Building Your First Junit 5 Test

This lesson provides step by step hands-on approach to run JUnit 5 test case on Eclipse IDE.

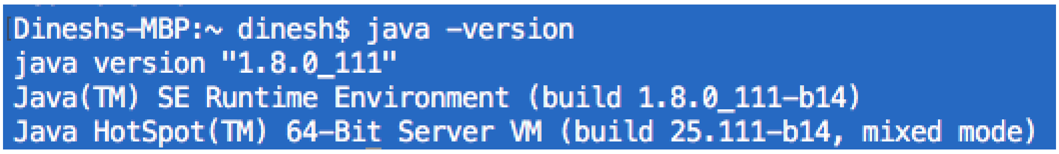
Project Setup

Please install the following softwares on your machine.

Java 8

JUnit 5 tests will run with Java 8 or above. Please download and install the latest update of Java 8 or above from the official Oracle web site. www.oracle.com/technetwork/java/javase/downloads/index.html .

After installing Java, you can check your Java version by running the following command -



```
Dineshs-MBP:~ dinesh$ java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)
```

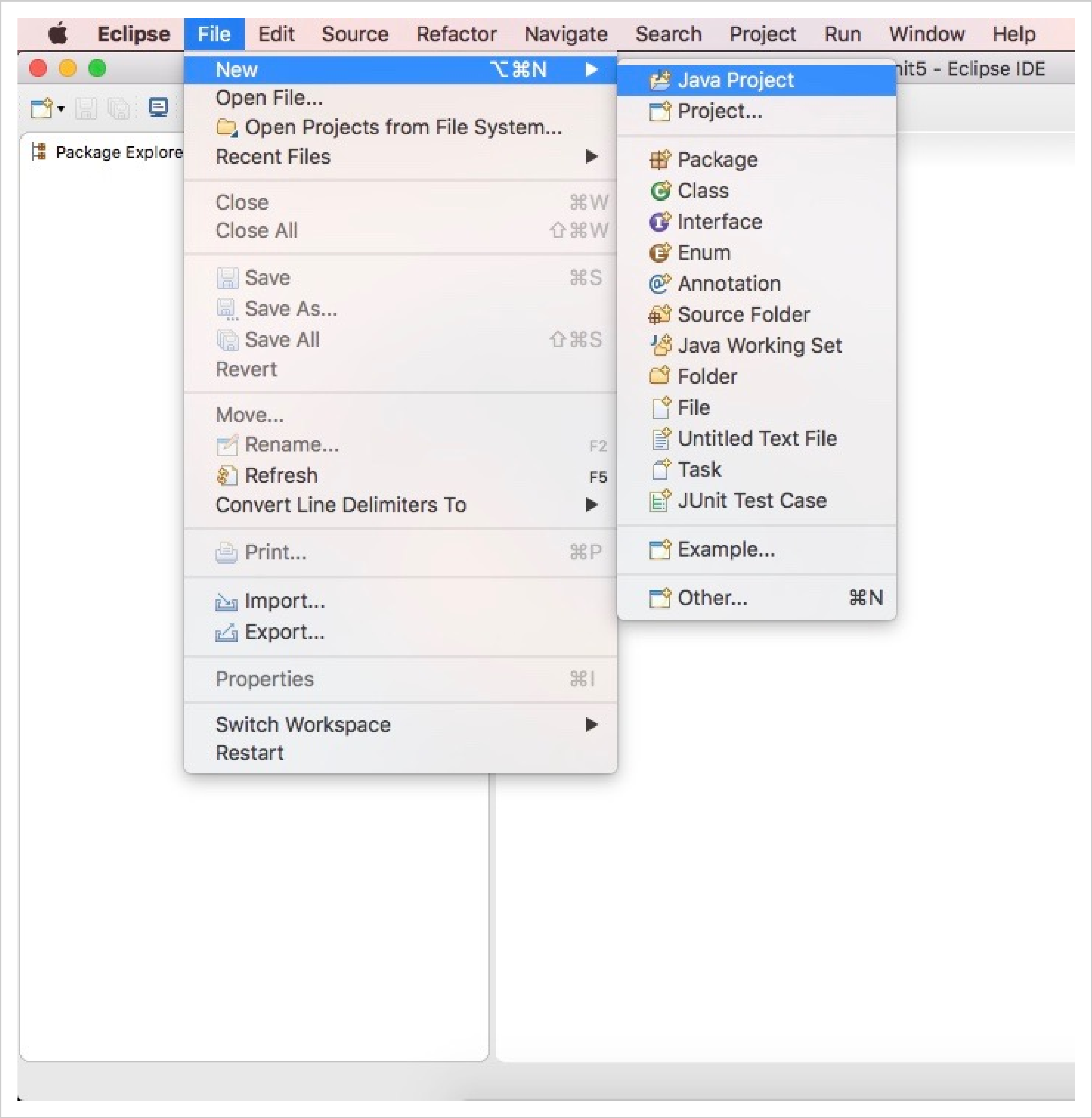
Eclipse Oxygen or above

You can download the latest version from Eclipse web site - https://www.eclipse.org/downloads/packages/release/neon/3/eclipse-ide-java-ee-developers .

Steps to write your first Junit 5 test case in Eclipse

Step 1 - Launch Eclipse IDE.

Step 2 - Create a new Java Project. Right click on File --> New --> Java Project as demonstrated in the figure below.



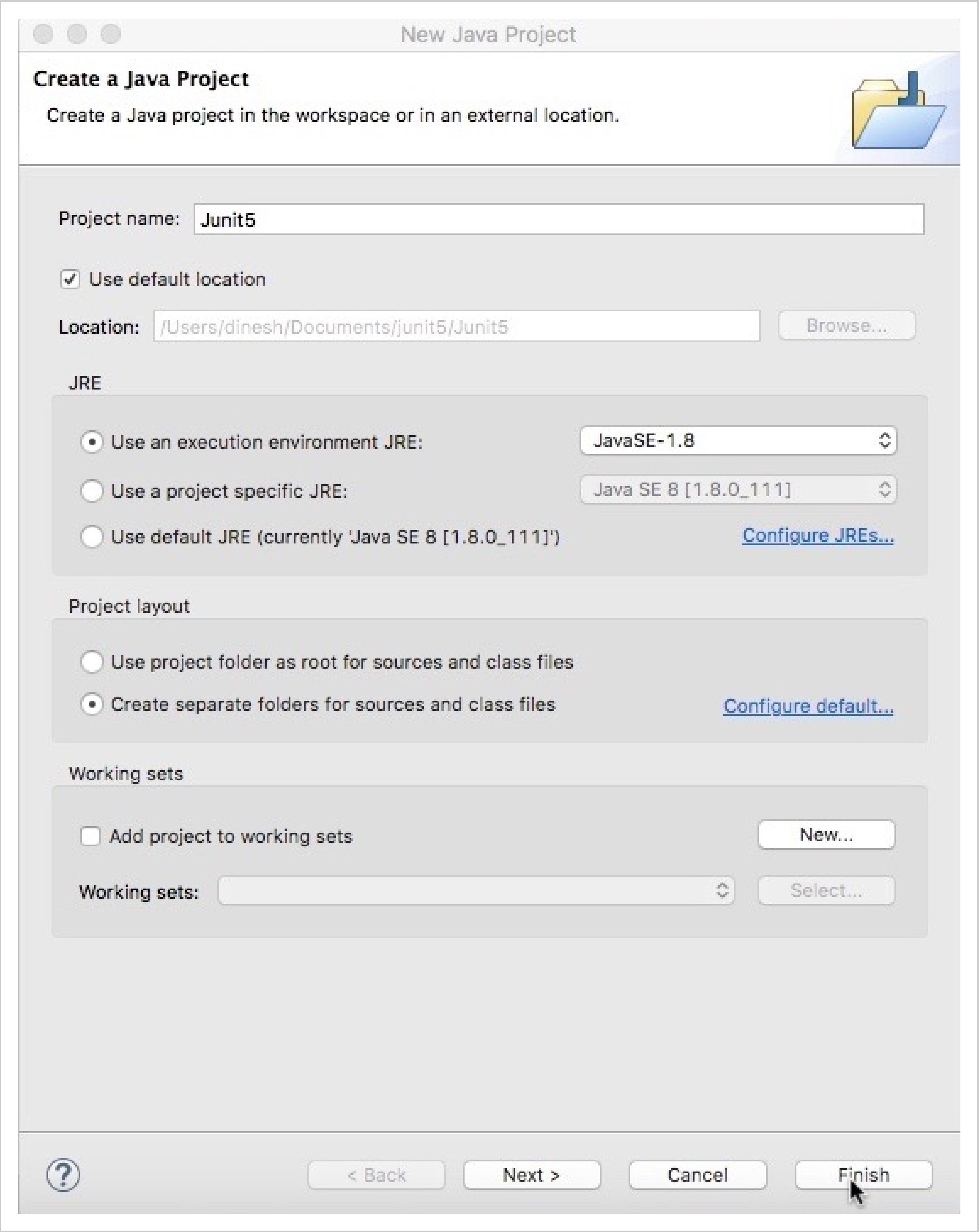
A new popup window by name “New Java Project” will be opened as shown in the figure below.

Step 3 - Provide a Project name: say, **Junit5**

Step 4 - Select JRE as 1.8 and above.

Step 5 - Click Finish.

A new Java Project will be created by name - **Junit5**



New Java Project

Create a Java Project

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

Project name: Junit5

☑ Use default location

Location: /Users/dinesh/Documents/junit5/Junit5

JRE

● Use an execution environment JRE: JavaSE-1.8

○ Use a project specific JRE: Java SE 8 [1.8.0_111]

○ Use default JRE (currently 'Java SE 8 [1.8.0_111]')

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files

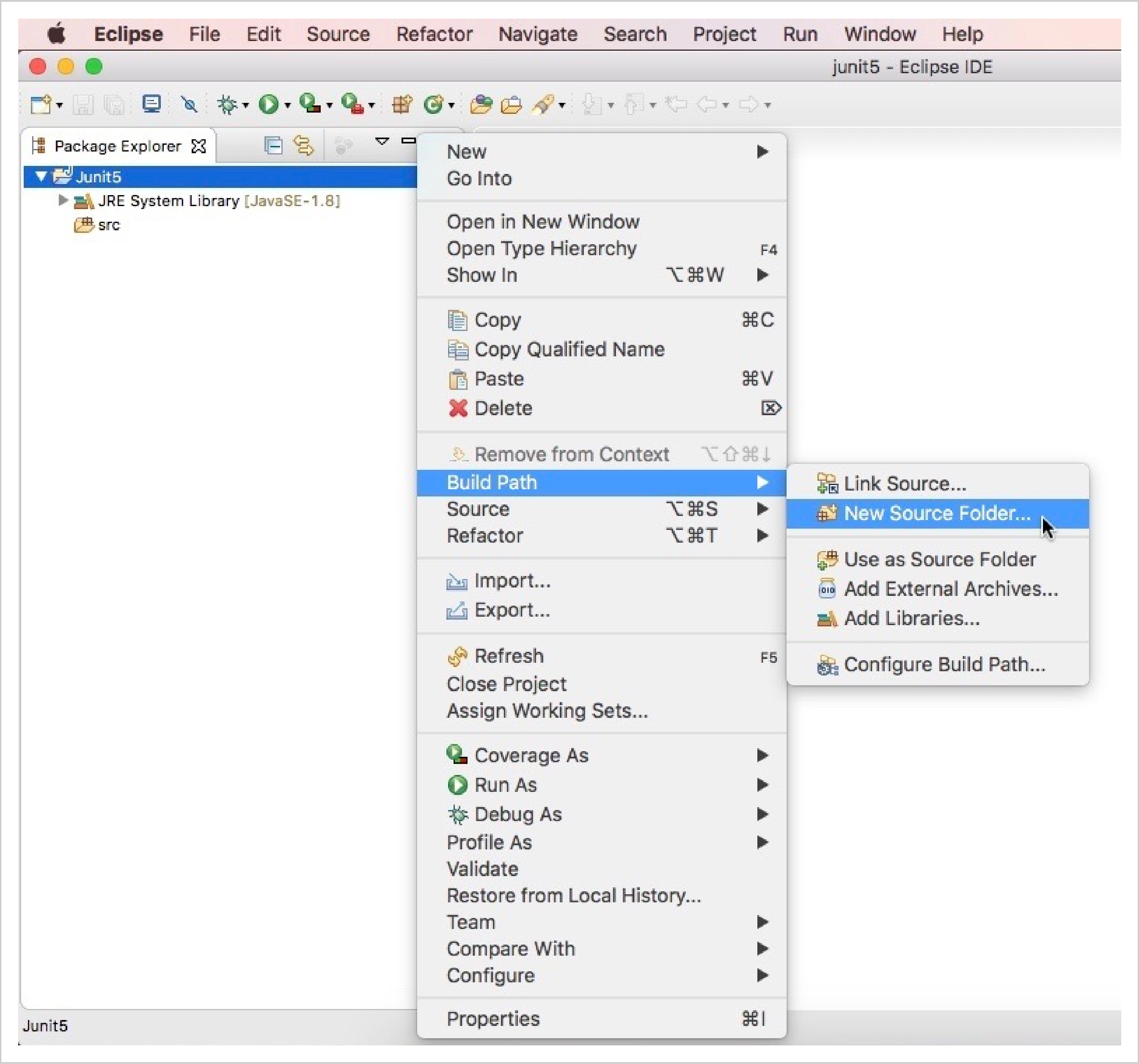
Working sets

□ Add project to working sets

Working sets:

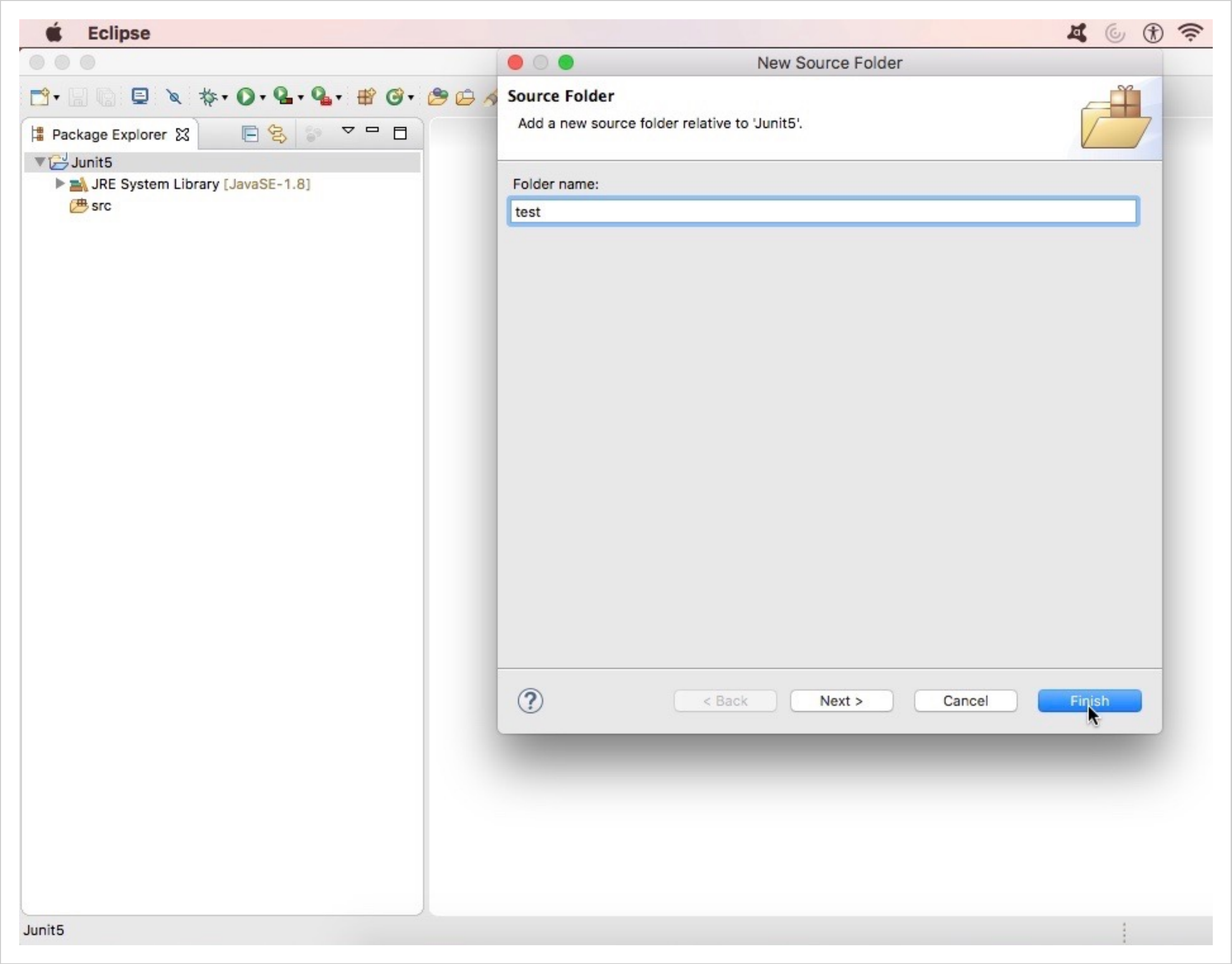
It’s a best practice to create JUnit test cases in the test folder.

Step 6 - Expand the Junit5 Java project. Right-click on project root. Traverse to Build Path --> New Source Folder… as shown in the figure below.



Step 7 - A "New Source Folder" popup will be opened as shown in the figure below.

Step 8 - Provide folder name as "test" and click Finish.



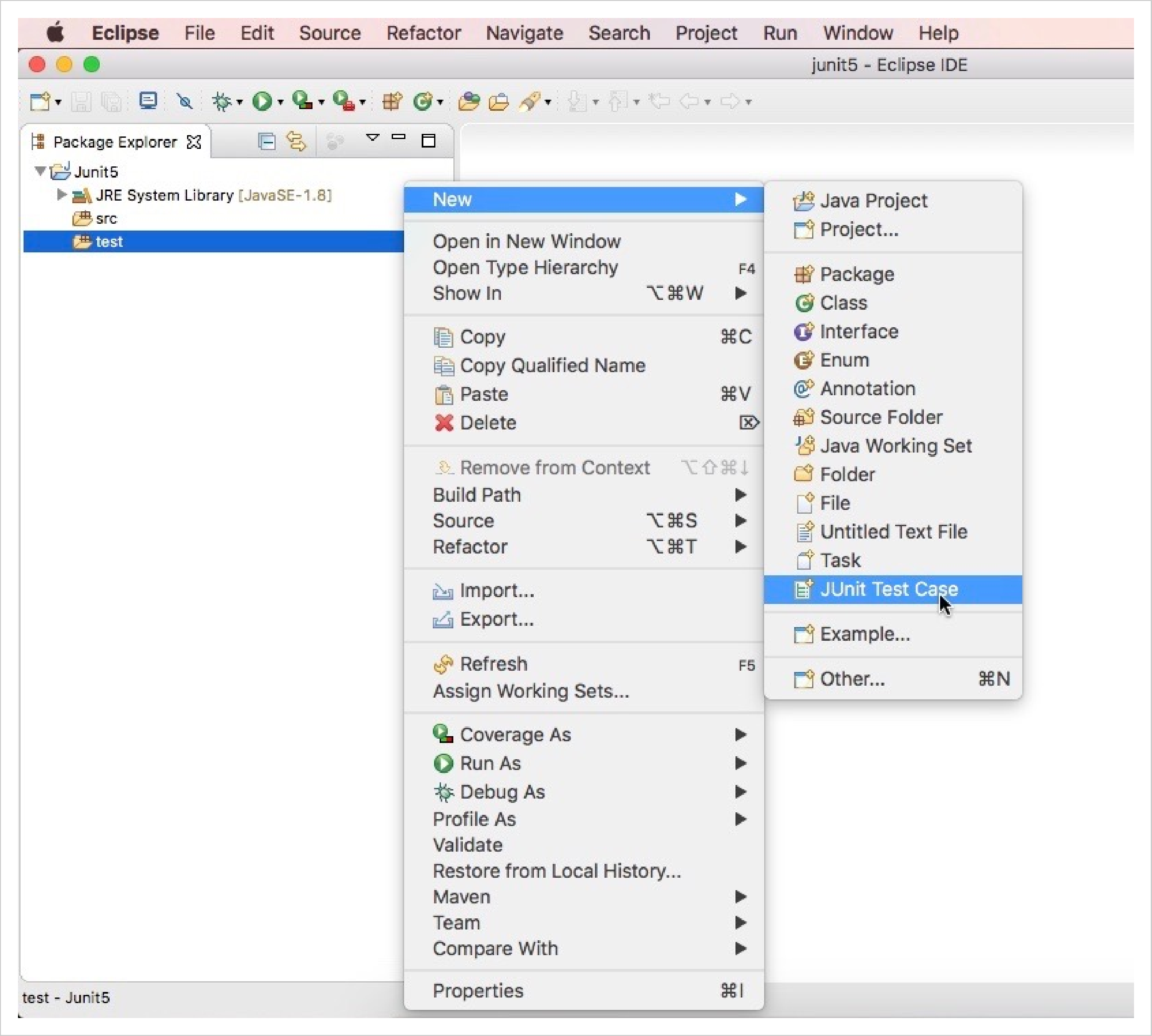
New Source Folder

Source Folder

Add a new source folder relative to 'Junit5'.

Folder name: test

Step 9 - Right-click on test folder and traverse to New --> Junit Test Case as demonstrated in the figure below.



Step 10 - A "New Junit Test Case" popup will be opened as shown in the figure below.

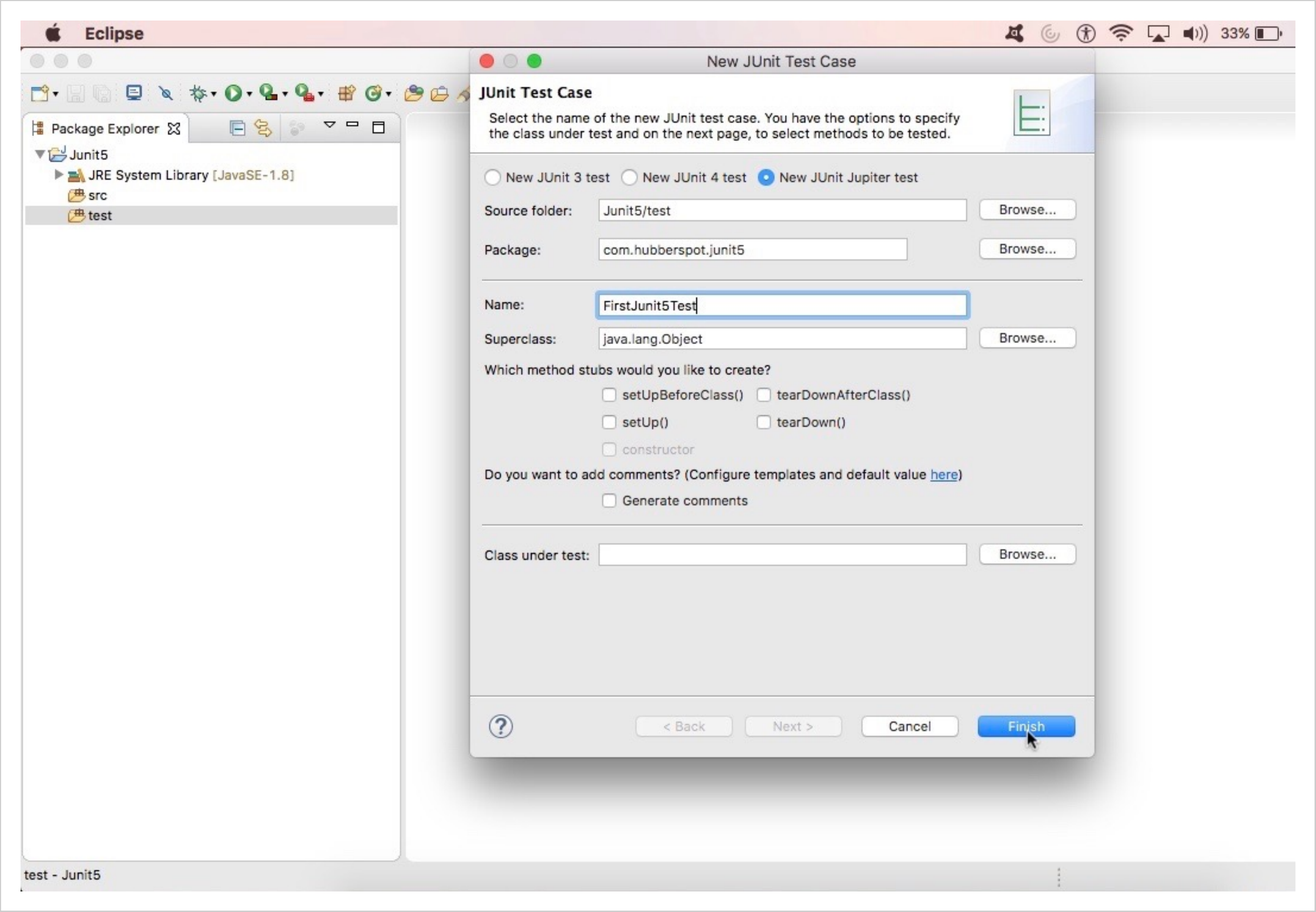
In order to create a new JUnit 5 test case, perform the following steps:-

Step 11 - Select "New Junit Jupiter test". This will create Junit 5 test case.

Step 12 - Provide any package name of your choice.

Step 13 - Provide class name as, FirstJunit5Test.

Step 14 - Click Finish.



New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

○ New JUnit 3 test ○ New JUnit 4 test ● New JUnit Jupiter test

Source folder: Junit5/test

Package: com.hubberspot.junit5

Name: FirstJunit5Test

Superclass: java.lang.Object

Which method stubs would you like to create?

□ setUpBeforeClass() □ tearDownAfterClass()

□ setUp() □ tearDown()

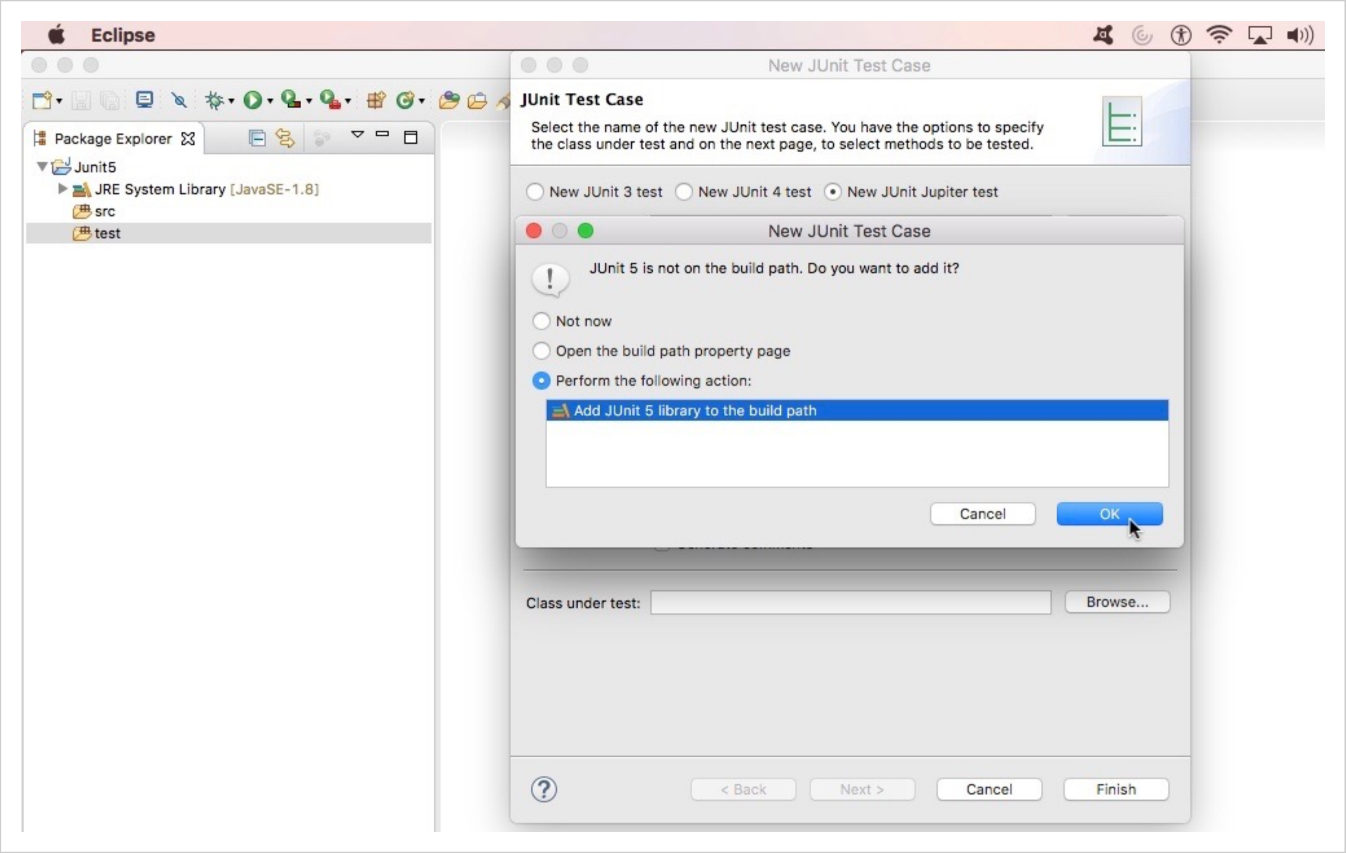
□ constructor

Do you want to add comments? (Configure templates and default value here)

□ Generate comments

Class under test:

Step 15 - Add JUnit 5 library to the build path. Click Ok as shown in the figure below.



New JUnit Test Case

JUnit 5 is not on the build path. Do you want to add it?

○ Not now

○ Open the build path property page

● Perform the following action:

Add JUnit 5 library to the build path

A new test class by name “FirstJunit5Test” will be created. It will have

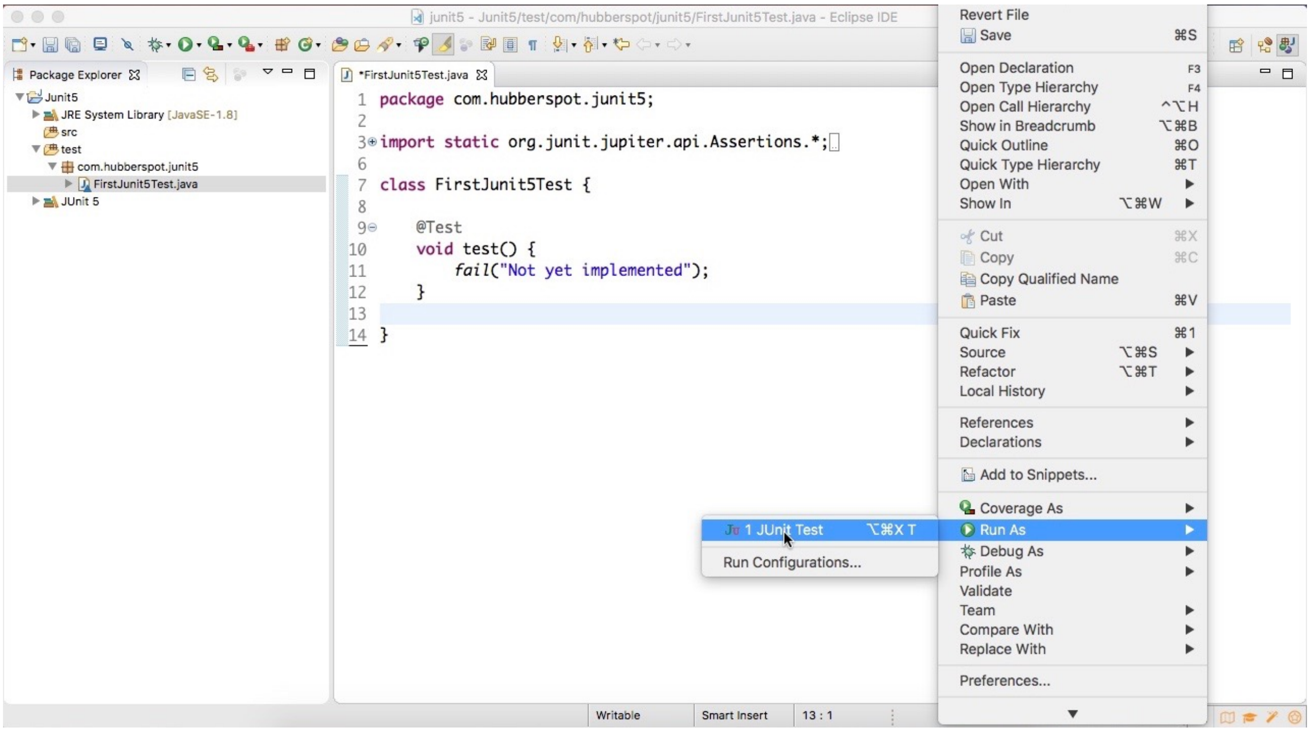
A new test class by name "FirstJunit5Test" will be created. It will have following snippet of code as shown below.

FirstJunit5Test.java

```
package io.educative.junit5;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class FirstJunit5Test {

        @Test
        void test() {
                fail("Not yet implemented");
        }

}
```

It has a method “test()”. This method has an annotation as, @Test. It signifies that this is a test method in to which we write our code to test.

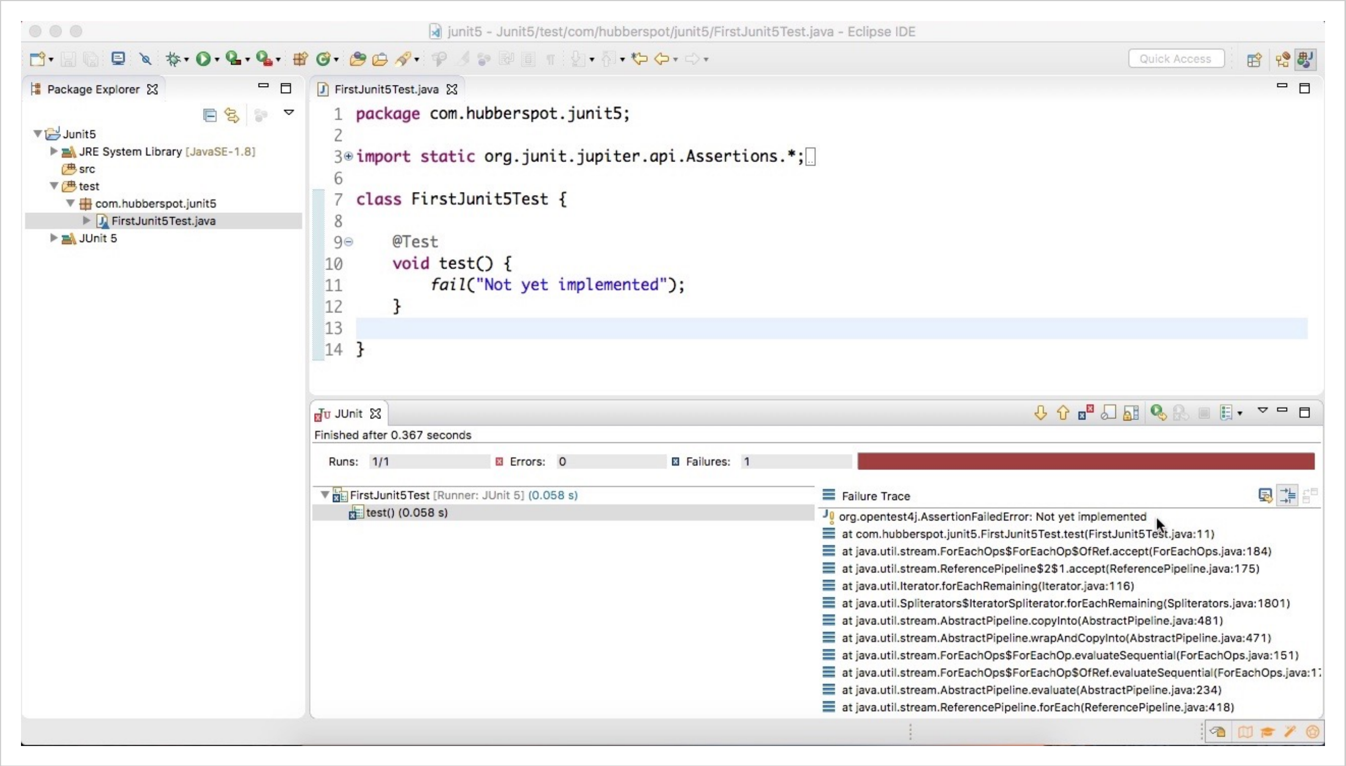
Step 16 - Right-click on editor window and run as Junit test case, as shown in the figure below.



junit5 - Junit5/test/com/hubberspot/junit5/FirstJunit5Test.java - Eclipse IDE

```
package com.hubberspot.junit5;

import static org.junit.jupiter.api.Assertions.*;

class FirstJunit5Test {

    @Test
    void test() {
        fail("Not yet implemented");
    }

}
```

The test case fails as shown in fig below. It gives “AssertionFailedError: Not yet implemented”. It is because in test() method it is written ‘fail(“Not yet implemented")' assertion. This assertion fails the test case as it is not yet implemented. More on @Test and fail() in upcoming lessons.



```java
package com.hubberspot.junit5;

import static org.junit.jupiter.api.Assertions.*;

class FirstJunit5Test {

    @Test
    void test() {
        fail("Not yet implemented");
    }

}
```

```
Finished after 0.367 seconds
Runs: 1/1    Errors: 0    Failures: 1

FirstJunit5Test [Runner: JUnit 5] (0.058 s)
    test() (0.058 s)

org.opentest4j.AssertionFailedError: Not yet implemented
at com.hubberspot.junit5.FirstJunit5Test.test(FirstJunit5Test.java:11)
at java.util.stream.ForEachOps$ForEachOp$OfRef.accept(ForEachOps.java:184)
at java.util.stream.ReferencePipeline$2$1.accept(ReferencePipeline.java:175)
at java.util.Iterator.forEachRemaining(Iterator.java:116)
at java.util.Spliterators$IteratorSpliterator.forEachRemaining(Spliterators.java:1801)
at java.util.stream.AbstractPipeline.copyInto(AbstractPipeline.java:481)
at java.util.stream.AbstractPipeline.wrapAndCopyInto(AbstractPipeline.java:471)
at java.util.stream.ForEachOps$ForEachOp.evaluateSequential(ForEachOps.java:151)
at java.util.stream.ForEachOps$ForEachOp$OfRef.evaluateSequential(ForEachOps.java:1
at java.util.stream.AbstractPipeline.evaluate(AbstractPipeline.java:234)
at java.util.stream.ReferencePipeline.forEach(ReferencePipeline.java:418)
```

Building Your First JUnit 5 Test

Java Unit Testing with JUnit 5

Building Your First JUnit 5 Test

Dinesh Varyani
https://www.hubberspot.com

Building Your First JUnit 5 Test

In the next lesson, we will look into `@Test` annotation in JUnit 5.