EECS1021 Winter 2019
Guide to Lab Test 2
When: Week of March 4

Chen-Wei Wang

In this lab test you will be required to code Java methods in Eclipse. A class declared with methods will be given to you, and you only need to worry about writing lines of code that serve as bodies of implementations of these methods.

- This in-lab programming test accounts for 10% of your course grade.
- This lab test is **purely** a programming test, assessing if you can write **valid** Java programs free of syntax and type errors.
- **Stringent Requirements**:
  - It is absolutely critical that you submit Java code that compiles (i.e., no red crosses shown on the Eclipse editor). You receive a **zero** for the test if your code contains any compilation errors (i.e., syntax errors or type errors).
  - You are **only allowed** to use primitive arrays (e.g., `int[]`, `String[]`, *etc.*) to store user-input values. Any use of the Java collection library (e.g., `ArrayList`, `LinkedList`, *etc.*) will result an **immediate zero** of this lab test.

1 Rationale for the Grading Standard

The two most important learning outcome of this course are:

1. Computational thinking (for which you build through lab exercises and assessed by quizzes and 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, this is only your second programming course, and you're still learning. But it is exactly this mind set that restricts your potential of becoming a competent programmer. If we want to train you to be a competent programmer, NOW is the time to enforce the strict (but justifiable) standard. This is also why I make your Lab Test 1 weight less than others, so that if you really can't make it, it would not cause you to fail and you still have a good chance to obtain a decent grade, as long as you take proper action (e.g., more time spent on practicing to program) to strengthen your programming skills.

2 Critical Tutorial Video to Study

- The way you write program in the lab test is different from how you write console applications for lab exercises.
- Study this tutorial video (same as for Lab Test 1) carefully to make sure you are clear about what is to be expected to be written in a lab test:

  `https://www.youtube.com/watch?v=itnLsJ37dmc&list=PL5dxAmCmjv_7EZ6ITlo987tajyscAFnj8&index=3&t=0s`
- Once you have studied this tutorial video, complete the test `ExampleTest2`.

3 Rules

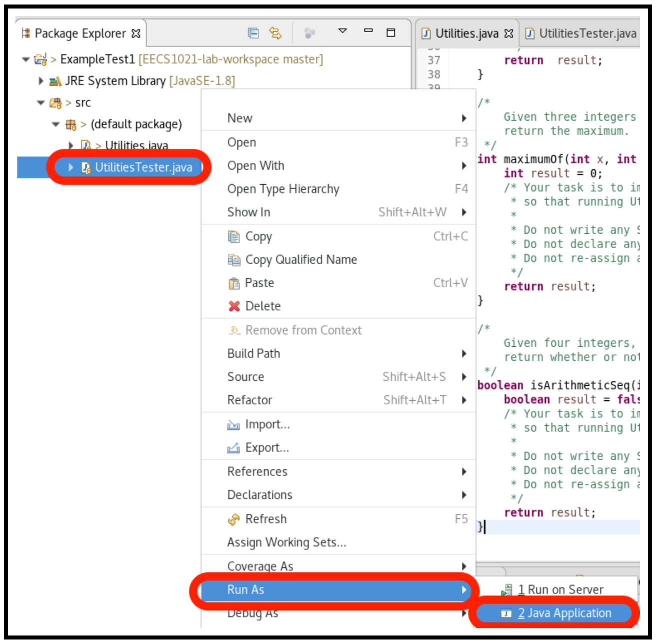
- The test takes place **in the beginning** of your registered lab session.
- You will be given **90 minutes** to complete the test.
- During the test, you are **required** to use Eclipse (but not any other programming tools) to develop your Java code.
- You are **allowed** to use a piece of paper (**blank** on both sides) for sketching your ideas. No sketch papers will be provided to you.
- You must show up for your **registered session only**.

  If you are not registered yet, you can only attend the Tuesday evening session (5pm) at WSC108.
- Arrive promptly, or you will be rejected to take the test.
- Wait outside WSC106/WSC108 when you arrive. The TAs will let you enter the room once the rooms are set up properly.
- Bring a piece of photo ID.
- No calculator will be allowed
- Put your mobile phone in your bag, and put your bag underneath the table.
- No data sheet will be allowed.

4 Format

- You will be given starter code which consists of two classes `Utilities` and `UtilitiesTester` that compile:
  - The `Utilities` class contains a list of methods, each of which with its body of implementation returning something useless. You are forbidden to change the input and output types of these methods.
  - The `UtilitiesTester` class is a user that calls methods of the `Utilities` class.

    To execute this tester class, in your Eclipse, go to the `Package Explorer` panel on the left, go under `ExampleTest2`, then right-click on `UtilitiesTester` and select `Run as Java Application`.



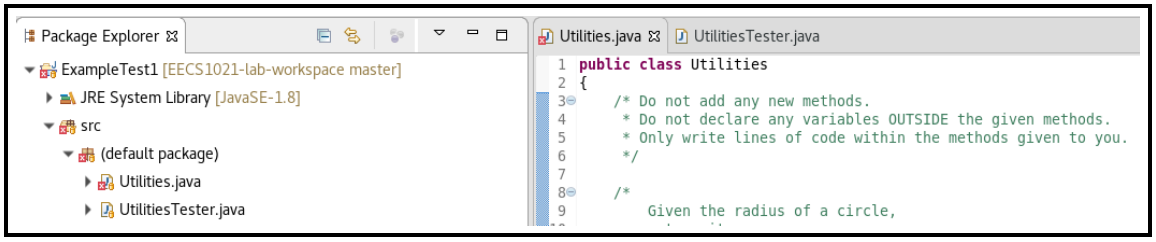
```
    return  result;
}

/*
    Given three integers
    return the maximum.
 */
int maximumOf(int x, int
    int result = 0;
    /* Your task is to in
     * so that running Ut
     *
     * Do not write any S
     * Do not declare any
     * Do not re-assign a
     */
    return result;
}

/*
    Given four integers,
    return whether or not
 */
boolean isArithmeticSeq(i
    boolean result = fals
    /* Your task is to in
     * so that running Ut
     *
     * Do not write any S
     * Do not declare any
     * Do not re-assign a
     */
    return result;
}
```

Running `UtilitiesTester` will execute methods in the `Utilities` class and display output on the console in Eclipse.

- Your tasks are to fill in, for each method in the `Utilities` class, lines of code in valid Java syntax as its body of implementation, such that running `UtlitiesTester.main()` will produce the identical output as expected. Expected output is specified in the `ExpectedOutput.txt` file in the Java project you are given.

– As you fill in bodies of implementations of methods in the `Utilities` class, you must not introduce any compile-time syntax or type errors.

An advice for you is to **pay close attention to the tags of these two classes: as soon as you see any of these two tags has a red cross**, e.g.,



```
public class Utilities
{
    /* Do not add any new methods.
     * Do not declare any variables OUTSIDE the given methods.
     * Only write lines of code within the methods given to you.
     */

    /*
        Given the radius of a circle,
```

then your priority is not to continue your development of code, but to fix the compile-time errors.

5 What You Submit at the End of the Test

– You will only need to submit the `Utilities.java` file:

- If this class does not compile due to any syntax or type errors, you receive **zero** marks for the test. There will be **no partial marks** rewarded to a class that does not even compile.

  This requirement will be imposed strictly, so your best preparation strategy is to familiarize yourself with all basic syntax of Java that's so far covered in the lectures and used in your labs.
- If your submitted class compiles, then you already receive 10 marks (out of 100) of the test.
- To determine the remaining 90% of your marks, we will run a number of test cases on your submitted class.

  For example, say we run 10 test cases (and say they happen to have equal weights) on your submitted code, and your submitted code complies and passes 6 of them, then your final marks are: $10 + 90 \times \frac{6}{10} = 64$.

6 Coverage

– All lecture materials (slides, recordings, and example codes) on loops and arrays.

7 Example Test

Click on here to download a set of practice questions.

8 Solution to Example Test

Click on here to see tutorial solutions to the practice questions. These solutions were completed on a different programming environment (Android Studio), but all Java programs developed there will work perfectly on Eclipse. You are advised to first attempt all problems.