EECS2011 (N & Z) Winter 2022
Guide to ProgTest2
When:
4:15 pm – 5:20 pm (EST), Monday, April 4
or
8:45 am – 9:50 am (EST), Tuesday, April 5

Chen-Wei Wang

- This programming test is **strictly** individual: plagiarism check will be performed on all submissions, and suspicious submissions will be reported to Lassonde for **a breach of academic honesty**.
- This programming test will account for **10%** of your course grade.
- This programming test is **purely** a programming test, assessing if you can write **valid** Java programs free of syntax, type, and logical errors.
- A starter project will be available for download on **eClass**, whereas the submission must be made through a **web submit** link (same as the assignments).

1 Policies

- **Timing Constraints**:
  - ProgTest2 will be:
    - ***opened*** for submission at **04:15 pm EST** on **Monday**, April 4; and
    - ***closed*** for submission at **05:20 pm EST** on the same day.
  - ProgTest2 will again be:
    - ***opened*** for submission at **08:45 am EST** on **Tuesday**, April 5; and
    - ***closed*** for submission at **09:50 am EST** on the same day.
  - For the above two submission periods, you are only allowed to attempt the test once.
  - There is **not** a 24-hours submission period: if you started your attempt later than the above-mentioned time where the submission is opened, then you would have less than 65 minutes available to complete the test. For example, if you started your attempt at 04:20 pm on Monday, the test's submission will still be closed at 05:20 pm, meaning that you have 60 minutes remaining to complete the test.

– **Programming Requirements**

1. You are **not allowed** to use primitive arrays (e.g., `int[]`, `String[]`) for implementing classes and methods to solve problems.

   Also, 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 ***50% 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 ***20% penalty*** on the resulting marks (e.g., if the revised submission received 50 marks, then the final marks for your test would be 40 marks).

– **Format**

The format of this programming test will be **identical** to that of your Assignment 1: given a JUnit test class containing compilation errors to begin with, derive, declare, and implement classes and methods in the `model` package.

- The `model` package is empty (to be added classes derived from the given JUnit tests).
- The `tests` package contains JUnit tests suggesting the required classes and methods.

– **Grading**

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

Therefore, it is up to you 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.

– **Submission for Grading**:

- Like your assignments, submission for this programming test (of an Eclipse Java archive `.zip` file) must be through the supplied **web submit** link (made available during the test).
- It is your sole responsibility for making sure that the correct version of project archive file is submitted. **After** uploading the submission, you should **re-download** the archive file and make sure it is the right version to be graded. **No** excuses or submissions will be accepted after your attempt times out.
- Email attachments may **only** be accepted:
  * if it is with a reason judged as valid by your instructor (e.g., **running out of time is not a valid reason** as you should have allocated enough time to complete the submission); **and**
  * if it is sent within ***5 minutes*** after the test end time.
- When accepted, there will be a ***15% penalty***.

2 Rationales: Grading Standard & Time Constraint

At this point of your degree, after studying three programming courses in Java, it should be reasonable to expect that when you implement classes and methods:

- They compile (i.e., free from syntax or type errors).
- They are correct:
  - The starter tests given to you are basic and meant only as a starting point.
  - It is expected that your solutions are programmed not just specific to input values supplied in the starter tests. Instead, they should cover scenarios that may be missing from the starter tests, but implied by the problem specification given to you as in-line comments in the starter tests.

  Passing only the given starter tests, but failed many or all of the additional tests, is a clear indication that your code: **1)** works only for those specific input values in the starter tests; and/or **2)** does not solve the problem completely, by failing those use cases that are consistent with the given problem descriptions. Consequently, the resulting mark may **not** be passing.

Working under stress is unavoidable. Your future programming interviews for jobs will expect you to do the same: given problems, program your solutions in front of a work station or a whiteboard within some (short) set time limit. More critically, after landing a job, whenever being called upon by your perspective workplace supervisor for some customer-reported bugs, most likely they need to be fixed within a short time interval. Arguably, not being able to perform well under stress can be a indication of a lack of **enough practice** (for which your instructor's supplied examples may **not** be sufficient for your specific needs, in which case it is **your responsibility** to go above and beyond to seek advice and find extra examples), which is surely unpleasant at first but also suggests how you can improve your skills.

3 How the Test Should be Tackled

- 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). See Steps 1.1 to 1.3 of Section 2.2 in the written notes ***Inferring Classes from JUnit Tests***.
  2. **Step 2: Pass all unit tests.** Add **private** attributes (if applicable) and local local variables to complete the method implementations, 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.***

4 Coverage for the Test

1. Singly-Linked and Doubly-Linked Lists [Parts C – K, Lecture 2]
2. General Tree [Parts B1 – B4, Lecture 5a]

   There will **not** be any written questions, but you may review the relevant lecture materials to clarify the concepts.

5 Study Tips for the Test

- Finish the given example test (for as many times as needed) to **familiarize yourself with the workflow of the test**.
- Review examples covered in the tutorial videos, lectures, and written notes. Make modifications to the example test accordingly by adding more methods and tests.

6 Example Test

- By the end of **Tuesday, March 29**:
  - A solution to your **Assignment 2** (covering general trees) will be made available .
  - A simple example test (based on the lecture and Q&A examples) will be made available under the **Programming Tests** section on the ***N & Z eClass site***. You can attempt this test for as many times as you wish. However, it is important to note that the example test is:
    - meant for familiarizing yourself with the **format** and **workflow** of the test;
    - **not** meant to cover **all** topics required by the actual test (you are expected to study **all** materials as listed in Section 4); and
    - on the easier side (in the actual test, there will be harder questions testing your understanding of the materials).
- You are encouraged to attempt exercises in sites such as LeetCode: `https://leetcode.com/tag/linked-list/` and `https://leetcode.com/tag/tree/`.