ECS 140A: Fall 2023
Homework Assignment 2

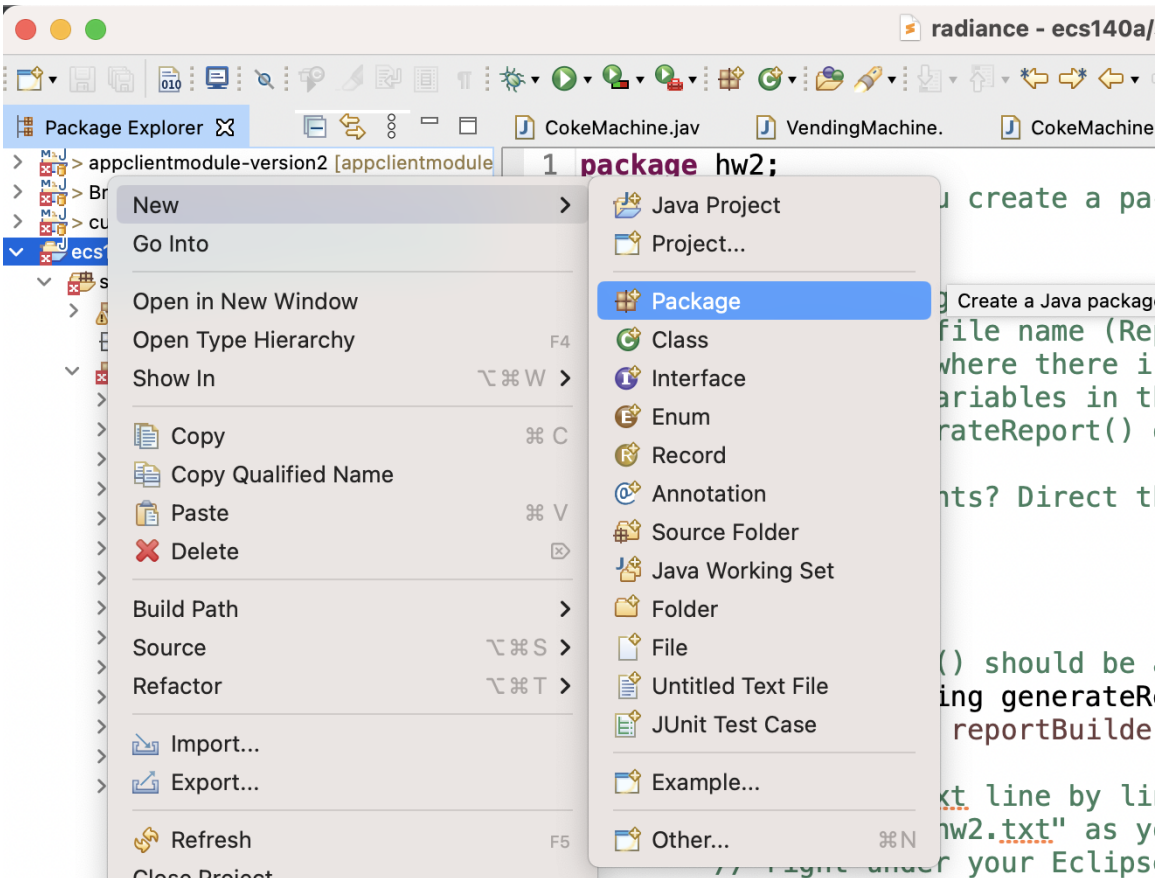
Due Date: No later than Monday, November 6, 11:00pm.
Much of what you need to know to complete this assignment has been covered in class, but you'll need to do some research to solve some of the problems you will encounter (e.g., how to read from a file, how to print).

All the files you need for this assignment are provided in **hw2.zip** on Canvas.

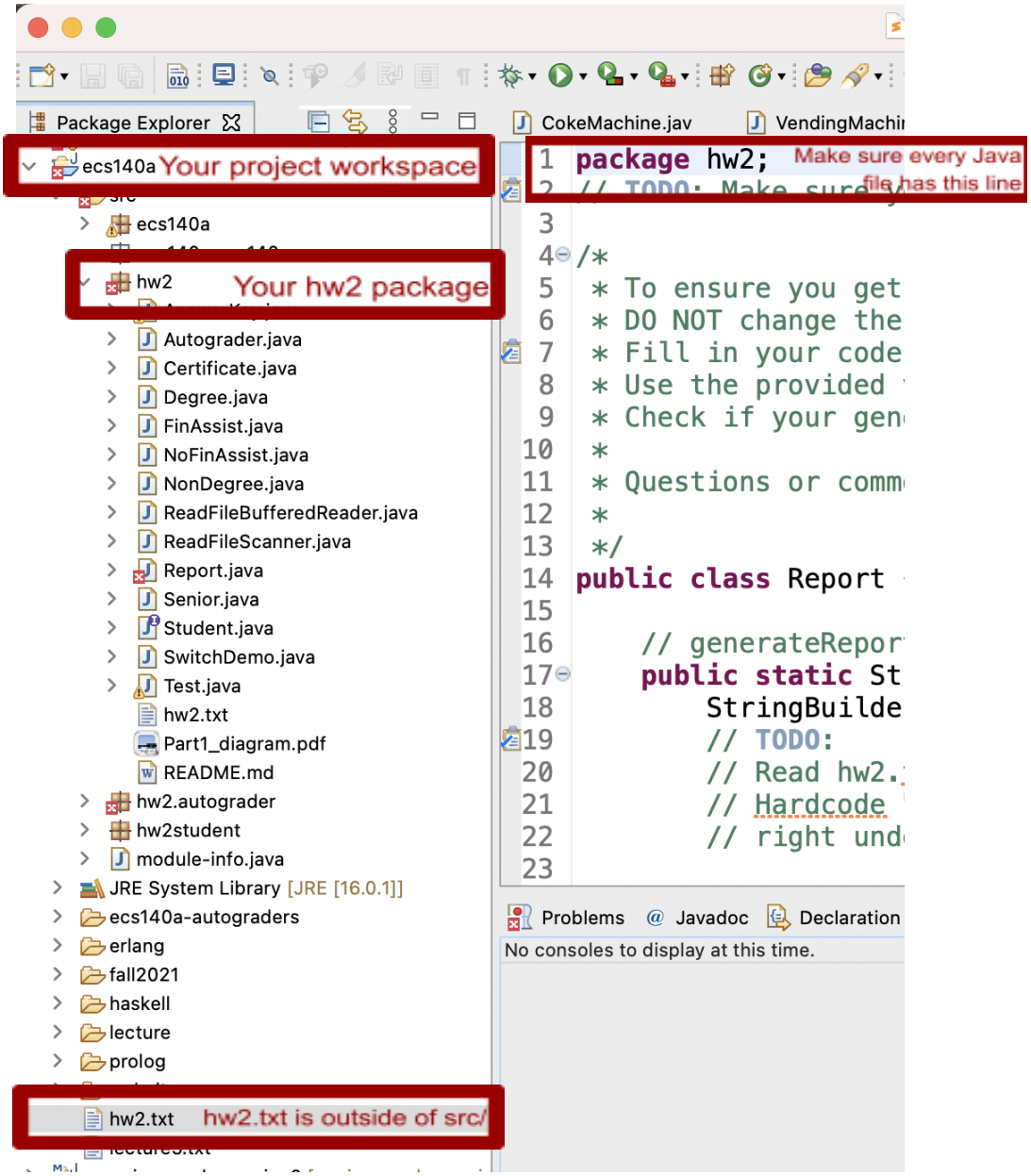
It is possible that we will have to make some adjustments to this assignment as the days go by. Please try to be flexible.

Part Zero - Java Project Setup

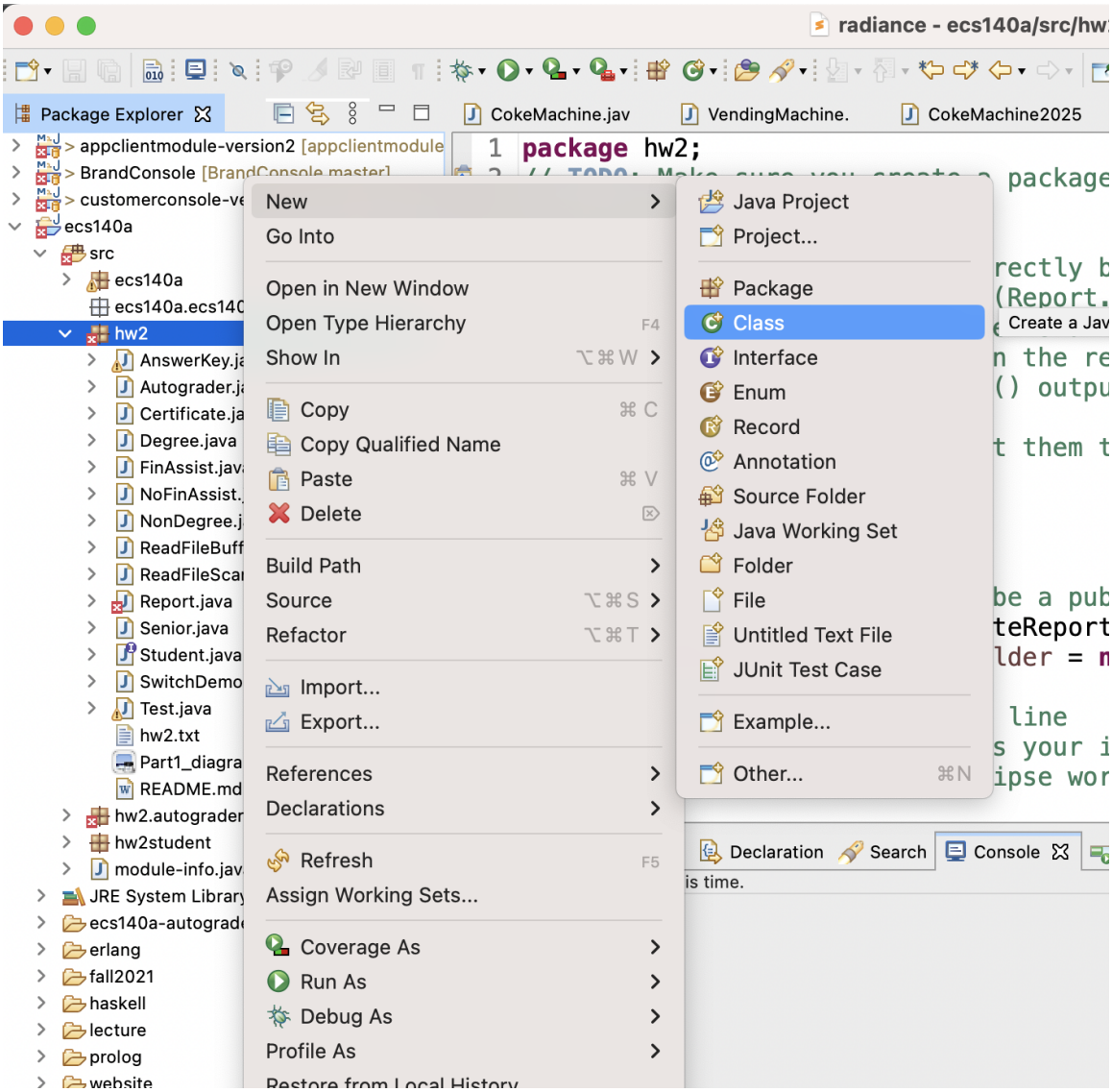
Before you start coding, you would need to set up your Eclipse workspace. You may follow the eclipse-instructions.pdf on Canvas to set it up. You can name your workspace/Java project as ecs140a. Within the Java project, **create a package named hw2** by right clicking on the project and choose New > Package. All your Java files must be under the hw2 package, with `package hw2;` being the first line of every Java file you create. Naming the package exactly as hw2 is necessary for the autograder to compile all your Java files correctly. Here is how you can create a Java package in Eclipse.



Download the input file **hw2.txt** and sample output file **hw2_output.txt** from Canvas and move them into your ecs140a/ directory (at the top level of ecs140a). This is important because you will hardcode the input file path as "hw2.txt" in Report.java for the autograder. Here is a screenshot of where hw2.txt should be. The sample output file is provided for you to check your output.



You can create your Java files by right clicking on the hw2 package we just created, then choose New > Class or New > Interface, depending on what you need.



Part One

Assume that you have been hired by the newly-founded Tonopah State University in Nevada to create software that will track and report financial information about its several types of students.

The information for every student will include:
- student identification number (six digits)
- first name
- last name
- age in years
- number of credit hours enrolled in this term

The entire population of students can be divided into two subgroups: degree-seeking students and non-degree seeking students. In addition to the information listed above, the University would like to keep track of the following information for degree-seeking students:
- major (gaming science; hotel management; lounge arts; beverage engineering)
- academic standing (good; warning; probation)

Note that non-degree seeking students are not eligible for financial assistance.

The University has no need to track this information for non-degree seeking students.

Within the category of degree-seeking students, the University further distinguishes between degree-seeking students who have financial assistance and those who do not. Students with financial assistance have all the attributes of students without financial assistance, with the addition of:
- dollar value of financial assistance available per term

The amount of the financial assistance will be deducted from the student's assessed fees.

Similarly, Tonopah State recognizes two different types of non-degree-seeking students: senior citizens and certificate students. Senior citizens are students who are 65 years of age or older and who are taking courses for personal enrichment but who are not pursuing a degree or certificate. (Note that a degree-seeking student or a certificate student might also be 65 or older; these students would not fall into the senior citizen category.) Certificate students are taking a short sequence of courses in pursuit of a vocational credential in one of the four major disciplines described above. Thus, your program should keep track of the kind of certificate each student is pursuing. Also, keep in mind that Tonopah State may want to add additional non-degree-seeking student subcategories in the future.

Tonopah State University wants to be able to print all the recorded data for any given student onto the monitor. This should be achieved by a single call to a method named `printData` on the corresponding object. Here's an example of what the results of the `printData` method should look like when applied to a degree-seeking student with financial assistance:

```
ID number: 046352
Name:      Moe Howard
Age:       32

Moe is a degree-seeking student enrolled in 11 credits
Moe receives $500.00 in financial assistance per term
Moe's major is beverage engineering
Moe's academic standing is good
```

Tonopah State also wants to be able to print the fees assessed to a given student for the term. This should be done by calling a method named `computeFees` on the corresponding object. The fee assessment is computed in different ways, depending on the type of student, as follows:

Certificate student: fees are a fixed assessment of $700 per term plus $300 for every credit hour the student has enrolled in this term.

Senior citizen: fees are a fixed assessment of $100 per term for six or fewer enrolled credit hours. Add an additional $50 for every credit hour greater than six.

Degree-seeking student without financial assistance: Fees are a recreation and athletics fee of $100, a student union fee of $50, and $275 per credit hour, up to a maximum of twelve credit hours. Thus, the maximum fee assessment would be $100 + $50 + $275*12 = $3450. If the student enrolls in more than twelve credit hours, the additional credit hours will appear in the student's records but the student is assessed only for twelve hours; the additional hours are essentially free of charge.

Degree-seeking student with financial assistance: Fees are the same as those for the degree-seeking student with financial assistance, ***except that the dollar value of financial assistance per term is subtracted from the fee assessment***. If the fee assessment drops below zero, then no fees are assessed.

In this part of the assignment, you have to design classes to represent the various categories of students described above. Interfaces, abstract classes, and fully implemented classes (also known as concrete classes) may all be useful. Or maybe not...that's up to you. This part of the exercise is more for your benefit than for ours. Nevertheless, we expect you to submit a graphical representation of your design in the form of a class hierarchy. It doesn't have to be perfect, it just needs to be readable. You may draw the diagram by hand if you like. For this part of the assignment, you will submit a PDF file with your design via Gradescope.

Part Two

In the second part of the assignment, you are asked to use Java to write the software necessary to generate two types of reports for Tonopah State. We provided the skeleton code **(Report.java)** on Canvas for you to format your outputs. Fill in the **generateReport()** method within the Report class after you have implemented your Student classes. Do not add, modify, or delete the StringBuilder statements in this file. Add your code to where there is a TODO comment in Report.java. You may call classes defined in other files.

First, you'll need to read the student data from a file. The input data is stored in a file in the following format:

```
046352;Moe;Howard;32;11;Y;E;G;Y;500.0
172458;Larissa;Pine;20;12;Y;A;W;N
611030;Dorothy;Gale;48;9;N;C;S
498545;Frank;Baum;68;3;N;S
```

Each line describes one student. The first string in the line is the student ID number, followed by the first and last names, the age, and the number of credit hours. These are followed by a single character code: Y denotes "Yes, this student is a degree-seeking student" and N denotes "No, this is not a degree-seeking student".

For degree-seeking-students, the "Y" is followed by a single-character code for the major (S = gaming Science, M = hotel Management, A = lounge Arts, and E = beverage Engineering), another single-character code for the academic standing (G = Good, W = Warning, and P = Probation), and one more single-character code (Y denotes "Yes, this student has financial assistance" and N denotes "No, this student does not have financial assistance"). If the student is receiving financial assistance, this last "Y" will be followed by the amount of financial assistance per term. If the student does not receive financial assistance, there will be no additional information.

For non-degree-seeking students, the "N" is followed by a single-character code for the type of non-degree-seeking student (C = Certificate student; S = Senior citizen). If the student is a certificate student, the "C" will be followed by a single-character code indicating the type of certificate being pursued (S = gaming Science, M = hotel Management, A = lounge Arts, and E = beverage Engineering). If the student is a senior citizen, there will be no additional information following the "S".

Your program must read data in this format from a file. Depending on the different codes, one line from the file will result in an object of the appropriate type being created. A reference to the object should then be stored in an array. The maximum size of this array is 100 (it's a very small university at this time), but your program should work with an arbitrary number of lines in the file, so long as the number of lines does not exceed 100. A file of test data is provided on Canvas as `hw2.txt`.

Once the data has been read from the file, your program should then be able to generate two different reports. The first report is a listing of all the students, one line per student, showing the fees each student has been assessed.

The second report is a summary of the amount of money the University has assessed each of the four types of student. Such a report might look like this:

```
Summary of student fees assessed:

Degree-seeking students without financial assistance:
$32,430
Degree-seeking students with financial assistance: $17,390
Certificate students: $8,050
Senior citizens: $2,100

Total fees assessed: $59,970
```

Your two reports should match exactly with the text in hw2_output.txt. Make sure you follow the detailed instructions in Report.java and make use of the skeleton code.

Submission Instructions

To submit your assignment, you will drag and drop your class diagram pdf from part one and all your Java files (including Report.java) to Gradescope by Nov. 6 at 11pm. Do not put them in any zip files or folders. You may want to submit earlier than the due date to ensure your code works with the autograder. You have unlimited attempts. If the autograder produces an error, try to debug by reading the error message and line number of your code with error. Make sure you have followed all the instructions in this document.