EECS1022 (M,N,O) Winter 2021
Lab6 (OOP Part 2)
Classes, Reference-Typed Attributes, Methods, JUnit Tests

Chen-Wei Wang

Release Date: Friday, February 26
Due Date: 23:59 EST, Friday, March 12

- The format of this lab (which will be identical to that of your Programming Test 3) is fundamentally different from that of Lab1 to Lab4. Read the instructions carefully.
- You are required to **work on your own** for this lab. **No** group partners are allowed.

  Plagiarism checks will be run on all submissions, and suspiciously similar ones will be reported to Lassonde.
- To complete this lab, it is **strictly forbidden** for you to use any library class (e.g., `ArrayList`). Violating this requirement will cause a **50% penalty** on your lab marks.
- For this lab, 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.
- 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 acceptable.
- Texts in blue are hyperlinks to the corresponding documents/recordings.

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.
- 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. Back up your work **periodically**, so as to minimize the damage should any sort of computer failures occur. Follow this tutorial series on setting up a **private** Github repository for your Java projects.
- The deadline is **strict** with no excuses. Refer to the course syllabus for the policy on a late submission.

Contents

Learning Outcomes

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

1. Exercise a simple workflow of Github.
2. In the Eclipse IDE (Integrated Development Environment):
   - Import a starter project archive file.
   - Given a computational problem, develop a Java solution composed of:
     - Numerical Literals and operators
     - String Literals and operators
     - Variables and assignments
     - (Nested) Selections/Conditionals/If-Statements
     - OOP Basics: Classes, Attributes, Constructors, Accessor and Mutator Methods, Method Invocations, Context Objects, Dot Notation
     - Declaring and manipulating (single-valued vs. multi-valued) reference-typed attributes
     - Inferring Java Classes from JUnit Tests
   - Use the given JUnit tests to guide the development.
   - Use the **debugger** features (step over/into/out/return) to find defects in programs.
   - Export an existing project as an archive file.

Assumptions

- You have already setup a Github account and stored work in a **private** repository: `EECS1022-W21-workspace`.

  Note. You only submit your lab through eClass, not Github. 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 the tutorial videos and this lab 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 Lab

- For this lab, 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.
- To complete this lab, it is **strictly forbidden** for you to use any library class (e.g., `ArrayList`). Violating this requirement will cause a **50% penalty** on your lab marks.
- The grading of your lab 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 2.5.
- Your lab submission will **only** be graded using JUnit (for this lab, the tests supplied to you plus some additional tests).
- For the JUnit test class `TestOnlineSchool.java` given to you:
  - Do **not** modify the test methods given to you.
  - You are allowed to add new nest methods.
- For each method which you are required to implement, derived from the JUnit test methods:
  - 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.
- You are welcome to ask questions related to this lab or the assigned tutorial videos on the forum. However, please be **cautious**:
  - You can help your fellow students understand the requirements of tasks.
  - **Do not share the code you developed to ask, or to answer, questions.**
    - Questions specific to the code you write would be best and most effectively addressed by TAs (during scheduled labs) or your instructor (during office hours or appointments).
    - Week 2's tutorial videos (Parts C to E) introduce to you **debugger in Eclipse**. You are encouraged to **set breakpoints and launch the debugger** when you are stuck at your own program.
  - **Hints** on how the solution should look like are **left only to the instructors** who moderate the forum.

1 Task 1: Complete Weekly Java Tutorial Videos

- For Lab6, you are assigned to study **Week 7 Part A to Part C1** and **Week 8** (available on March 5) of the Java tutorial series:

  `https://www.youtube.com/playlist?list=PL5dxAmCmjv_7x3Qn5px_zS0qqgaBK9Sc1` [Week 7 only]

  `https://www.youtube.com/playlist?list=PL5dxAmCmjv_6JyoGf4zvQmg3piNzipWdb` [Week 8 only]

  To reference tutorial videos from the previous weeks, see:

  `https://www.youtube.com/playlist?list=PL5dxAmCmjv_6wy2m0yq2wObIWPz4tAxW6` [All Weeks]

  These Java tutorial videos assigned to you are meant for you to:

  1. Obtain extra hands-on programming experience on Java, supplementing your weekly lectures.
  2. Complete the lab assignment with the necessary skills and background.

  Though we do **not** require the submission of the weekly Java tutorial project (like in Lab0), **examples and insights discussed in these tutorials will be covered in your (written and programming) tests and exam**: should you decide to skip the weekly tutorial videos, it would be your choice.

  As you study through the example Java classes in the tutorial videos, you are advised to **type them out** (but absolutely feel free to add new Java classes to experiment) on Eclipse.

- You can find the iPad notes of illustrations from the tutorial videos here:

  `https://www.eecs.yorku.ca/~jackie/teaching/tutorials/notes/EECS1022%20Tutorial%20on%20Java.pdf`

Important Notes to Read (New)

- See this notes on how to manipulate objects with reference-typed, multi-valued attributes:

  `https://www.eecs.yorku.ca/~jackie/teaching/lectures/2021/W/EECS1022/notes/EECS1022_W21_Tracing_PointCollectorTester.pdf`

Notes from Lab5

- See this notes on how to infer classes and methods from given JUnit tests:

  `https://www.eecs.yorku.ca/~jackie/teaching/lectures/2021/W/EECS1022/notes/EECS1022_W21_Inferring_Classes_from_JUnit.pdf`

- You can find here the example covered in the notes for practice:
  - Starter: `https://www.eecs.yorku.ca/~jackie/teaching/lectures/2021/W/EECS1022/notes/EECS1022_W21_Inferring_Classes_from_JUnit.zip`
  - Solution: `https://www.eecs.yorku.ca/~jackie/teaching/lectures/2021/W/EECS1022/notes/EECS1022_W21_Inferring_Classes_from_JUnit_Solution.zip`

Getting Prepared for Programming Test 3

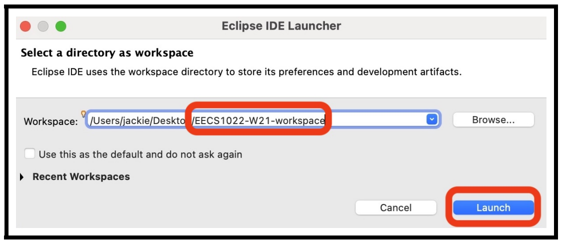
- Your Programming Test 3 will cover classes and objects, and its format will be (virtually) identical to this lab (and Lab5).

2 Task 2: Complete Programming Exercises

Starting Task 2 should mean that you have already completed the weekly Java tutorial videos (Section 1).

2.1 Step 1: Download and Import the Starter Project

1. Download the Eclipse Java project archive file from eClass: `EECS1022_W21_Lab6.zip`
2. Launch Eclipse and browse to `EECS1022-W21-workspace` as the `Workspace` then click on `Launch`, e.g.,



Eclipse IDE Launcher

Select a directory as workspace

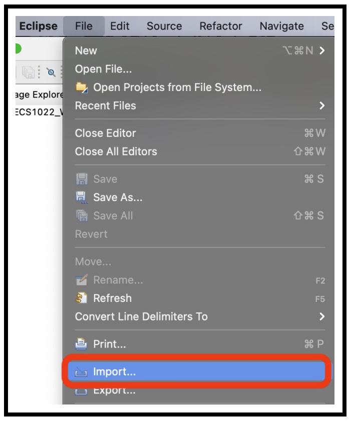
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: /Users/jackie/Deskto/EECS1022-W21-workspace

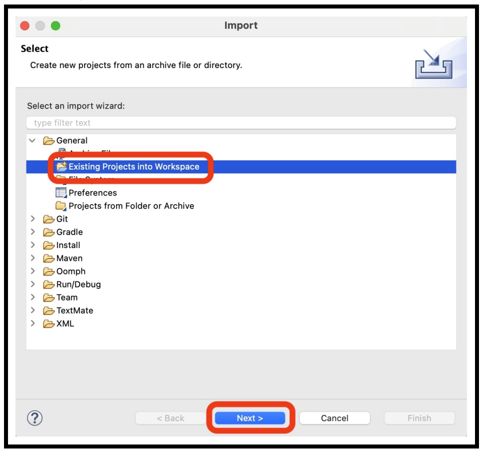
Use this as the default and do not ask again

3. In Eclipse:

3.1 Choose `File`, then `Import`.



3.2 Under `General`, choose `Existing Projects into Workspace`.



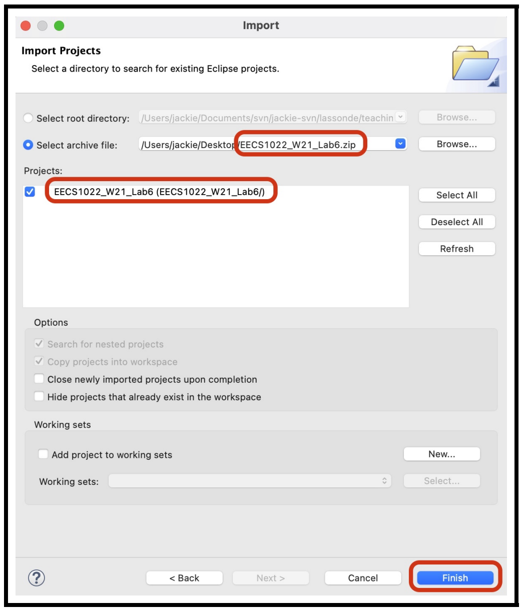
Import

Select

Create new projects from an archive file or directory.

Select an import wizard:

3.3 Choose `Select archive file`. Make sure that the `EECS1022_W21_Lab6` box is checked under `Projects`. Then `Finish`.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory: /Users/jackie/Documents/svn/jackie-svn/lassonde/teachin

Select archive file: /Users/jackie/Desktop/EECS1022_W21_Lab6.zip

Projects:

☑ EECS1022_W21_Lab6 (EECS1022_W21_Lab6/)

Options

✓ Search for nested projects

✓ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

Working sets

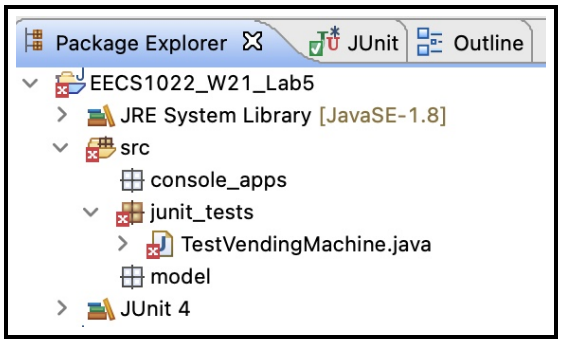
☐ Add project to working sets

Working sets:

2.2 Step 2: Programming Tasks

From the `Package Explorer` of Eclipse, your imported project has the following structure.

- The `console_apps` package is empty. You may add new console application classes here to test the implemented methods if you wish. However, these added console application classes will **not** be graded.
- It is expected that the `TestOnlineSchool` JUnit class contains **compilation errors**. This is because that declarations and definitions of the required class(es) and method(s) it references are missing.



- 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 `TestOnlineSchool` class (after the line `package junit_tests;`):

     ```
     import model.Foo;
     ```

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

     To run them, as shown in the Java tutorials on Week 1, right click on `TestOnlineSchool.java` and run it as JUnit tests. Of course, none of the given tests would pass to begin with.

  You must not modify these given JUnit tests, as they suggest how the intended class(es) and method(s) should be declared.

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.

2.3 The Online School Problem

You are required to develop an object-oriented program solving a (simplified) online school problem, where there is a list of participants registered in courses taught by qualified instructors:

- Each *instructor* is characterized by their name, campus phone extension (e.g., 70310), and contact email.
- Each *registration* is characterized by its subject title, numerical marks, and instructor (who may not be assigned when the course is first created).

  Given a registration object:

  - A grade report may be returned as an array of length 2, e.g., {`"B", "Good"`}, where the first element stores the letter grade and the second element stores its qualitative description. Consider the following table summarizing how each numerical marks (assumed to range between 0 and 100) maps to its grade, description (whose spellings should be exact), and grade point:

Range of Raw Marks	Letter Grade	Qualitative Description	Grade Point
90 – 100	A+	Exceptional	9
80 – 89	A	Excellent	8
70 – 79	B	Good	7
60 – 69	C	Competent	6
50 – 59	D	Passing	5
0 – 49	F	Failing	0

  - A string information object may be returned. There are two cases to consider, depending on whether or not the course instructor has been assigned. In the case where the instructor is present, the returned string should contain the course title, instructor name, the marks, and its corresponding grade and description (see the above mapping table).

- Each *participant* object is characterized by the name of student and the list of added registrations.

  Given a participant object, we may:

  - Add a new registration, either by an input registration object, or by the name of course (from which a registration object may be created accordingly). The maximum number of registrations allowed for a participant is 5: attempting to add registrations beyond this limit will have no impact (i.e., the list of registrations remains the same). Furthermore, there is **no** need to check if there are duplicated registrations added (e.g., two registrations with the same course name).
  - Retrieve its list of registrations as an array (i.e., `Registration[]`), whose length is less than or equal to the maximum allowable number (i.e., 5).
  - Clear its list of registrations (e.g., allowing further registrations to be added).
  - Retrieve the marks of a course with the given name. If the name of a non-registered course is given, then return `-1` as its marks.
  - Update the marks of a course with the given name. If the name of a non-registered course is given, then nothing should be changed.
  - Obtain a report of the GPA (grade point average) over the list of added registrations.

- Each *online school* object is characterized by its list of participants.

  Given an online school object, we may:

  - Add a new participant by an input participant object. The maximum number of participants allowed for a school is 100: attempting to add participants beyond this limit will have no impact (i.e., the list of participants remains the same). Furthermore, there is **no** need to check if there are duplicated participants added (e.g., two participants with the same name).
  - Retrieve the list of participants of a course, given its name, as an array. If the input name denotes a non-existing course, then an empty array is returned.

- **Other intended functionalities of above kinds of objects can be inferred from the given JUnit test class `TestOnlineSchool`.**

2.4 Hints and Requirements

- See this notes on how to declare and manipulate reference-typed, multi-valued attributes:

  https://www.eecs.yorku.ca/~jackie/teaching/lectures/2021/W/EECS1022/notes/EECS1022_W21_Tracing_PointCollectorTester.pdf

  Try to finish the above notes before Week 8 Java Tutorials are released (March 5).

- See this notes on how to infer classes and methods from given JUnit tests:

  https://www.eecs.yorku.ca/~jackie/teaching/lectures/2021/W/EECS1022/notes/EECS1022_Inferring_Classes_and_Methods_from_JUnit_Tests.pdf

  Programming IDEs such as Eclipse are able to fix such compilation errors for you. **However, you are advised to follow the guidance as specified in the notes to fix these compilation errors manually, because: 1) it helps you better understand how the intended classes and methods work together; and 2) you may be tested in a written test or exam without the assistance of IDEs.**

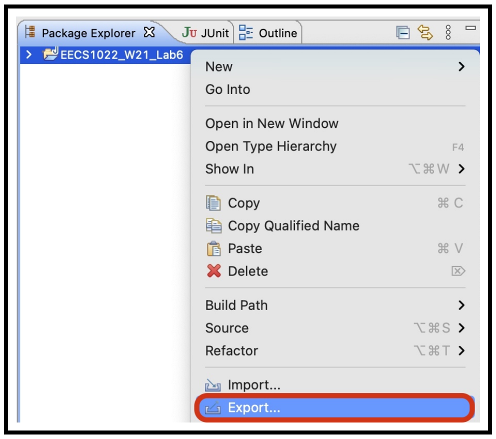
- Any new class(es) you add must reside in the `model` package.
  - Once the necessary class(es) and method(s) are declared, you can add as many attributes as necessary to implement the body of each method.
  - Study carefully example instances as specified in `TestOnlineSchool.java`: they suggest the how the intended class(es) and method(s) should be declared and implemented.
  - Focus on *gradually* passing one test at a time.
  - You **cannot** use any Java library classes (e.g., `ArrayList`) or methods for implementation. That is, there must **not** be any `import` statement in the class(es) you add to the `model` package.

2.5 Step 3: Exporting the Completed Project

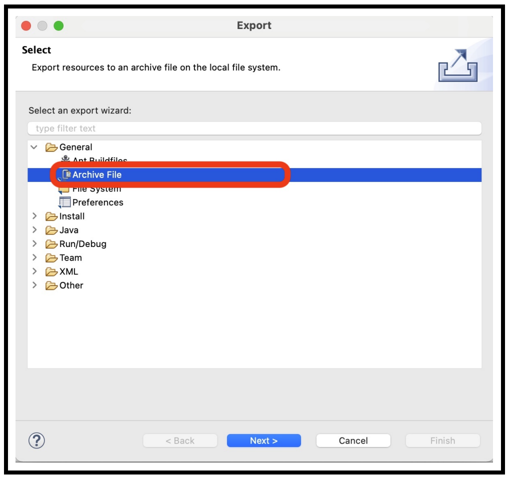
You are required to submit a Java project archive file (.zip) consisting all subfolders.

In Eclipse:

1. Right click on project `EECS1022_W21_Lab6`. Then click `Export`



2. Under `General`, choose `Archive File`.

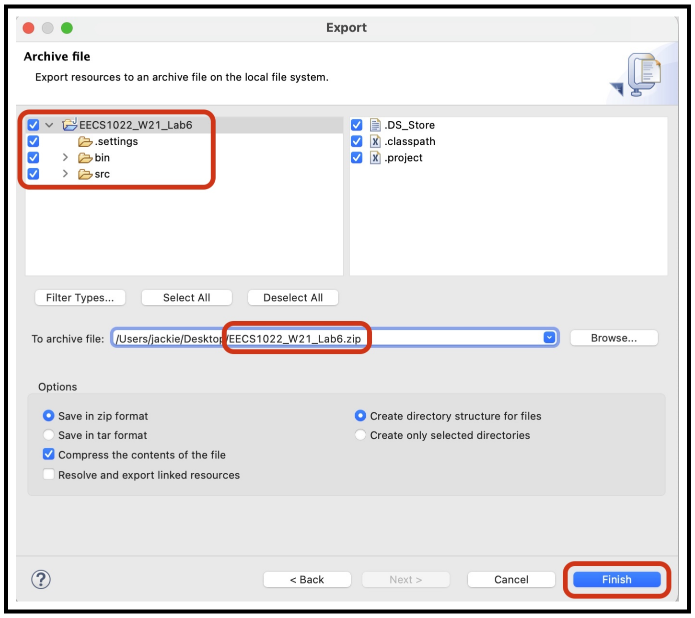


3. Check the top-level `EECS1022_W21_Lab6`

Make sure that all subfolders are checked: `.settings`, `bin`, and `src`.

Under `To archive file:` browse to, e.g., desktop, and save it as `EECS1022_W21_Lab6.zip` (case-sensitive)

Then `Finish`.



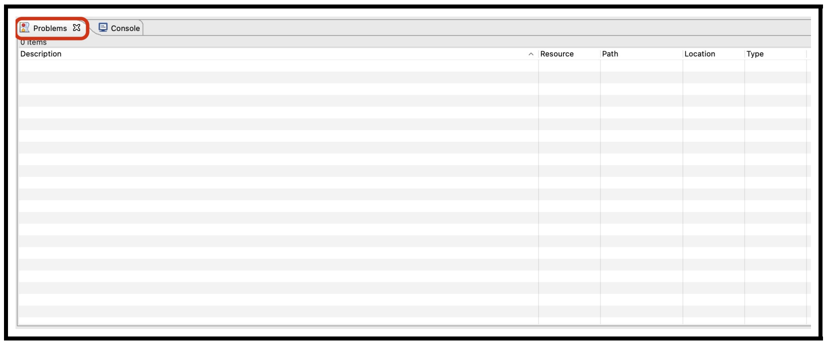
To archive file: /Users/jackie/Desktop/EECS1022_W21_Lab6.zip

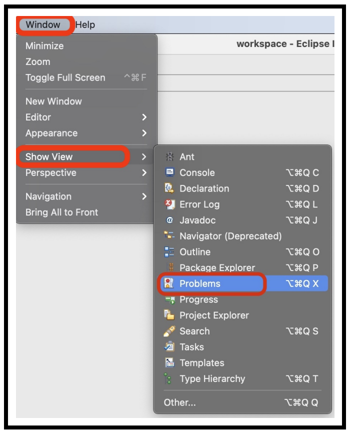
[✓ Save in zip format] Save in tar format
[✓ Compress the contents of the file]
Resolve and export linked resources
[✓ Create directory structure for files] Create only selected directories

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 **<u>are</u>** 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 2.5 asks you to **export** the Java project as an archive file:

   `EECS1022_W21_Lab6.zip`

3. Go to the eClass site for Sections M,N,O: `https://eclass.yorku.ca/eclass/course/view.php?id=6214`

4. Under the `Lab Submissions` section, click on `Lab6` to submit the Java archive file: `EECS1022_W21_Lab6.zip`

   - 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.
   - **Once you click on the `submit` button, you can no longer upload another draft version.**