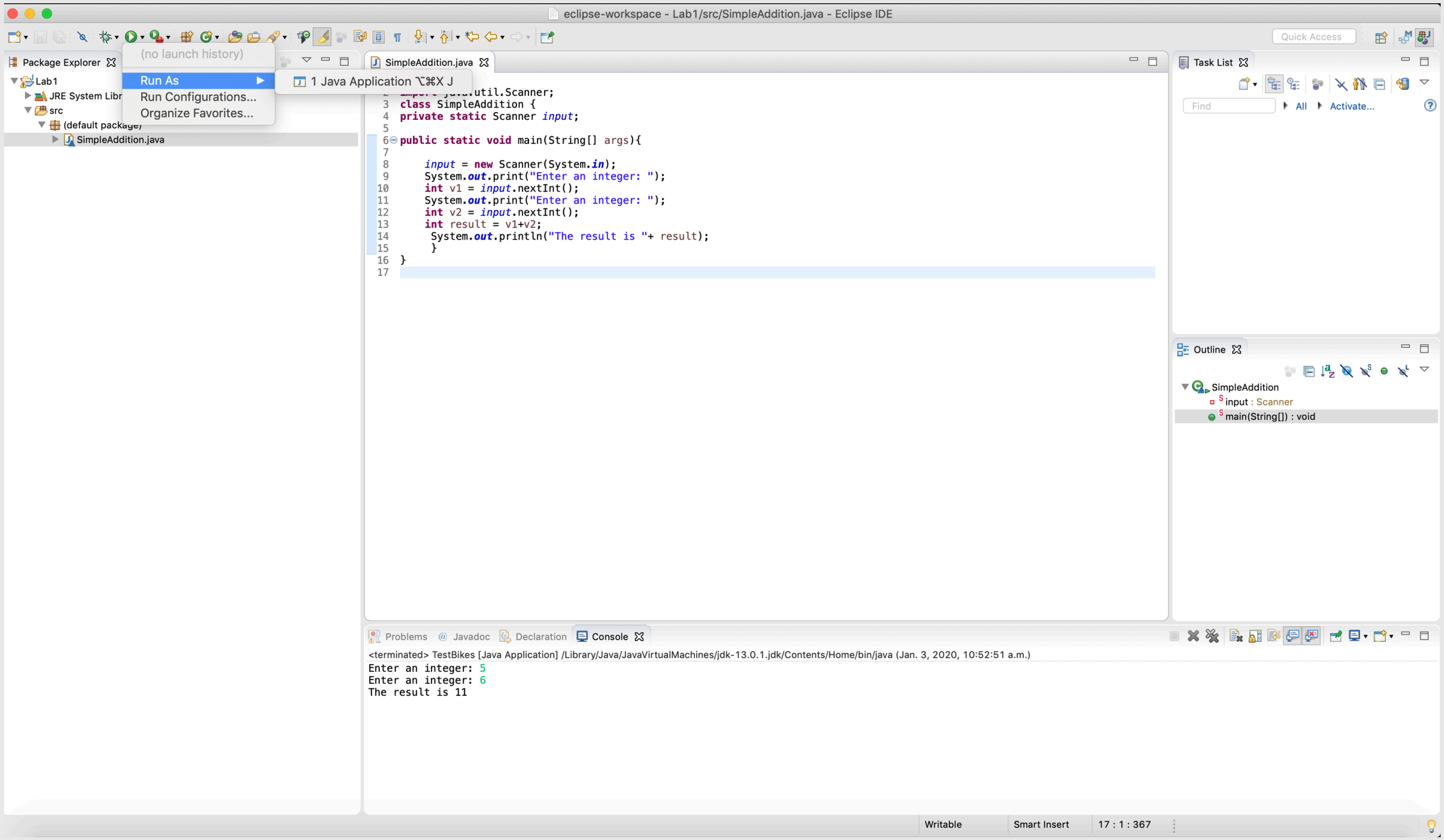
```
import java.util.Scanner;
class SimpleAddition {
private static Scanner input;

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..

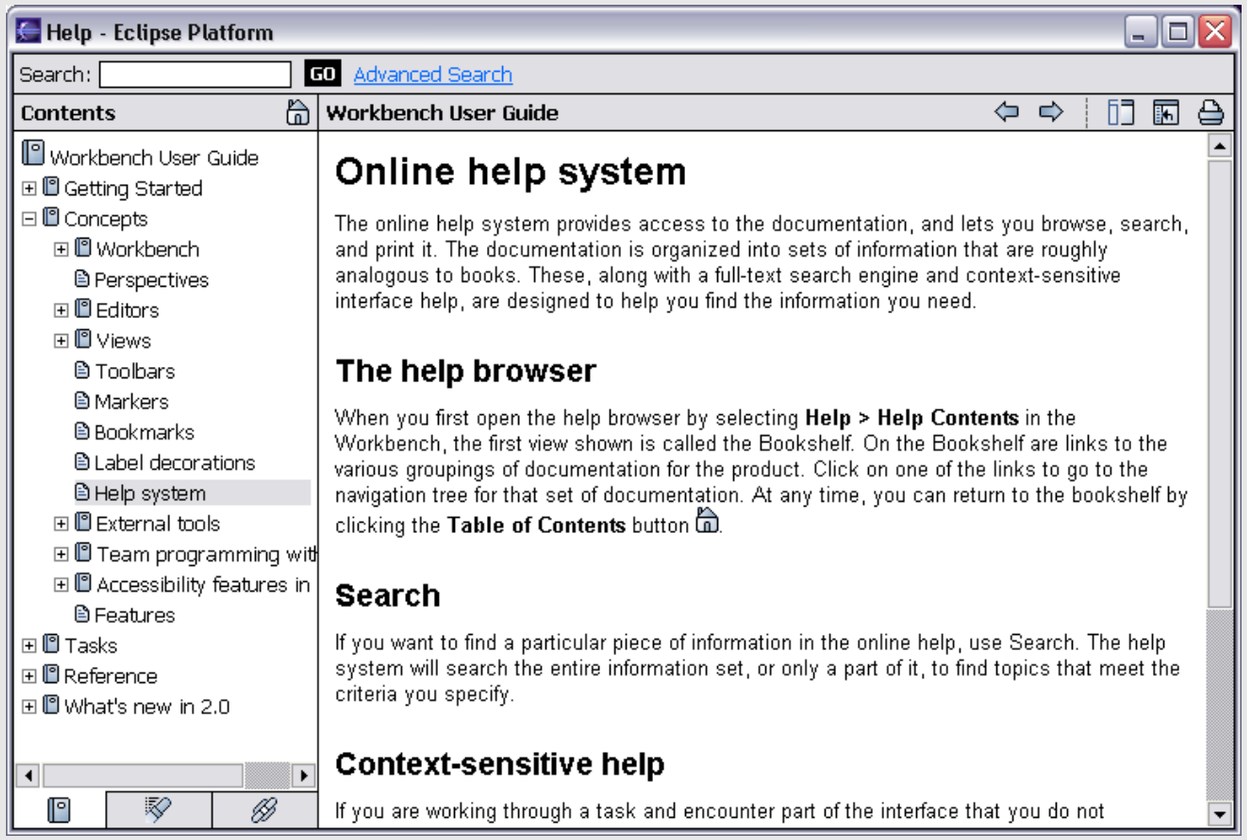
To run your code, right-click on your code and then select Run As->Java Application.



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



Eclipse Help

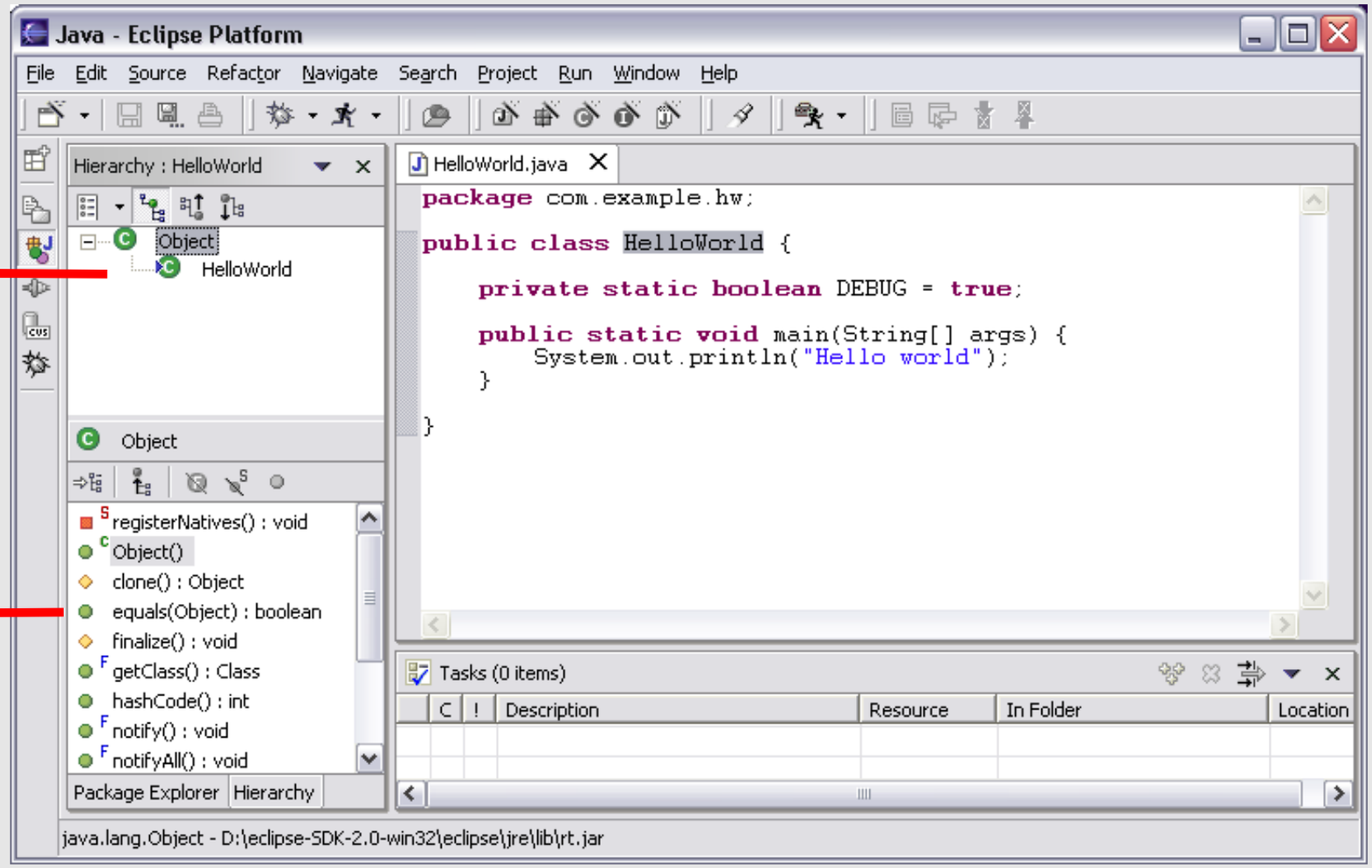


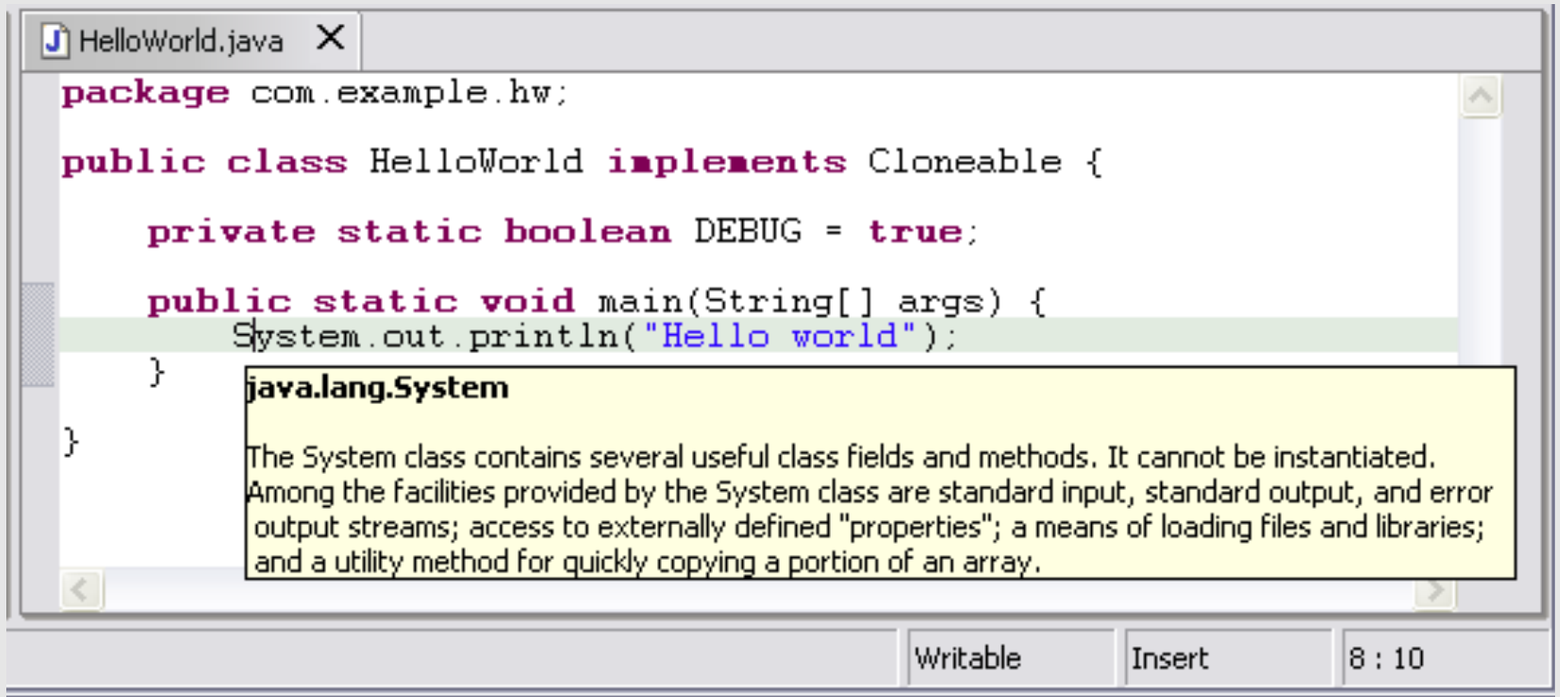
Source: www.eclipse.org

Java Perspective

Type hierarchy

Selected type's members





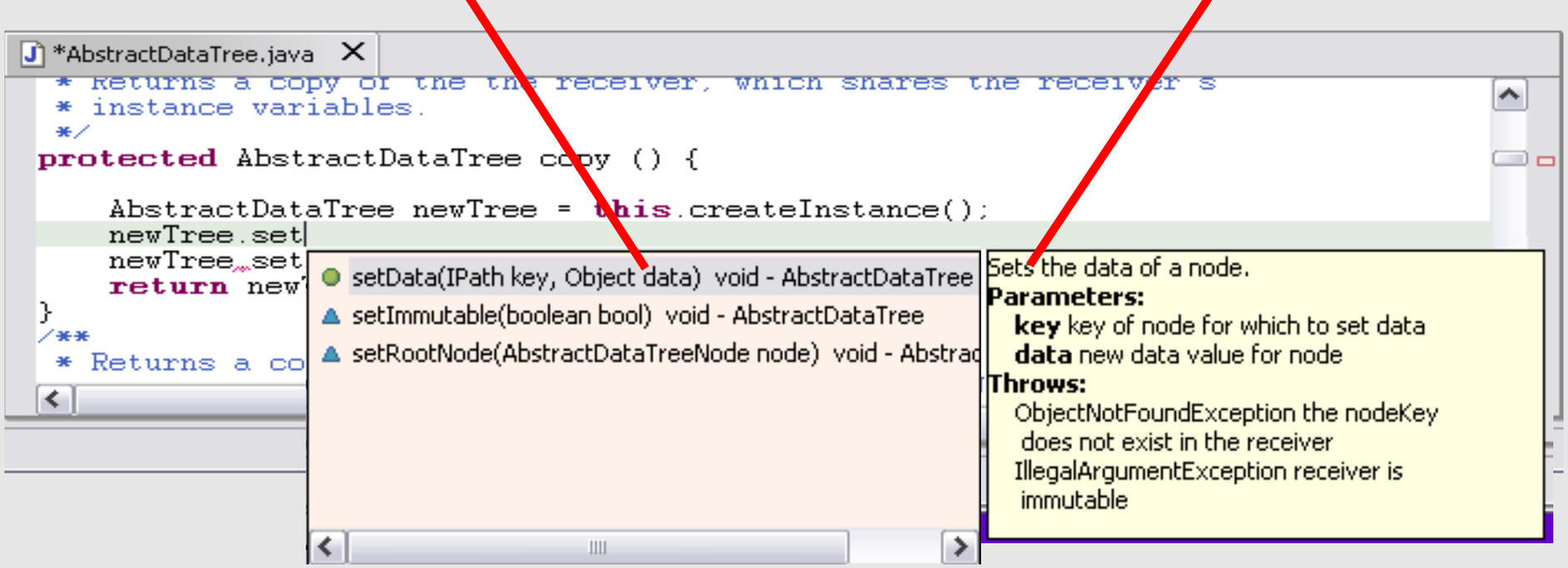
Hovering over identifier shows Javadoc spec

Java Editor

- Method completion in Java editor

List of plausible methods

Doc for method



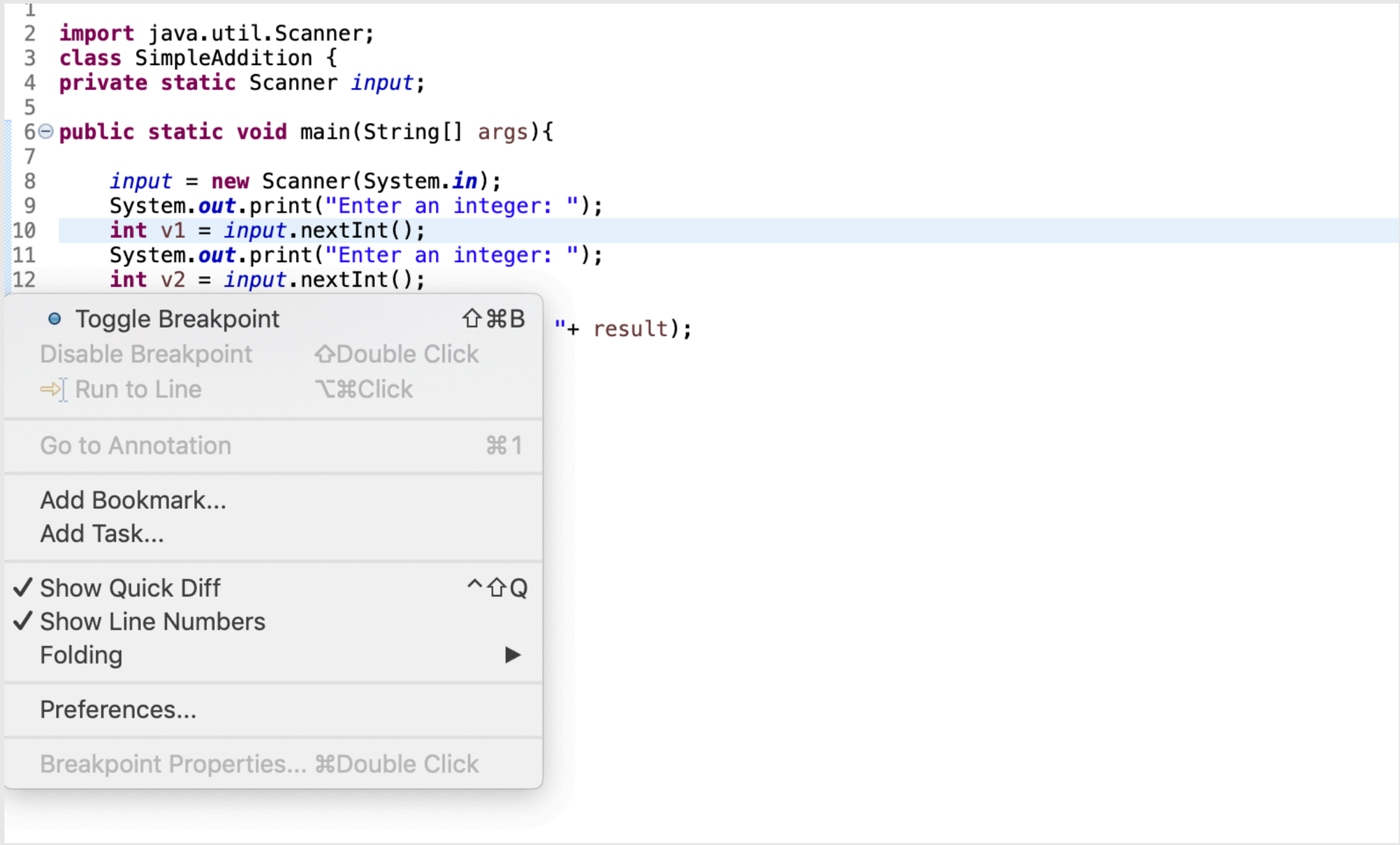
Eclipse Debugger

- 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

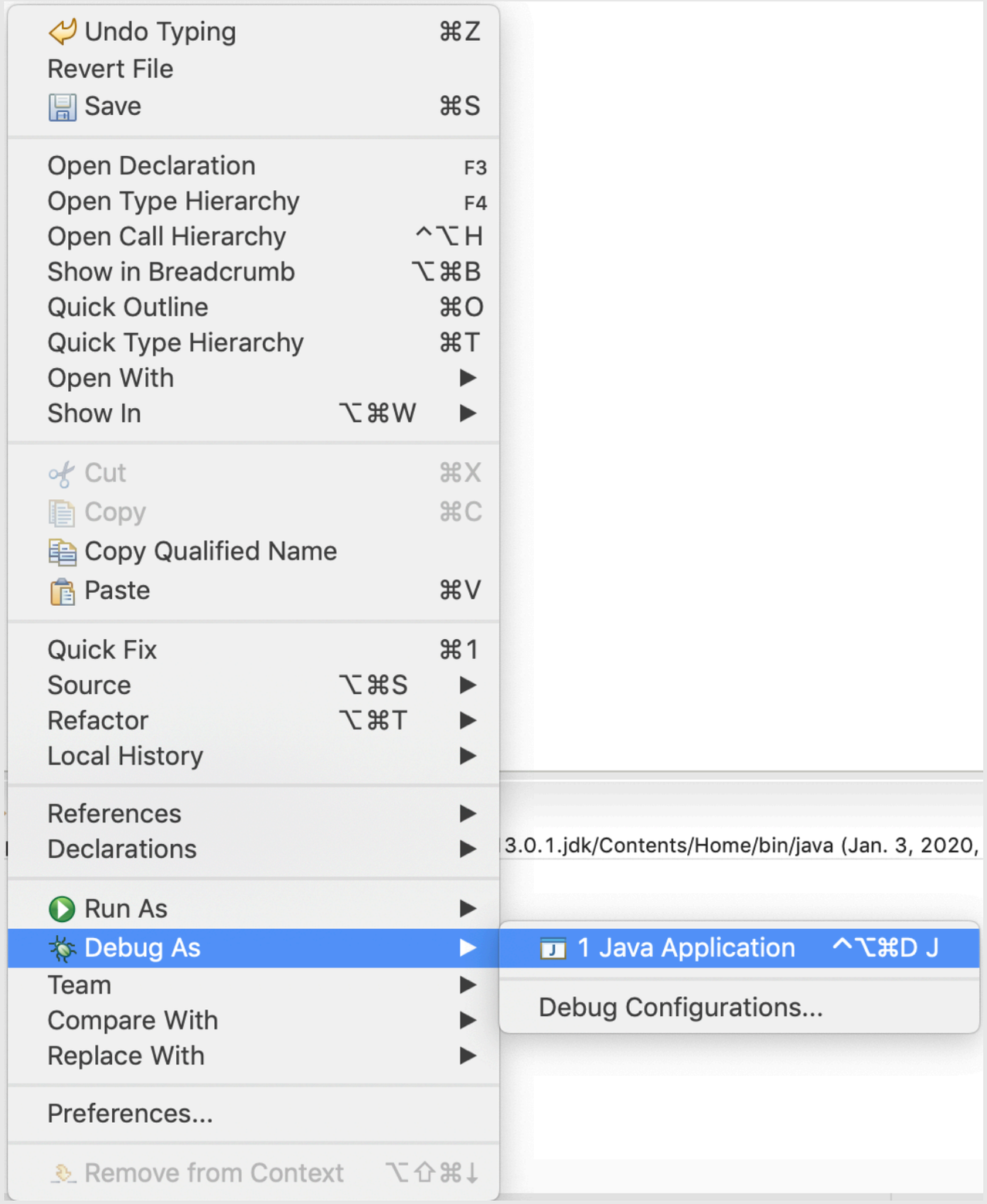
Eclipse Debugger



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

Eclipse Debugger

Right click, then select
Debug As->Java Application



Eclipse Debugger

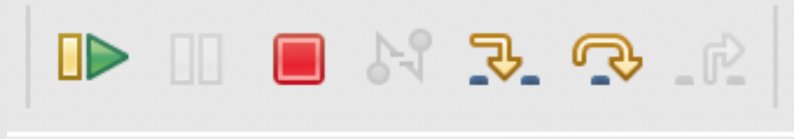
A small blue-dot will show up

```
1
2  import java.util.Scanner;
3  class SimpleAddition {
4  private static Scanner input;
5
6  public static void main(String[] args){
7
8      input = new Scanner(System.in);
9      System.out.print("Enter an integer: ");
10     int v1 = input.nextInt();
11     System.out.print("Enter an integer: ");
12     int v2 = input.nextInt();
13     v1++;
14     int result = v1+v2;
15      System.out.println("The result is "+ result);
16      }
17 }
18
```

Eclipse Debugger

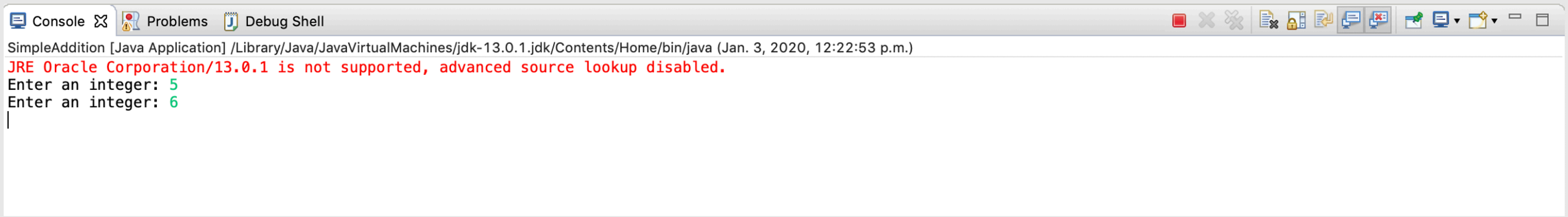
When the debugger is running, Eclipse will expect that you are going to Navigate the code line-by-line. There are two kinds of “stepping’:
1-Step into: which will go to the details of each line, starting with the toggled line.
2-Step over: which will not go to the details of the line.

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.



Eclipse Debugger

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