EECS2101 (E) Summer 2025
Assignment 3
Programming with Generic Linked-Node Based Trees

Adapted from Prof. Jackie Wang's Assignment 3 and Submission Instructions for EECS2101

Instructor: Sunila Akbar

Released: Saturday, June 21
Due Date: 11:59 pm, Sunday, July 13

- The format of this assignment closely reflects that of Assignment 4 and programming test. Please read the instructions carefully.
- Although group work is permitted, you are strongly encouraged to first attempt the assignment independently before seeking assistance.
- This assignment is designed to help you develop a solid understanding of Singly Linked Lists (SLL). To support this learning objective, please refrain from using:
  - Any library classes such as `ArrayList`, and
  - Primitive arrays such as `String[]`, `Integer[]`, or `int[]`

  in your implementation of the required class(es) and method(s). Using these constructs may compromise your learning experience and will result in a deduction of up to **50%** in your marks for the programming test.
- For this assignment you are provided with only a limited number of starter JUnit tests to help you begin. You are encouraged not only to use these starter tests, but also to supplement them with additional tests that evaluate your code across a wider range of input values. This practice will be particularly helpful in programming test, where additional test cases beyond the provided ones may be used for grading. For this assignment, make your best effort before the submission deadline. After submission, additional tests will be provided so that you can independently evaluate the robustness of your solution.
  - 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

1 Task 1: Background Studies **4**

2 Task 2: Complete Programming Tasks **5**
2.1 Step 1: Download and Import the Starter Project 5
2.2 Step 2: Programming Tasks 6
2.3 Hints on Tasks 7
2.4 Step 3: Exporting the Completed Project 8

3 Submission **9**

4 Amendments **10**

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.

Requirements of this Assignment

In the context of a **programming test**:

- During the programming test:
  - You will be required to work on the Eclipse IDE. Otherwise the exported project from another IDE may fail the required structure (by the grading program) and receive a penalty.
  - The objective of this assignment is to help you gain hands-on experience with singly-linked nodes by implementing the required algorithms *from scratch*, without using any external data structures or libraries.

    To ensure that you meet the learning goals of this assignment, please carefully adhere to the following restrictions:
    - Do **not** use primitive arrays (e.g., `String[]`, `Integer[]`, `int[]`).
    - Do **not** use any Java library classes such as:
      - `Arrays` class (e.g., `Arrays.copyOf`)
      - `System` class methods (e.g., `System.arrayCopy`)
      - `ArrayList` class
      - String methods such as `substring`
    - Some built-in classes may not require `import` statement, but these are still restricted unless explicitly allowed.
    - The following **exceptions** are permitted if needed: `equals`, `format`, and `length` in the `String` class.

    Note: Failure to follow these restrictions may result in a **50% deduction** on your marks in the subsequent programming tests.
  - Ensure that your submitted project compiles with the provided starter test class. If your code has syntax or type errors, TAs may attempt minor fixes (if straightforward), but a penalty will be applied.
  - The grading of your submission will **start** by automatically **unzipping** the submitted Java project archive file (**.zip**) and extracting the required class(es). It is therefore crucial for you to follow **precisely** the spelling of the archive file name. **Penalty** will be taken if the grading cannot proceed due to carelessness on following the instructions in Section 2.4.
  - You will be graded not only by the starter JUnit tests given to you, **but also additional tests** covering some other input values. This is to encourage you to take more responsibility for the correctness of your code, by writing your own tests.

  It is therefore your best interest of completing the assignment using the Eclipse IDE and practicing according to the above-mentioned expectations.

- For the JUnit test class `StarterTests.java` given to you:
  - Do **not** modify the test methods given to you.
  - If you wish to add new test methods, do so by creating another test class in the `tests` package.

- Derived from the given JUnit test methods:
  - Each class you introduce and implement must be placed under the `model` package.
  - Do **not** add any class that is not required by the starter tests: any such class will not be graded.
  - If considered necessary, you may declare additional attributes and/or helper methods if you wish.
  - For each method you implement:
    - No `System.out.println` statements should appear in it.
    - No Scanner operations (e.g., `input.nextInt()`) should appear in it.
      Instead, declare input parameters of the method as indicated by the JUnit tests.

1 Task 1: Background Studies

1. This assignment requires the coverage of singly-linked lists (SLLs) (e.g., Lectures 8 to 13 - [**eClass**]) and linked-node based general trees (e.g., Lectures 13 to 16 - [**eClass**])

2. You already worked with generics in Assignment 1: refer to its instructions PDF for quick reminders on the basic use of generics.

3. This assignment requires the more intermediate use of generics. **Study carefully the TestGeneralTrees class given to you (in the tests package)**, which contains how the following two classes work together:

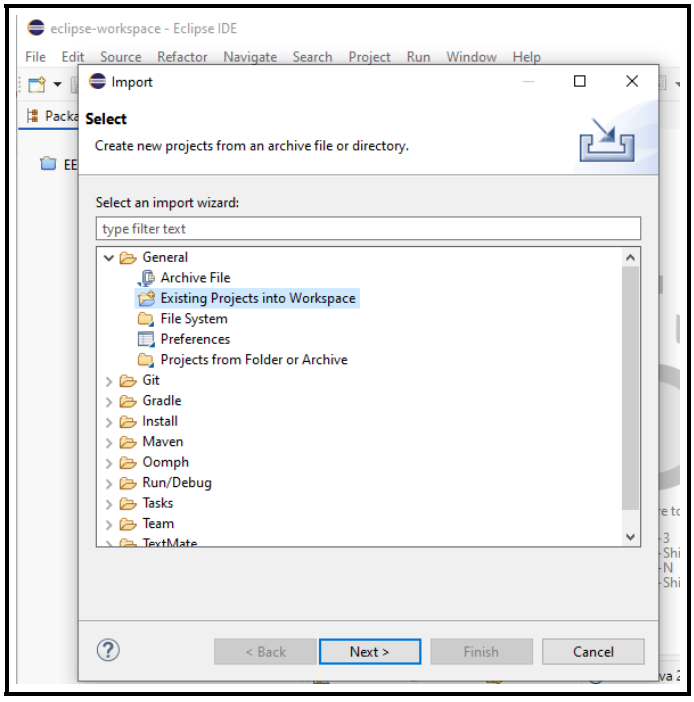
   - **`SLLNode<E>`**: This class functions as a standard singly linked list node, similar to the `Node` class used in Assignment 1.
   - **`TreeNode<E>`**: This class resembles the `TreeNode` class discussed in Lecture 14, but with an important distinction: its children are maintained as a chain of singly linked nodes rather than using primitive arrays. Note that **primitive arrays are not allowed** in this assignment.
   - In the `TestGeneralTrees` class, observe the following type declarations:
     - `TreeNode<String> n;` declares a tree node storing a string value. The element can be accessed using `n.getElement()`. Here, the generic type parameter `E` of `TreeNode<E>` is instantiated as `String`.
     - `TreeNode<Integer> n;` declares a tree node storing an integer value. The stored value can be retrieved using `n.getElement()`. In this case, the generic parameter `E` is instantiated as `Integer`.
     - `SLLNode<TreeNode<String>> tn;` declares a singly linked list node that stores a reference to a tree node, which itself stores a string. To access the string, we write `tn.getElement().getElement()`. Here, the outer generic parameter `E` of `SLLNode<E>` is instantiated by `TreeNode<String>`, and `TreeNode<E>` is instantiated by `String`.
     - `SLLNode<TreeNode<Integer>> tn;` declares a singly linked list node storing a reference to a tree node that contains an integer. The integer can be accessed using `tn.getElement().getElement()`. In this case, `SLLNode<E>` is instantiated with `TreeNode<Integer>`, and `TreeNode<E>` is instantiated with `Integer`.

2 Task 2: Complete Programming Tasks

Starting Task 2 should mean that you have ***already completed*** the background studies (if necessary) as outlined in Section 1.

2.1 Step 1: Download and Import the Starter Project

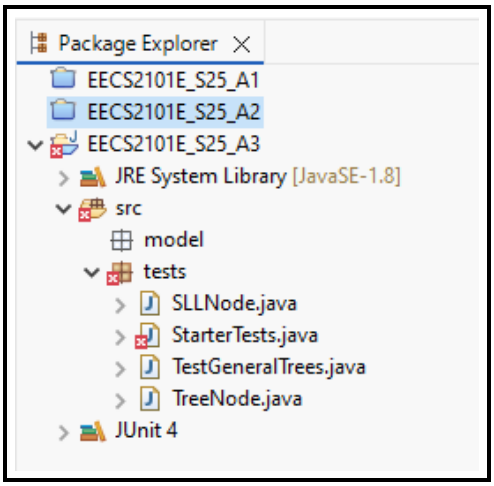
1. Download the Eclipse Java project archive file from eClass: `EECS2101E_S25_A3_Starter.zip`
2. Launch Eclipse and browse to, e.g., `EECS2101E-S25-workspace`, as the `Workspace` then click on `Launch`.
3. In Eclipse:
   - 3.1 Choose `File`, then `Import`.
   - 3.2 Under `General`, choose `Existing Projects into Workspace`.



   - 3.3 Choose `Select archive file`. Make sure that the `EECS2101E_S25_A3` box is checked under `Projects`.
   - 3.4 Then click `Finish`.

2.2 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;
     ```

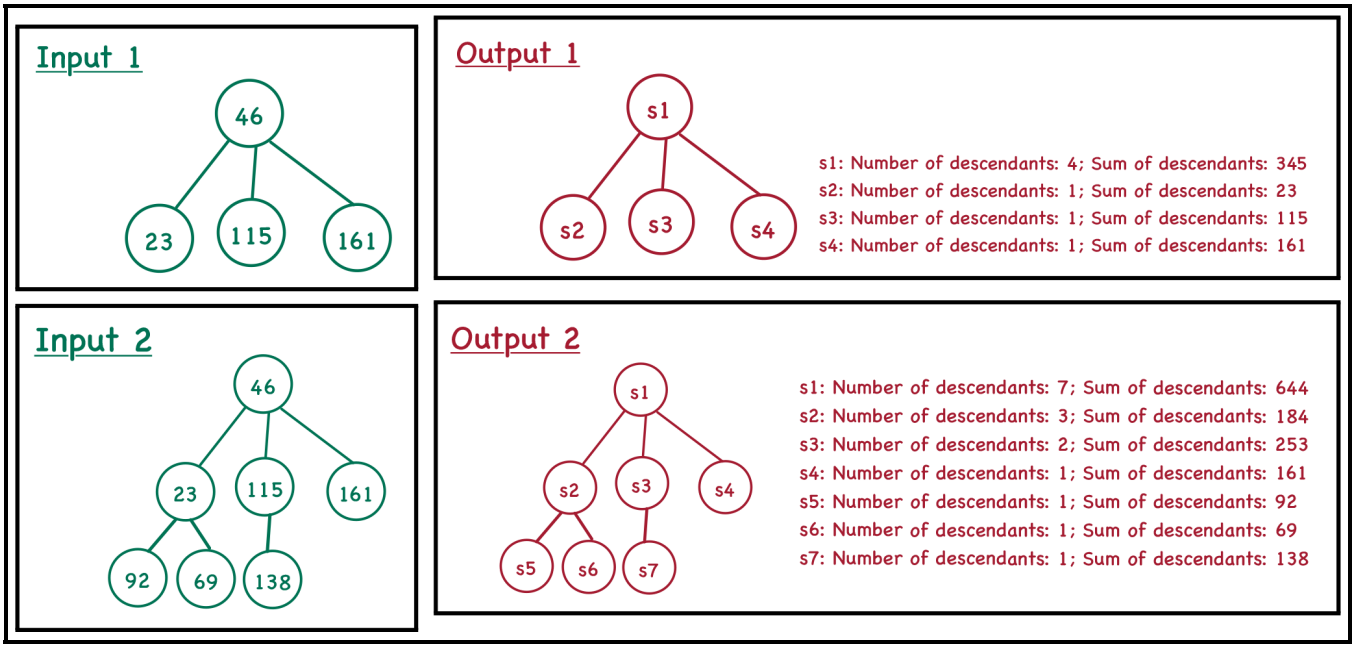
  2. Pass **all** JUnit tests given to you (i.e., a **green bar**).

     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.

2.3 Hints on Tasks

For the required method `getStats`, students are encouraged to visualize both the input and output tree structures—particularly for `test_getStats_1` and `test_getStats_3`. Similar code fragments may appear in future assessments. After attempting your own visualizations, you may compare your sketches with the model answers provided below.



2.4 Step 3: Exporting the Completed Project

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

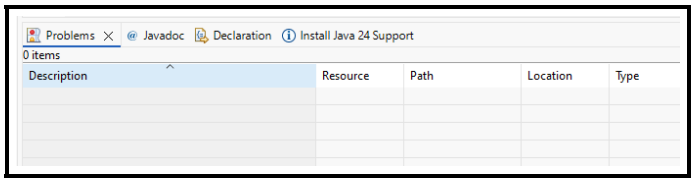
In Eclipse:

1. Right click on project `EECS2101E_S25_A3`. Then click `Export`.
2. Under `General`, choose `Archive File`.
3. Check the top-level `EECS2101E_S25_A3`. 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_A3.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.

3 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.

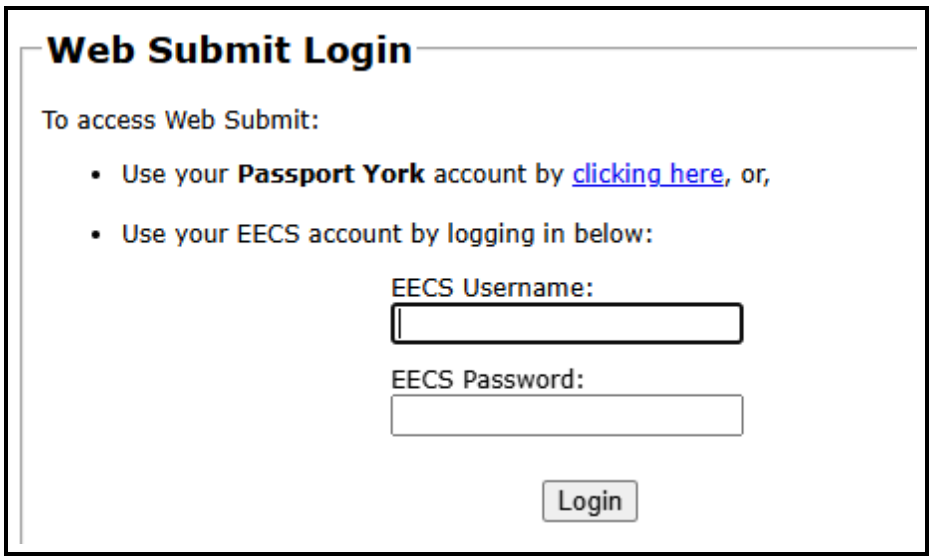
2. Section 2.4 asks you to **export** the Java project as an archive file:

`EECS2101E_S25_A3.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=a3`

- 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 (`A3`) are chosen. Then, browse to the archive file `EECS2101E_S25_A3.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).

4 Amendments

Clarifications or corrections will be added to this section.

-