EECS1022 (sec. A) Fall 2022
Lab 8 – Classes & ArrayLists

May Haidar

Due Date: 23:59 EST, Sunday, December 4, 2022

- Your lab assignment is **not** graded during the weekly scheduled lab sessions.
- Follow the instructions to submit (via eClass) the required file(s) for grading. Emailing your solutions to the instructor or TAs will **not** be accepted.

Policies

- **Your (submitted or un-submitted) solution to this lab exercise (which is not revealed to the public) remains the property of the EECS department. Do not distribute or share your code in any public media (e.g., a non-private Github repository) in any way, shape, or form. The department reserves the right to take necessary actions upon found violations of this policy.**
- When you submit your lab, you claim that it is **solely** your work. Therefore, it is considered as **a violation of academic integrity** if you copy or share **any** parts of your Java code during **any** stages of your development. Having said that, you may discuss with your colleagues problem solving techniques to address the questions in this lab.
- When assessing your submission, the instructor and TA may examine your code, and suspicious submissions will be reported to the department if necessary. **We do not tolerate academic dishonesty**, so please obey this policy strictly.
- You are entirely responsible for making your submission to the TA in time. The deadline is **strict** with no excuses. Refer to the course syllabus for the policy on a late submission.

Learning Outcomes

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

- In the Eclipse IDE (Integrated Development Environment):
  - Import a starter project archive file.
  - Given a computational problem, develop a Java solution (i.e., a utility method) composed of:
    - Classes and Objects
    - Java APIs and Arraylists
    - Using complex nested for loops
    - Using complex nested while loops
  - Run a Java class with the main method as a console Java application.
  - Use the given JUnit tests (calling the utility methods) to guide the development.
  - Export an existing project as an archive file.
- Understand the separation of concerns: model, console, and junit tests.

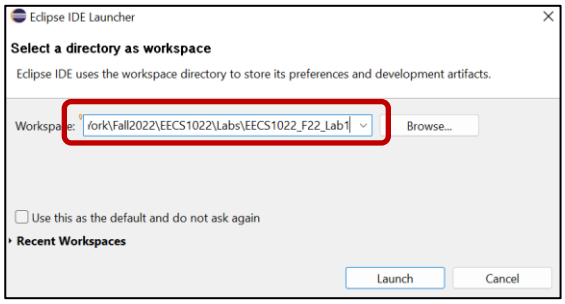
Lab Requirements

You will be graded not only by 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. Also, your lab submission will **not** be graded manually using the console application given to you.

- For the JUnit test class Lab8Test.java given to you:
  - Do **not** modify the test methods given to you.
  - You are allowed to add new test methods.
- For each method in the Lab8 class that you are assigned to implement:
  - No System.out.println statements should appear in each of the utility method.
  - No Scanner operations (e.g., input.nextInt()) should appear in each of the utility method. Instead, refer to the input parameters of the method.

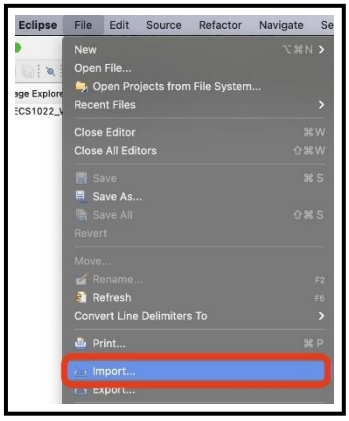
1. Download and Import the Starter Project

1. Download the Eclipse Java project archive file from eClass: EECS1022_F22_Lab8.zip
2. Launch Eclipse and browse to EECS1022-F22-workspace (for instance) as the Workspace. This should be where the EECS1022_F22_Lab8.zip is stored. Then click on Launch.

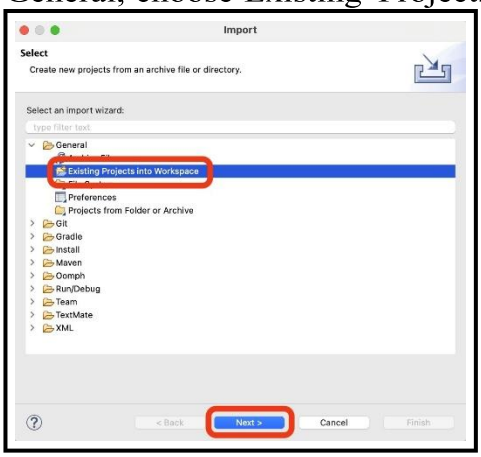


3. In Eclipse:

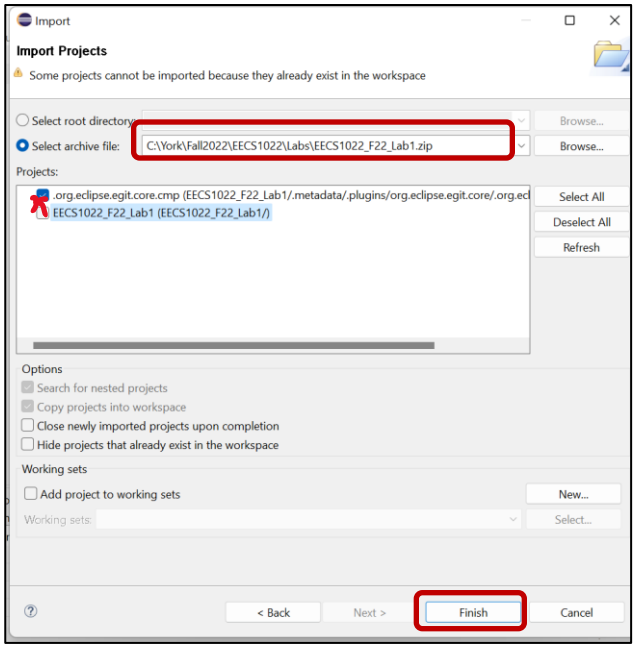
3.1 Choose File, then



3.2 Under General, choose Existing Projects into Workspaces



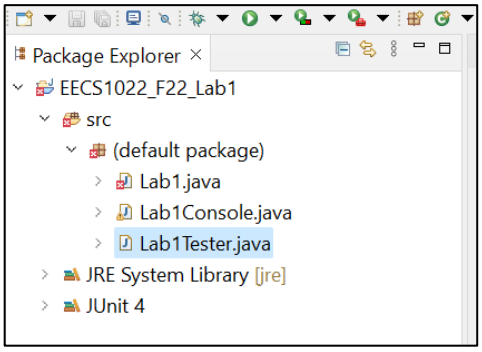
3.3 Choose Select archive file. Make sure that the EECS1022_F22_Lab8 box is checked under Projects. Then Finish.

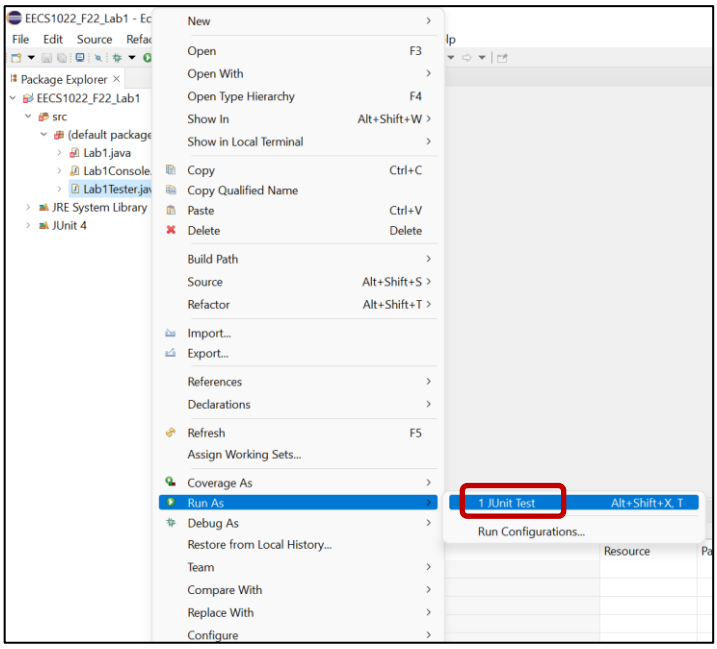


2. Programming Guidelines

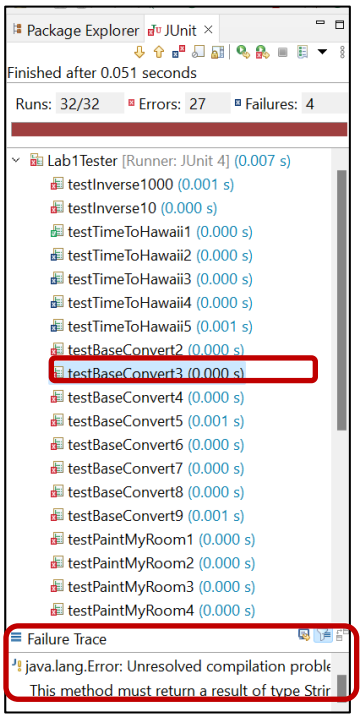
From the Package Explorer of Eclipse, your imported project has the following structure (note that the pictures correspond to Lab1 illustrations).

- You can manually test the assigned methods using the console class given to you *Lab8Console.java*. You may test all the methods in the same class. To test each method alone, you may want to comment out the rest of the testing code for the rest of the methods.
- Your goal is to pass **all** JUnit tests given to you (i.e., a **green bar**). To run them, right click on *Lab8Tester.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 utility method.



3. Programming Tasks

3.1 Task 1: switchPairs

Write a method called **switchPairs** that takes as a parameter an Arraylist of strings. The method switches the order of values in an ArrayList of strings in a pairwise fashion. Your method should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on. If there are an odd number of values in the list, the final element is not moved.

For example, if a list stores [to, be, or, not, to, be, hamlet], you should modify it to store [be, to, not, or, be, to, hamlet].

Call	ArrayList Value
`switchPairs(["to","be","or","not","to", "be", "hamlet"])`	`["be","to","not","or","be","to", "hamlet"]`
`switchPairs(["Yes","No"])`	`["No","Yes"])`

3.2 Task 2: acronym

Write a method called **acronym** that takes an ArrayList of strings as a parameter and returns an acronym made by combining the capitalized first letter of each word in the list.

For example, the list [laughing, out, loud] produces the acronym "LOL".

Call	Retun Value
`acronym(["see", "You", "later"])`	`SYL`
`acronym("laughing", "out", "loud"])`	`LOL`
`acronym("Talk","to","YOU","later"])`	`TTYL`

3.3 Task 3: Patient-Physician Classes

In this task you are going to write a solution to the task of Lab 7 using Arraylists instead of arrays. With a few changes. You will experiment with code reuse of existing API methods of the Arraylist.

Your first task is to create a class called **Patient**. A patient has a name, social security number, age, gender, and address.

The class should have as **constructors**:

- **Patient ():** constructor that sets the attributes to their default values.
- **Patient (String name, String ssn, int age, char gender, String address):** constructor to initialize the attributes of a patient object with the parameter values passed to the constructor.

Public Methods:

- **Setters** and **getters** for all the attributes
- **String toString ():** returns the string representation of a Patient object as follows:
  "Patient Name: name \t SSN: ssn \t Age: age \t Gender: gender \t Address: address"

You also have to create a class **Physician**. a name, phone, and an arraylist of patients. Also, the class keeps track of the number of patients of each physician.

The class should have two **constructors**:

- **Physician ():** the default constructor with no parameters.

- **Physician (String name, String phone):** constructor to initialize the attributes name and phone of a Physician with the parameter values passed to the constructor.

Public Methods:

- **Setters** and **getters** for all the attributes
- **admitPatient (Patient patient):** this method adds a patient to the physician's list. If a patient record exists in the physician's list, the new patient is not added to the list. You can check if a patient is in the list by checking if a patient with the same SSN exists in the list.
- **releasePatient (Patient patient):** this method takes a patient object and removes this object from the list of patients of the physician. If no patient in the array matches the patient parameter, the operation should be simply ignored.
- **patientsReport ():** this method reports all the patients of the physician. It returns a string that includes the patients information of the arraylist patients. Note that you **do not need** to explicitly build the string. Use the available methods in the ArrayList class:
  "[Patient Name: name \t SSN: ssn \t Age: age \t Gender: gender \t Address: address, "
  "Patient Name: name \t SSN: ssn \t Age: age \t Gender: gender \t Address: address, "
  "Patient Name: name \t SSN: ssn \t Age: age \t Gender: gender \t Address: address]"

Some of the details about the classes attributes, methods, and parameters should be deduced from the test cases written for you to test the classes.

4. Submission

You are required to submit ONLY the java file Lab8.java.

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 a mark 0.
2. Go to the eClass site: https://eclass.yorku.ca/course/view.php?id=64388
3. Under the Practice section, click on Lab5 to submit the **Java file ONLY**: **Lab8.java**
   - You may **upload** as many draft versions as you like before the deadline.
   - You must explicitly **submit** the draft version for grading before the deadline.