EECS2101 (E) Summer 2025
Assignment 4
Programming with Generic Linked-Node Based Trees (Part II)

Adapted from Prof. Jackie Wang's Assignment 4 and Submission Instructions for EECS2101

Instructor: Sunila Akbar

Released: Sunday, July 20
Due Date: 11:59 pm, Sunday, Aug 3

- The format of this assignment closely reflects the previous Assignment 3. Please read the instructions carefully.
- Although group work is permitted, you are strongly encouraged to first attempt the assignment independently before seeking assistance.
- Follow the instructions to submit (via web submit) the required file (a Java project archive zip file).
- Emailing your submission to the instructor or TAs will **not** be acceptable, under any circumstance.
- Texts in blue are hyperlinks to the corresponding documents/recordings.

Policies

- **Please note that your solution to this assignment - whether submitted or not - remains the intellectual property of the EECS Department.** As such, we kindly ask that you do **not** share or distribute your code through any public platform (e.g., a non-private GitHub repository). Sharing solutions publicly may violate academic integrity policies, and the department reserves the right to take appropriate action if necessary.
- You are responsible for submitting your work electronically through the web submit system before the deadline. We strongly recommend that you **back up your work regularly** to prevent loss due to unexpected technical issues. To help you manage your projects securely, we encourage you to follow this tutorial series on setting up a **private** GitHub repository for your Java code.
- Please be aware that the deadline is **firm**. To ensure fairness for everyone, late submissions cannot be accepted. We recommend planning ahead and starting early to avoid last-minute issues.

Contents

Learning Outcomes

By completing the assigned exercises of this assignment, you are expected to be able to:

1. Implement and test algorithms on linked-node based trees, from scratch and without the aid of any other data structure of library class.
2. Use the debugger tool to identify and fix errors.

Assumptions

- You have already setup a Github account and stored work in a **private** repository: e.g., `EECS2101E-S25-workspace`.

  Note. Though not required, it is highly recommended that you adapt to the practice of backing your work using a versioning tool like Github.

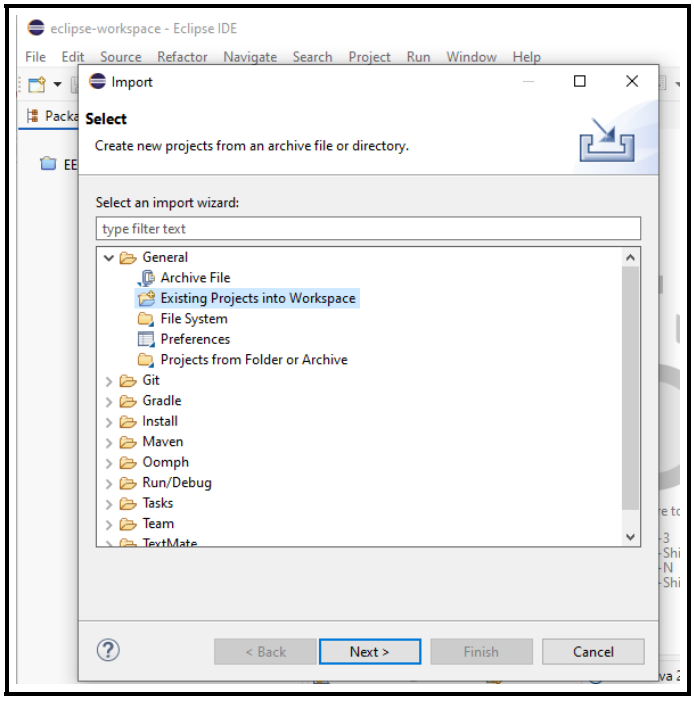
- You are able to use Eclipse to complete this assignment on either your own machine or the EECS remote labs.

  Note. The starter project was created using Eclipse and an Eclipse project archive file is expected to be submitted. Therefore, you may not want to use other IDE such as IntelliJ.

1 Task 1: Complete Programming Tasks

1.1 Step 1: Download and Import the Starter Project

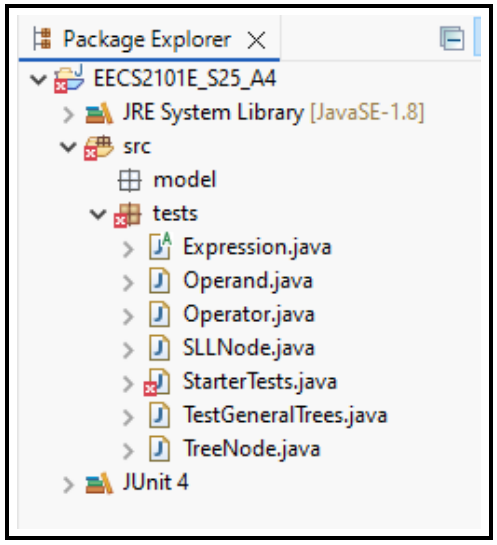
1. Download the Eclipse Java project archive file from eClass: `EECS2101E_S25_A4_Starter.zip`
2. Launch Eclipse and browse to, e.g., `EECS2101E-S25-workspace`, as the `Workspace` then click on `Launch`.
3. In Eclipse:
   - 3.1 Choose `File`, then `Import`.
   - 3.2 Under `General`, choose `Existing Projects into Workspace`.



   - 3.3 Choose `Select archive file`. Make sure that the `EECS2101E_S25_A4` box is checked under `Projects`.
   - 3.4 Then click `Finish`.

1.2 Step 2: Programming Tasks

From the `Package Explorer` of Eclipse, your imported project has the following structure:



- The `tests` package contains the `StarterTests` JUnit test class.
  - It is **expected** that the `StarterTests` JUnit class contains **compilation errors** to start with. This is because that declarations and definitions of the required class(es) and method(s) it references are missing.
  - Study carefully the test methods listed in this test class, as they suggest:
    - the required class(es) and method(s) to be implement in the 'model' package
    - how the required class(es) and method(s) should be implemented.

    You must not modify these given JUnit tests, as they suggest how the intended class(es) and method(s) should be declared.
  - Test methods included here are meant to get you started. **Therefore, you are expected to write additional tests to ensure that your submitted code is able to handle other input values implied by the problem specification (see the in-line comments in `StarterTests`).**
  - The `Node` class contains the complete implementation of a tempalte for nodes existing in a singly-linked list. When you implement the required class(es) and method(s) in the `model` package, you `must` use this version of the `Node` class. Do `not` modify this class. When your submission is graded, the same starter version of the `Node` class will be used, meaning that if you made any changes to this class, they would be wiped out and your submitted classes may just stop compiling.
- The `model` package is empty. Class(es) and method(s) derived from the given JUnit class **must** be added to this package. Class(es) added to a package other than `model` will **not** be graded.

  Therefore, your tasks are:

  1. Inferring from the given JUnit tests, add the missing class(es) and method(s) into the `model` package. For example, if you add class `Foo` in the model package, make sure that you write a line in the beginning of the `StarterTests` class (after the line `package tests;`):

     ```
     import model.Foo;
     ```

  2. Pass **all** JUnit tests given to you (i.e., a **green bar**).

     To run them, as shown in the Review Tutorial Series, right click on `StarterTests.java` and run it as JUnit tests. Of course, none of the given tests would pass to begin with.

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 implemented method. Furthermore, when needed, you should a **breakpoint** at the line of the failing assertion, then launch the **debugger** to pinpoint where the error came from.

1.3 Step 3: Exporting the Completed Project

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

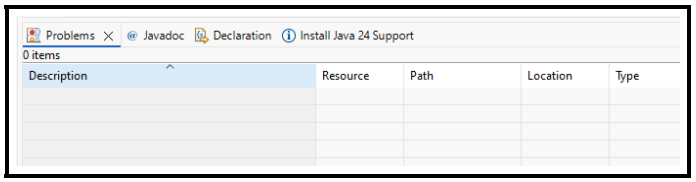
In Eclipse:

1. Right click on project `EECS2101E_S25_A4`. Then click `Export`.
2. Under `General`, choose `Archive File`.
3. Check the top-level `EECS2101E_S25_A4`. Make sure that all subfolders are checked: `.settings`, `bin`, and `src`.. Under `To archive file:` browse to, e.g., desktop, and save it as `EECS2101E_S25_A4.zip`.
4. Then click `Finish`.

   Note. In case you have concerns about exporting and submitting the **.setting** subfolder: it will be kept confidential and access-protected on eClass.

2 Submission

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.

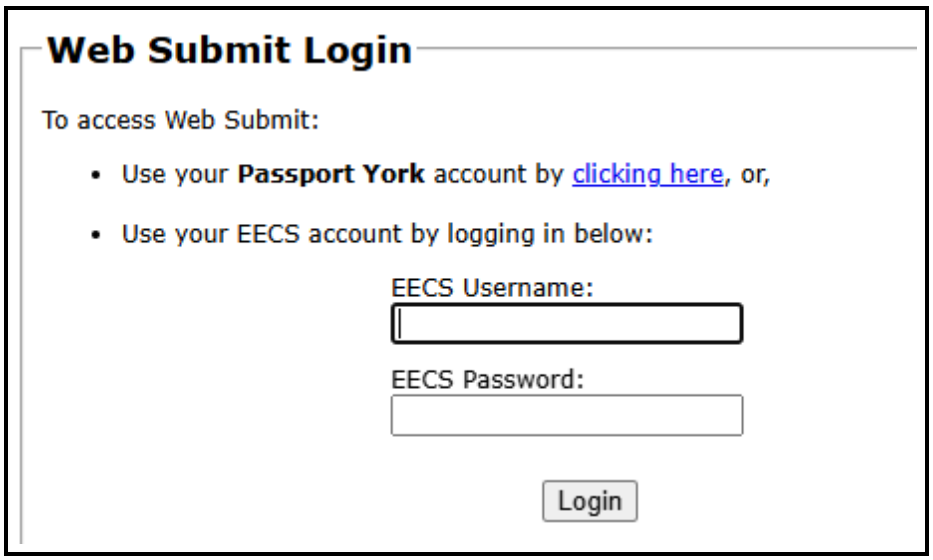
2. Section 1.3 asks you to **export** the Java project as an archive file:

`EECS2101E_S25_A4.zip`

Click on the following link (for which you will be prompted to enter your EECS account login credentials):

`https://webapp.eecs.yorku.ca/submit?acadyear=2024-25&term=S&course=2101E&assignment=a4`

- You **must** login into the web submit page using your EECS login credentials (otherwise, your submitted folder on the EECS server may not be identified properly):



Note. If you are prompted for your PPY login instead, then it might be due to an earlier login session. In this case, login first with your PPY account credentials, then log out. Then, clicking on the above submission link should lead you to the login page for EECS account credentials.

- Ensure that the correct academic year (`2024-25`), term (`S`), course (`2101E`), and assignment (`A4`) are chosen. Then, browse to the archive file `EECS2101E_S25_A4.zip` and click on `Submit Files`.
- You may upload as many draft versions as you like before the deadline – only the **latest** submitted version of your work before the deadline will be graded.
- It is your **responsibility** to **download and ensure** that the submitted zip file is the one you intend to be graded (e.g,. non-empty, not the starter project).

3 Amendments

Clarifications or corrections will be added to this section.

-