Homework 3 Due: Friday, July 28, 11:59 AM

You will rewrite Homework 2 to run as a JavaFX program, with a few other changes. Create project Homework 3 for this assignment. IntelliJ will create a java directory under src; delete it. Place Hw3Driver.jave and the two packages in src. Reuse as much of your code from Homework 2 as possible. The main method will change – it's a JavaFX program, so of course it will call launch( ).

These options are ***not*** implemented in this version (due to time constraints): search by last name; search by last and first name; sort (because only one student is shown at a time); enter or delete a course for an existing (or new) student – note that this means that new students will not have a schedule file generated. When a CSV file of student data is saved, the order may not be what you expected, depending on your Map.

When creating a JavaFX program, IntelliJ requires a Group field. I took the default, com.example, and placed the StudentPackage and SchedulePackage inside it. This required some editing of import statements. It also requires that the model-info.java file open and export these packages; here is my version of that file:

```
module com.example.homework3 {
    requires javafx.controls;
    requires javafx.fxml;


    opens com.example.homework3 to javafx.fxml;
    opens com.example.homework3.SchedulePackage to javafx.fxml;
    exports com.example.homework3;
    exports com.example.homework3.SchedulePackage;
}
```

Changes to the default package (src)

class Hw3Driver
Declare a `Map<String, Student>` as a public static variable outside of main; the key is the id field for a Student object. This replaces the ArrayList<Student> from Homework 2. Create it when the FileChooser is invoked on the Open menu choice. Also declare StudentRecordProcessor and StudentReaderWriter as private static outside of main (instead of inside).
class StudentMenuProcessing – no longer needed.

main() – call launch( ).
Other methods: up to you – try to organize the setup of each part of the GUI.

Changes to package StudentPackage

class StudentReaderWriter – change these methods:
`public static Map<String, Student> readRecords(String filename)` - reads the file, creates and returns the map (or null if the file does not exist).
`public static void writeRecords(String filename, Map<String, Student)` – writes the file from the map.

class StudentRecordProcessor – change these methods:
contstructor – takes the map as a parameter.
`public static Student searchByID(String id)` – finds the Student object with this id in the map and returns it, or null if not found.
`public static void append(Student s, Map<String, Student> studentMap)` – writes this Student object to the map.

There are four entries on the File menu: About, which shows the program name and your name; Open, which opens and loads a file of Student records – use a FileChooser to get the file name – this replaces prompting the user for the file name at the beginning of the program; Save, which writes a file of Student records – use a FileChooser for this, too, using its showSaveDialog() method – this replaces prompting the user for the output file name at the end of the program; and Quit, which ends the program.

The Schedule menu has one choice, Open Schedule, which opens and reads a Schedule file and displays it in a TextArea in a popup window (a new stage). This code will help anchor the window:

```
myStage.initModality(Modality.APPLICATION_MODAL);
myStage.initOwner(stage);
```

where stage is the overall app stage. Handling schedule data this way is a little klunky: the user must enter the CSV data correctly, instead of having separate text fields for each part of the Course data.

Use a JavaFX `GridLayout` for laying out the labels, text fields, and buttons dealing with student data. Add the parts to the grid using column and row numbers, like this (first int is the column):

```
studentGridPane.add(idLabel, 0, 0);
studentGridPane.add(idField, 1, 0);
```

There's also a status field to show things like "FILE NOT LOADED" if the student data file is empty.

The buttons in the grid do the following: Search by ID scrapes the id text field and looks for that student in the map and displays it and sets the status field to "FOUND" or displays "NOT FOUND" in the status field if not found. New Student first blanks out all the data fields, disables the Search by ID and New Student buttons, and enables the Enter Data button (which is normally disabled). The Enter Data button (when enabled) scrapes the data in the text fields (except the age field – this is still computed from the date of birth), adds a new student to the map, and sets the status field to "NEW STUDENT ENTERED", then disables itself and enables the other buttons; if any of the data fields are blank, a student cannot be created, so set the status field to "BAD DATA", disable the button and enable the others. Use Button's setDisable(true) to disable a button or setDisable(false) to enable it.

Use a JavaFX `TableView` to display the courses in the schedule. To set the columns, create a new TableColumn<Course, String> with a name, then associate it with the specific Course field, then add it to the TableView. For example, the first column would be:

```
nameColumn = new TableColumn<>("Name");
nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));
tableView.getColumns().add(nameColumn);
```

and similarly with the other columns. Then, when you have a Course object to display in the table, do this:

```
tableView.getItems().add(course);
```

which will set all the columns with course's data.

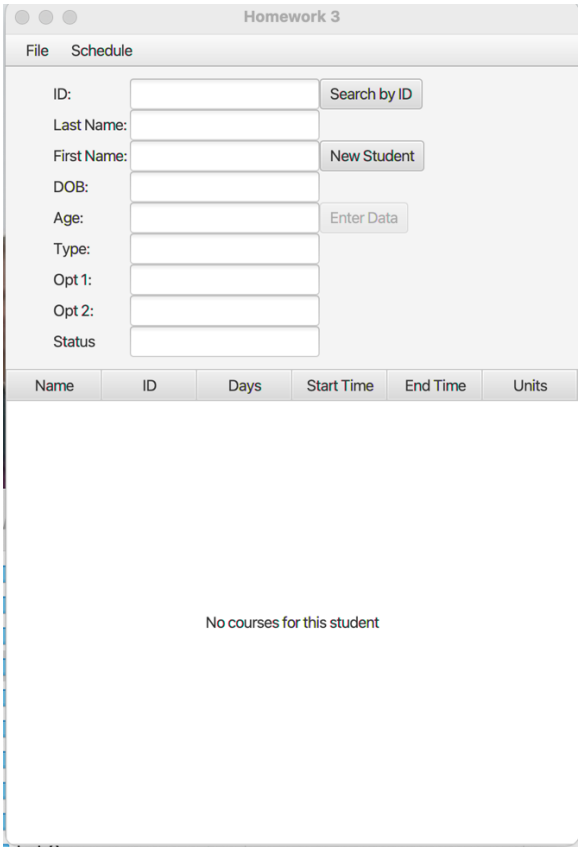
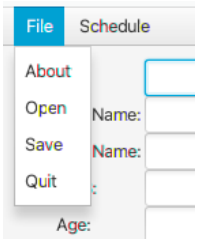


Figure 1: Program startup. The TableView default message was set with `setPlaceholder(new Label("No courses for this student"));`



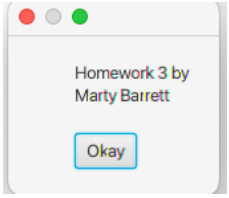


Figure 2: File menu and About choice. The Okay button closes the popup.

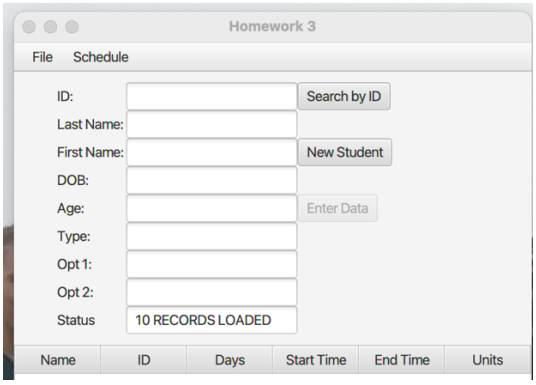


Figure 4: After students.csv is read using the Open option in the File menu. The FileChooser is not shown. Status field shows the number of Student records read.

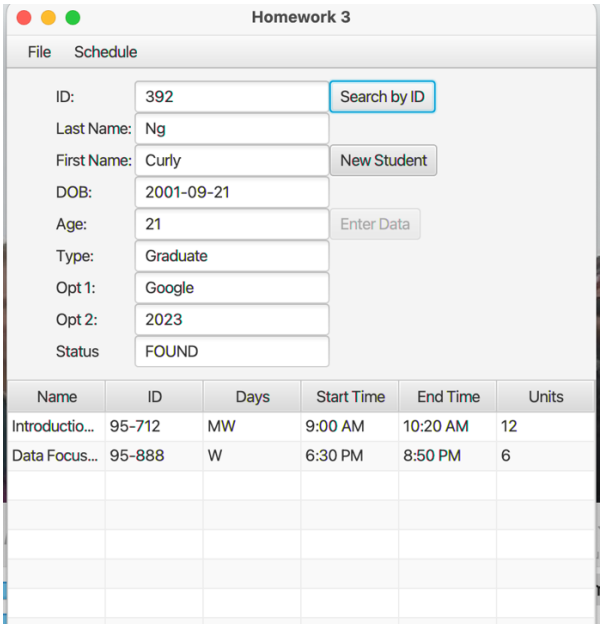


Figure 5: After Search By ID for Graduate student #392. The TableView is populated by reading "392.csv". The Type field is either Undergraduate or Graduate. For Undergraduate students, Opt 1 will contain the major and Opt 2 will be blank.

For Type, Opt 1, and Opt 2, you'll likely need to use the instanceOf operator.

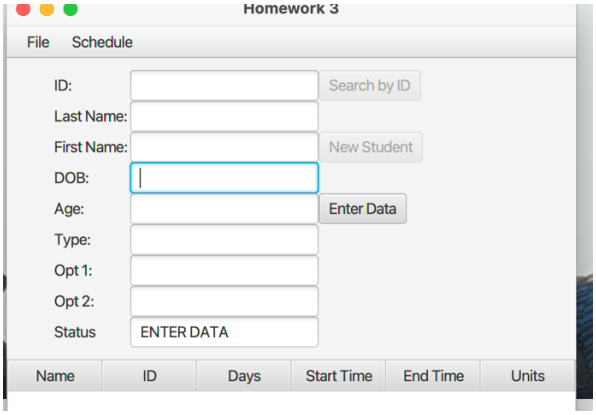


Figure 6: After pressing New Student – clear the data fields and gray-out the other buttons.

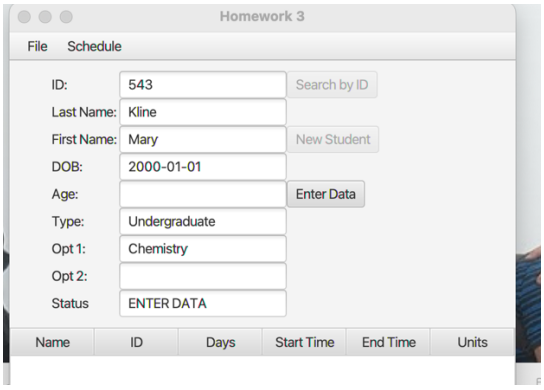


Figure 7: New data entered, before pushing Enter Data. The Date of Birth ***must*** be properly formatted; the formatting could be forced by the text field, but here we trust the user to enter it as yyyy-mm-dd. The age field is left blank.

The user must type out "Undergraduate" or "Graduate". For Undergraduate, the major is entered into Opt 1; for Graduate, the Internship is entered into Opt 1 and the year is entered into Opt 2.

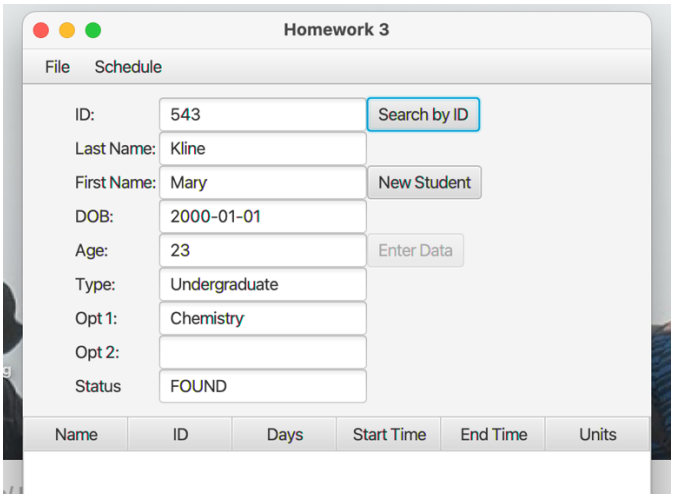


Figure 8: After pressing Search By ID, the same record is re-displayed; note the correct Age field, computed in Student. Also note this student does not have a Schedule.

Instructions:
Zip up all your java files. Name the zip file <your-andrew-id>-hw3.zip and submit it to Canvas.
Grading Rubric:
1. 80% on console output and manual testing.
2. 5% documentation: document each class and each method. Document your code only where needed – where the person grading it might have trouble understanding.
3. 5% Code quality: variable names, optimal loops, etc.
4. 5% Robustness: your code should not crash while processing reasonable data.
5. 5% Following submission instructions. NO LATE SUBMISSION, PLEASE!