ALGONQUIN COLLEGE

OBJECT ORIENTED PROGRAMMING (JAVA) (CST8284)

LAB 4

UNIT TEST FRAMEWORK (Java Unit Test – JUnit 4)

Important Information

- **Review** the Hybrid tasks specified this week. Content of the hybrid component will be **demonstrated** in the lab.
- To avoid mistakes ensure that you review the hybrid component for Week #4 **first before** starting off with this lab exercise.
- Review some **helpful videos** included in the **Hybrid** section of the Brightspace page to enrich your learning experience.

Some Introduction

The essence of this lab is to learn about, and use a **unit testing** framework.

- Unit testing frameworks are used to **execute** and **evaluate** test suites, as well as make it easy to add test cases.
- In this lab we will use the **JUnit 4**
- Available at: https://junit.org/junit4/

You do not need to install JUnit if using Eclipse

Some Introduction (2)

- In **JUnit** testing, you design a **test case** for **each class** you develop.
- A **method** is provided for each of the test cases you wish to run.
- Your test methods are annotated. **Annotations** are used to **mark** the test methods (advanced Java features that place markers in the code, which is interpreted by another tool).
- In JUnit, **@Test** annotation is used to mark methods. Other methods may exist in the text class without @Test (e.g. methods performing steps to be shared among test methods).

Some Introductions (3)

- For each test case, conditions can be computed to check if **true**.
- Result is passed to a method (**assertEquals** method) that sends it to the framework.
- The **assertEquals** method takes the **expected** and **actual** values as **arguments** (for floating point numbers it takes a **tolerance** value). Method fails if the two given values are unequal.
- It is **good practice** to end the name of the test class with "**Test**", for example: CovidTest

Assertions and their definitions

From your Hybrid tasks for this week, what **assertions** can you **identify** that are used in JUnit testing methods? What are their descriptions?

Hint: assertNull(…), assertNotNull(…), assertTrue(test), assertFalse(test), assertSame(…), etc.

PART 1

YOUR TASKS

ALGONQUIN COLLEGE

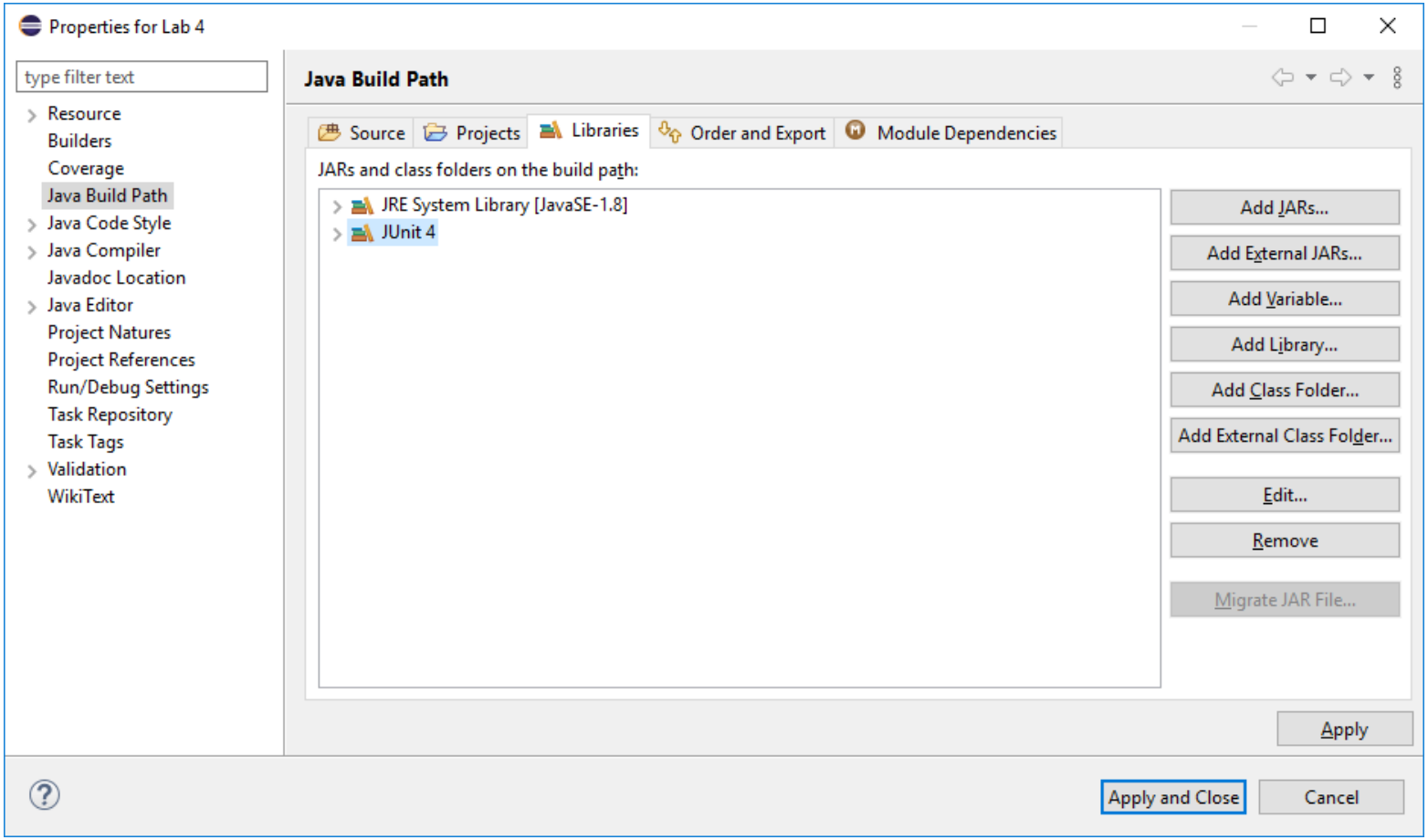
You are Required to…

- **Inspect** the code files that you received and **load them** into eclipse (Lab 4)
- Check to ensure that you have **JUnit 4** appropriately **set up** in your eclipse.
- Do the following if you need to **correctly set up** in eclipse:
  - **Package Explorer** at the top bar of your eclipse editor

You are Required to… Ensure Correct set up of JUnit 4

- Click on **Project**
- Scroll down to **Properties** on the menu
- Scroll down to **Java Build Path** on the menu
- Select **Add Library** from the list on the right-hand side of the page
- Select **JUnit 4**
- Click **Apply and Close**

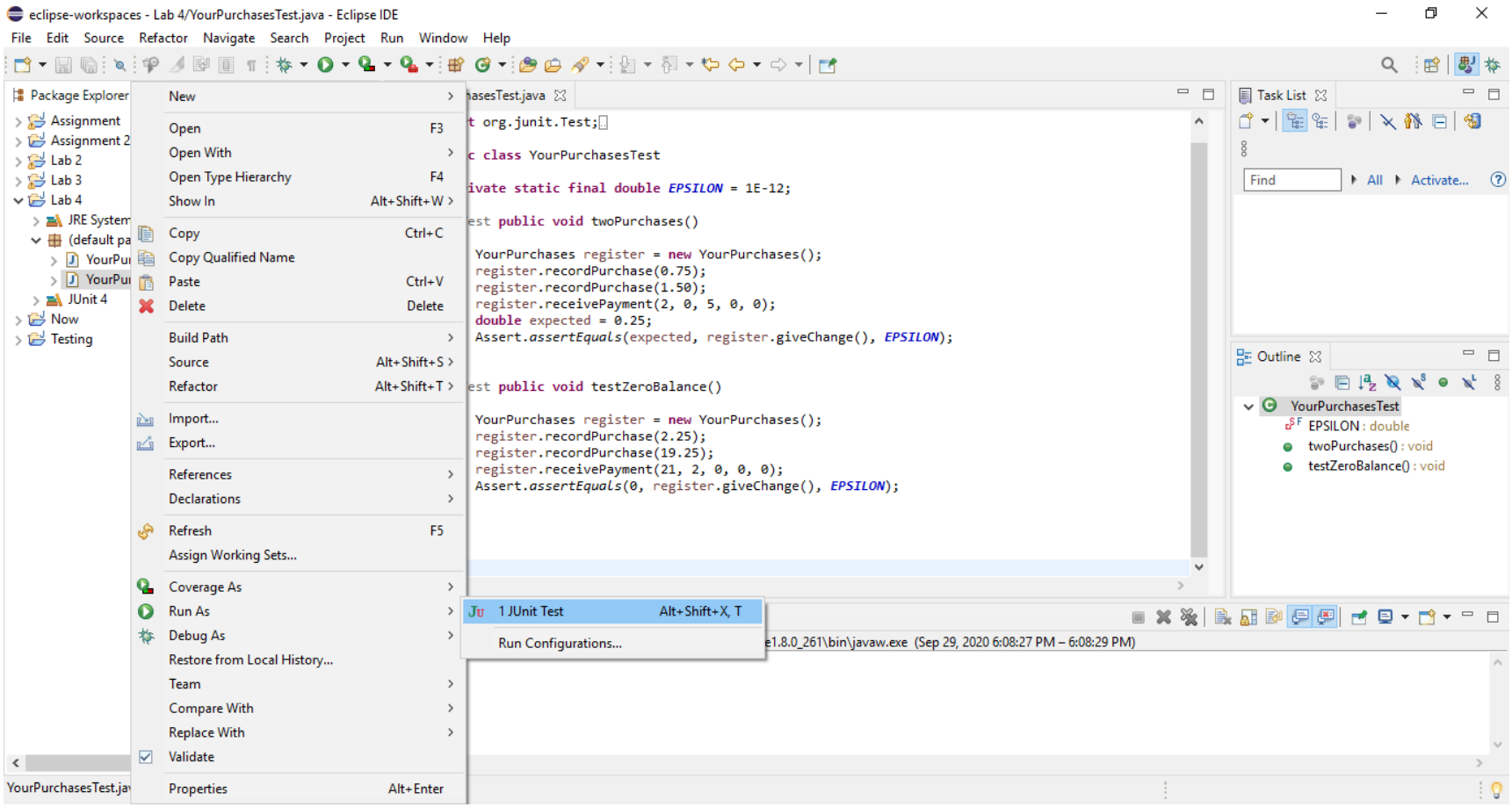
You are Required to… Ensure Correct set up of JUnit 4 (2)



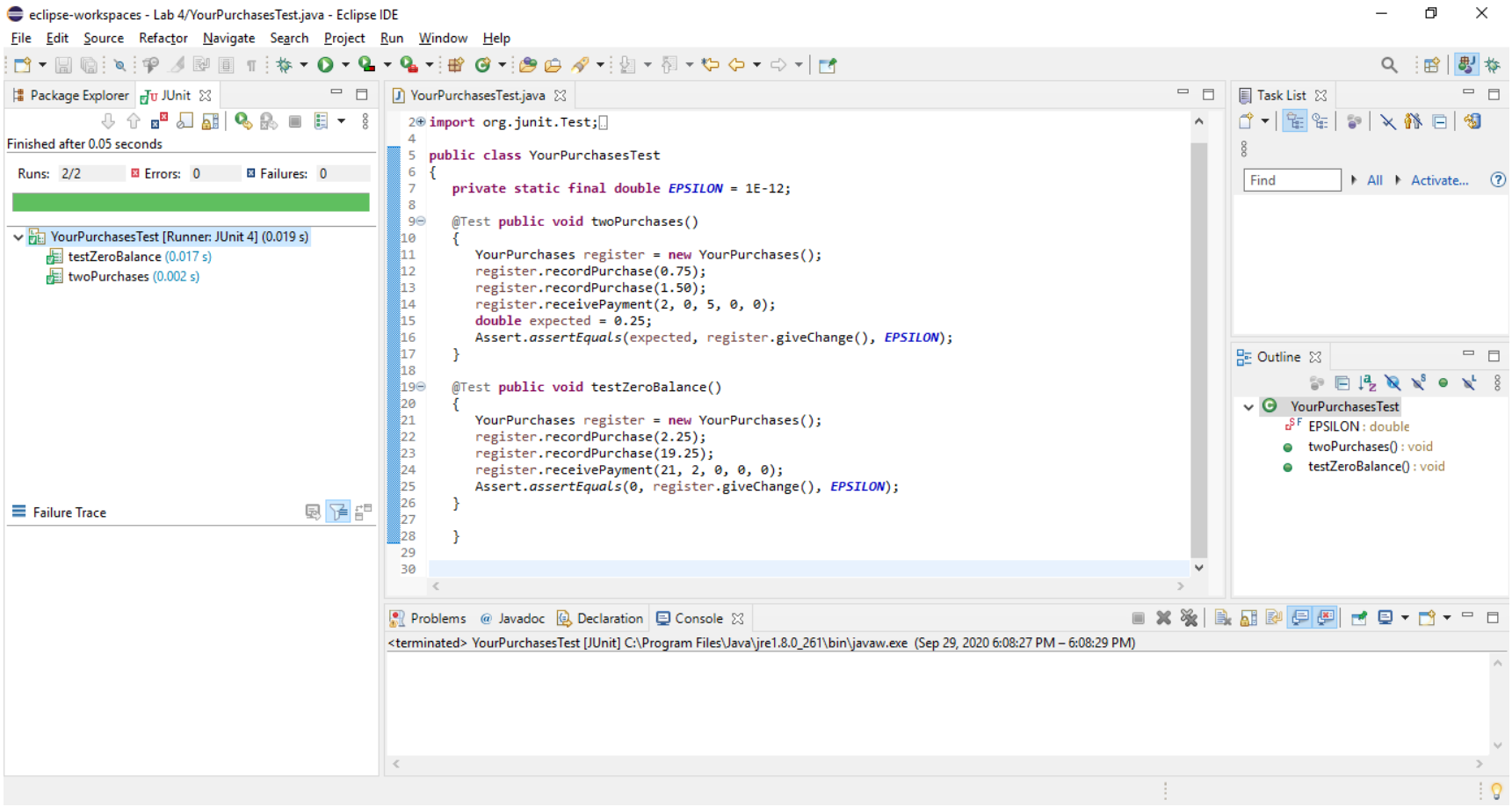
Execute as Junit Test

- To execute, **right click** on the YourPurchasesTest and select **Run As,** and then **JUnit Test** (as shown in slide next page)
- When your execution is **successful**, a **green bar** shows in the panel in eclipse, cataloging a **Runs, Errors and Failures** (see slide below)
- When **unsuccessful**, a **red bar** shows in the panel and the catalog of Runs, Errors and Failures (see slide below)

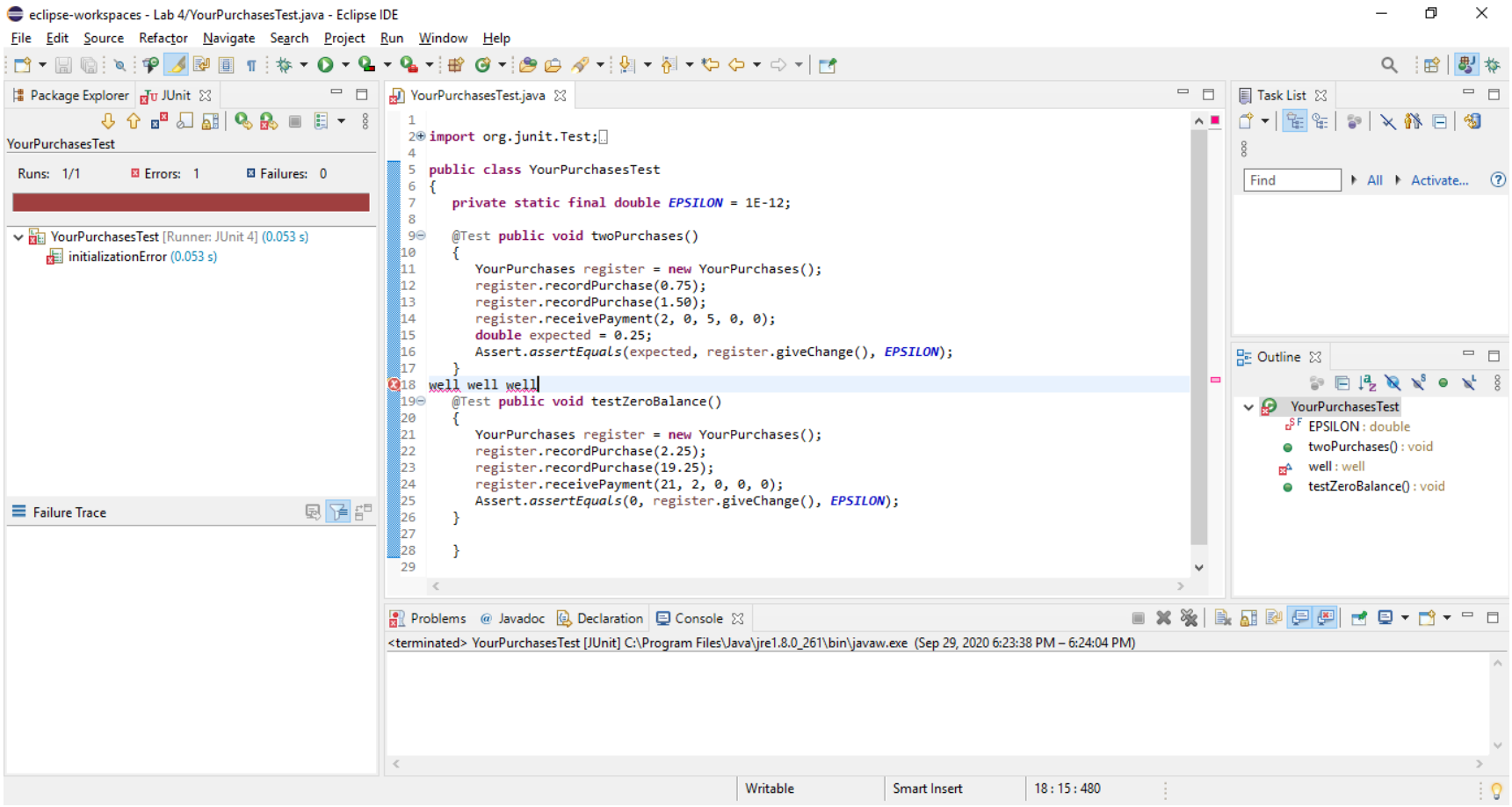
Execute as JUnit Test



When your run is successful…



Error is encountered: unsuccessful run



Your Demo For Part 1

- Run the code provided to you: (**YourPurchases.java** and **YourPurchasesTest.java**)
  - **Explain** the output on the execution screen to your professor
- Modify the code you received to show **unsuccessful** execution (with errors).
  - **Explain** the output on the execution screen to your professor
- **Improve the code** you received by adding **just two** Test Cases to the already **existing** Test Suite. Use a different form of assertion (not assertEquals) in at least one of your tests.
- Run your **improved Test suite** to show a successful run, and explain **any changes** you may have made.

PART 2

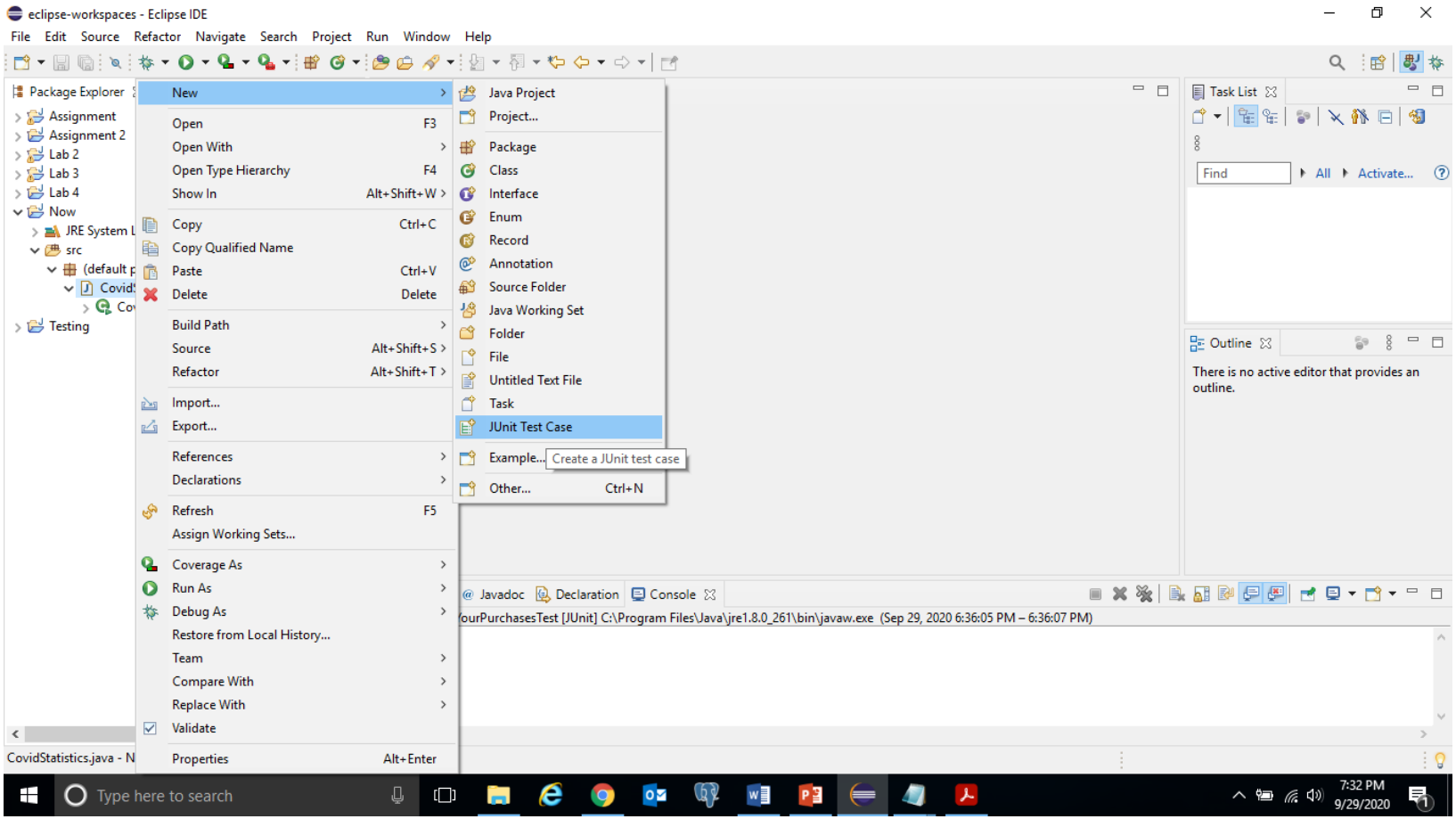
CREATING A NEW TEST CASE

ALGONQUIN COLLEGE

Creating a New Test Case

- In your eclipse editor, select the source code for which the Test case is required
- Make a **right click** on the source file to display a long menu window (as shown in next slide).
- At the **top** of the menu, select **New,** and then scroll down to select **Junit Test Case**

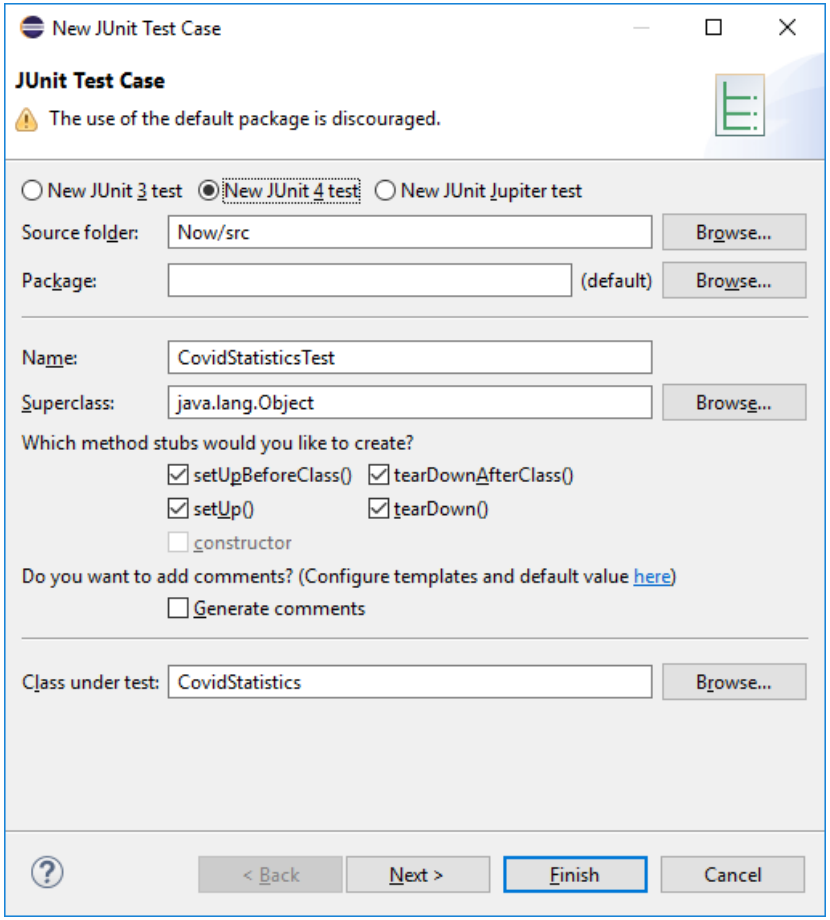
Creating a New Test Case (2)



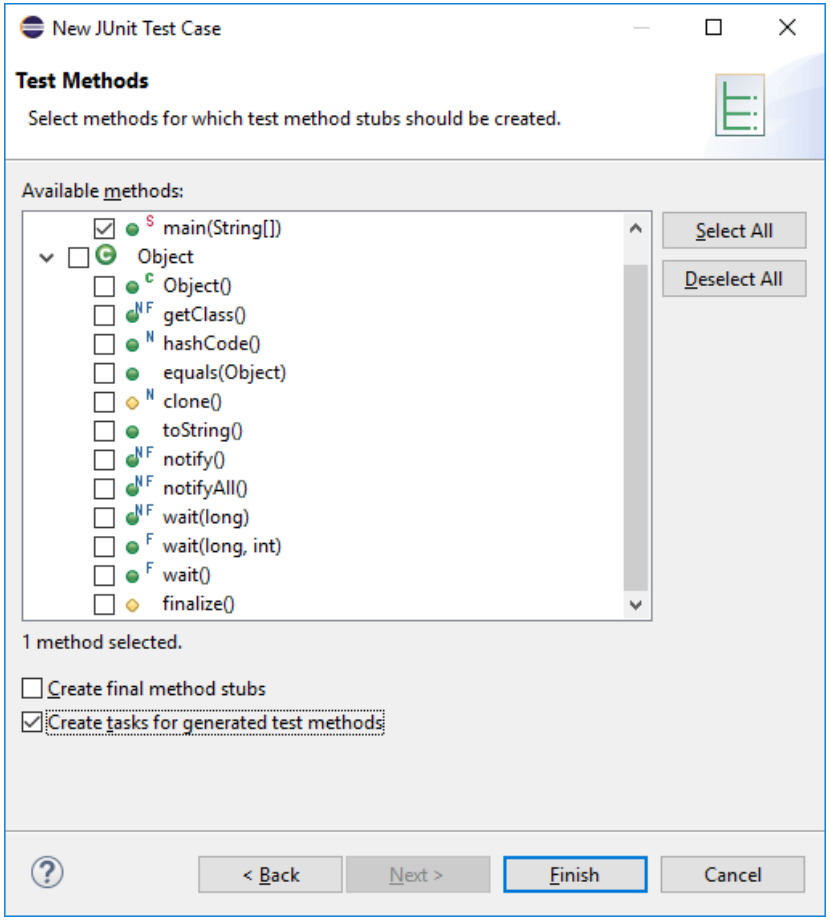
Creating a New Test Case (3)

- In the New JUnit test case window, ensure to select the **New Junit 4 Test**
- Select the **Package** (default packages are discouraged – see slide (4) below)
- Enter the **Name** (ending with “Test”)
- Click **Next** to select the **methods** for which your test cases are required (see slide (5) below) and then **Finish**

Creating a New Test Case (4) - Example



Creating a New Test Case (5) - Example



Your Demo for Part 2 – Create Example Test Case

- **Demonstrate** to your Professor (using your code) **how you will create** a completely new Test Case.
- Show your professor the eclipse **editor screen** of your new Test Case created and run.
- **Explain** the meaning of the **annotations** that you see in the Test Case:
  - **Hint:** @Before, @After, @Test and others.
  - **Explain:** *fail("Not yet implemented"); //TODO*

Javadoc

You are encouraged to insert Javadoc style comments into your code.

However, you are not required to generate it for this particular lab.

References

- Java How to Program, Early Objects Plus MyProgrammingLab with Pearson eText -- Access Card Package, 11/E. Author: Deitel ISBN: 9780134800271

- Big Java Early Objects, 7/E. Author: Horstmann, C. Wiley. ISBN: eText: 978-1-119-49909-1 or loose-leaf paper: 978-1-119-74020-9.