

EECS2030 (Section E) Summer 2026:
Advanced Object-Oriented Programming
Programming **PracticeTest0** (Version A)

Requirements of this Programming Test

- This programming test is **strictly** individual: plagiarism check may be performed and suspicious submissions will be reported to Lassonde for **a breach of academic honesty**.
- **Programming Requirements**
 1. You are **only allowed** to use primitive arrays (e.g., `int[]`, `String[]`, `Facility[]`) for implementing classes and methods to solve problems related lists/collections.
Any use of a Java library class or method is forbidden (that is, use selections and loops to build your solution from scratch instead):
 - Some examples of **forbidden** classes/methods: `Arrays` class (e.g., `Arrays.copyOf`), `System` class (e.g., `System.arraycopy`), `ArrayList` class, `String` class (e.g., `substring`), `Math` class.
 - The use of some library classes does not require an `import` statement, but these classes are **also forbidden** to be used.
 - Here are the exceptions (library methods which you **are allowed** to use if needed):
 - * `String` class (`equals`, `format`)You will receive a **30% penalty** if this requirement is violated.
 2. If your submitted project contains any compilation errors (i.e., syntax errors or type errors), TAs will attempt to fix them (if they are quick to fix); once the revised submission is graded, your submission will receive a **30% penalty** on the resulting marks (e.g., if the revised submission received 50 marks, then the final marks for your test would be 30 marks).
- The time limit is **strict** so you are solely responsible for leaving enough time (≈ 3 minutes) to export the completed Java project and upload/submit the archive (.zip) file to WebSubmit.
- Your submission will only be graded by:
 - JUnit tests given to you in the starter tests; and
 - Additional JUnit tests on input values not covered in the starter tests.
- Reminders of Some Java Syntax:
 - To compare primitive values (e.g., `int`, `double`, `char`, *etc.*), use the `==` relational operator.
 - To compare the contents of two strings `s1` and `s2`, write `s1.equals(s2)`.

It is recommended that,
for **every 20 minutes**,
you **export** your latest developed project and
submit it to WebSubmit.

See subsequent pages for instructions on importing,
exporting, and submitting your project.

1 Getting Started

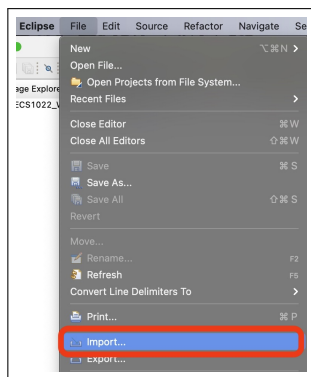
1.1 Step 1: Download and Import the Starter Project

1. Download the Eclipse Java project archive file **PracticeTest0_Starter.zip** from:
here and choose **Save** to save it. By default it will be downloaded to your home directory (e.g., `/eecs/home/ltstu`).

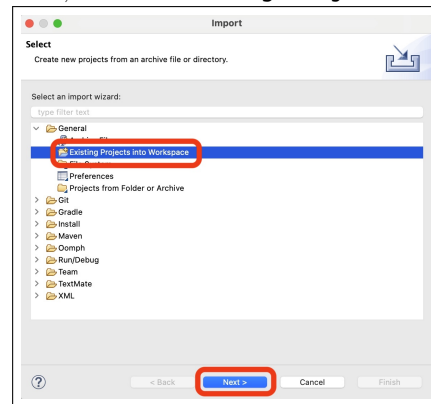
Note. Do not attempt to modify the URL of the download link: You will receive a zero if you attempt a version that's different from one suggested by the download link.

2. Launch Eclipse and browse to the **default workspace**
(which is something like: `/eecs/home/ltstu/eclipse-workspace`) there and click **ok**.
3. In Eclipse:

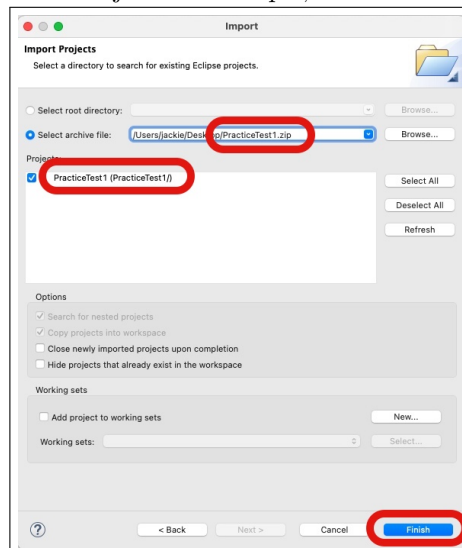
3.1 Choose File, then Import.



3.2 Under General, choose Existing Projects into Workspace.



- 3.3 Choose **Select archive file**. Make sure that the **PracticeTest0** box is checked under **Projects**. Then **Finish**. (diagram below is just for example; name should be **PracticeTest0.zip**)



1.2 Step 2: Programming Tasks

The format of this programming test is identical to that of your Lab1: given a JUnit test class containing compilation errors begin with, derive, declare, and implement classes and methods in the `model` package. You are **not** asked to build console applications for grading.

- The `junit_tests` package contains a collection of JUnit tests suggesting the required classes and methods.

Note. For this programming test, you will **also** be graded by an additional list of JUnit tests (e.g., you are given 5 tests, and there are another additional five tests not given, and your submission will be graded by all 10 tests).

Therefore, you are expected to test your program with extra inputs by writing more JUnit tests. You can always add a new test by copying, pasting, and modifying a test give to you.

- The `model` package is empty (to be added classes derived from the given JUnit tests).
- **To make progress on your development, read carefully both:**
 - The given starter tests; and
 - Comments written in the JUnit test file.

Your **expected workflow** should be:

1. **Step 1: Eliminate compilation errors.** Declare all the required classes and methods (returning default values if necessary), so that the project contains no compilation errors (i.e., no red crosses shown on the Eclipse editor).
2. **Step 2: Pass all unit tests.** Add **private** attributes and complete the method implementations accordingly, so that executing all tests result in a **green** bar.

If necessary, you are free to declare (private or public) helper methods.

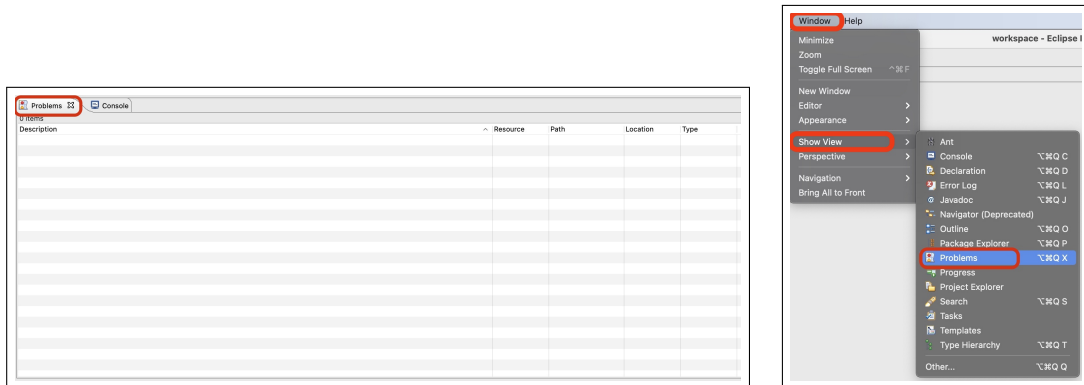
It is critical that you complete Step 1 first, so that you will not receive a penalty for submitting a project containing compilation errors.

How to Deal with a Failed JUnit Test? From the JUnit panel from Eclipse, click on the failed test, then **double click** on the first line underneath **Failure Trace**, then you can see the **expected value** versus the **return value** from your utility method.

2 Getting Ready for Submission

You are required to submit a Java project archive file (.zip) consisting all subfolders.

1. Before you submit, you must make sure that the **Problems** panel on your Eclipse shows **no errors** (warnings are acceptable). In case you do not see the **Problems** panel: click on **Window**, then **Show View**, then **Problems**.

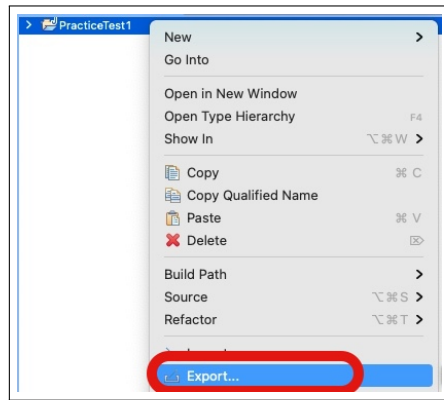


Submitting programs with errors (meaning that it cannot be run for grading) will result in possible partial, but low, marks.

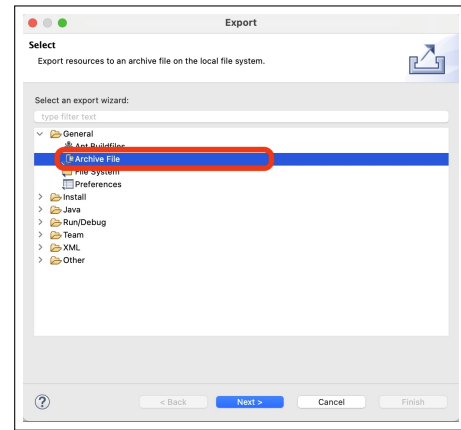
3 Submission

In Eclipse:

1. Right click on project **PracticeTest0**.
Then click **Export**



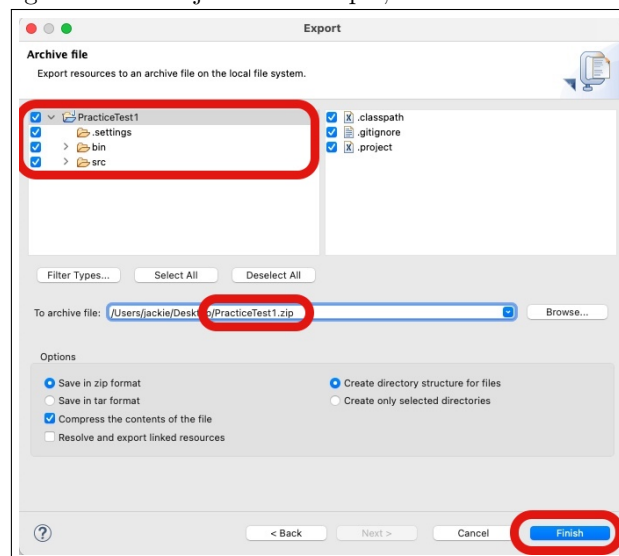
2. Under **General**, choose **Archive File**.



3. Check the top-level **PracticeTest0**

Make sure that all subfolders are checked: **.settings**, **bin**, and **src**.

Under **To archive file:** browse to, e.g., desktop, and save it as **PracticeTest0.zip** (**case-sensitive**)
Then **Finish**. (diagram below is just for example; name should be **PracticeTest0.zip**)



2. Go to the following WebSubmit link to upload & submit your developed project: **here**.
3. Attach the Java archive file: **PracticeTest0.zip**
 - You may **upload** & **submit** as many draft versions as you like before the deadline.
 - Only the last-submitted project archive file, if any, will be graded.
 - **Only a valid project archive (.zip) file will be accepted for grading.** For example, submitting individual Java files will only be graded with **penalty**.