EECS2030 (B & E) Fall 2021
Guide to Programming Test 2
When: Thursday (Nov 4) & Friday (Nov 5)

Chen-Wei Wang

1 Policies

– 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.

– **Timing Constraints**:

- Programming Test 2 will be *opened* at **02:00pm EST** on **Thursday**, November 4.
- Programming Test 2 will be *closed* at **02:00pm EST**, on **Friday**, November 5.
- During the 24-hours submission period, there is a **single attempt of 90 minutes** for you to complete the test.
  * Once you click on the test link and choose to start it, a timer of **90 minutes** (**including** the time for you to download and import the starter project, as well as for you to export the upload the completed project to **eClass** for grading) will start.
  * The time limit is **strict**: as soon as the timer **expires**, eClass will **disable** the submission. Therefore, you are **solely responsible** for leaving enough time ($\approx$ 10 minutes) to export the completed Java project and upload/submit the archive (.zip) file to eClass.

– **Submission for Grading**:

- Unlike your labs, submission (of an Eclipse Java archive `.zip` file) for this programming test must be through the ***B & E eClass site***.
- It is your sole responsibility for making sure that the correct version of project archive file is submitted. **Before** you click on the **`submit`** button on eClass, 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 out of the given 90 minutes to complete the submission); **and**
  * if it is sent within ***5 minutes*** after your attempt end time (as logged by eClass).
- When accepted, there will be a ***15% penalty***.

- **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).

2 Format

The format of this programming test will be **identical** to that of your labs and Programming Test 1: given a JUnit test class containing compilation errors begin with, derive, declare, and implement classes and methods in the `model` package. You will **not** be asked to build console applications for grading.

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

3 Grading

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, 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.

4 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 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.*

5 Rationales: Grading Standard & Time Constraint

The two most important learning outcome of this course are:

1. Computational thinking (for which you build through labs and assessed by written tests and the exam)
2. Being able to write *runnable* programs (for which you are assessed through computer tests)

When you write an essay, if there are grammatical mistakes, it can still be interpreted by a human. Computer programs are unlike essays: when your program contains compile-time syntax or type errors, it just cannot be run, end of story. When a computer program cannot be run, its runtime behaviour is simply unknown; and this is particularly the case when your program contains if-statements and loops.

When you land a job upon graduation, you would not expect your supervisor or colleagues to read your code that does not run, because it does not even compile, would you? True, you're still learning. But it is exactly this mind set that restricts your potential of becoming a competent programmer. This is already your third programming course. If we want to train you to be a competent programmer, NOW is the time to enforce the strict (but justifiable) standard.

Why is the **time constraint**? 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, which is surely unpleasant at first but also suggests how you can improve your skills fundamentally.

6 Coverage for the Test

1. Basic Object Oriented Programming

   Review Tutorial Series: Part 1 and Part 2

2. Exceptions [Lectures 2a and 2b]
3. `equals` method [Lecture 3]
4. Copy Constructors (for achieving *composition*) [Lecture 4]

   Notes.

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

   – The concepts about Github, remote labs, and terminal commands are **not** covered in the test.

7 Study Tips for the Test

– The actual test will require methods to be implemented with object creations, method calls, and loops (possibly embedded with selections).

– 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. For example, some methods in the actual test will be at the similar difficulty level as the `getPointsInQuadrantI` method as explained in the required written notes.

8 Example Test

- Solutions to your **Lab2** (covering exceptions) and **Lab3** (covering the `equals` method and copy constructors) will be made available by the end of **Monday, November 1**.
- An example test is made available under the **Programming Tests** section on the ***B & E eClass site***. You can attempt this test for as many times as you wish.

  This example test will be **closed** for submissions shortly before the actual test starts (i.e., 13:55 EST on Thursday, November 4).
- It is important to note that these questions are meant for familiarizing yourself with the **format** and **workflow** of the test, and they represent **only** as an example: you are expected to study **all** materials as listed in Section 6.
- The **level of difficulty** of the actual test will be somewhere between the example test and your Lab2 and Lab3.