CS 272 Final Exam

Databases, Model-View-Controller and JavaFX

Earthquakes: Please download **FinalExam_Programming.zip** and import this existing project into Eclipse (you do not need to unzip the file).

Open DBModel.java

This Java class serves as the model for our application and uses the earthquakes.csv file (data from the USGS) to populate a SQLite database.

Here is the one task to complete in DBModel (everything else is done):

1. `[5 points] private void createTable() throws SQLException`

In this method, you are to create an SQL string to create a database table if it does not exist.

This method is supposed to loop through all the fields and values to construct a StringBuilder with the following format:

```
"CREATE TABLE IF NOT EXISTS earthquake(_id INTEGER PRIMARY KEY,
area TEXT, month INTEGER,  day INTEGER,year INTEGER, hour
INTEGER, minute INTEGER, magnitude REAL)"
```

Be sure to execute the update after the SQL string is constructed.

Open Controller.java

This Java class utilizes the Singleton design pattern, such that there is only one controller for the entire application. A controller accepts input from the View (Java FX) and converts it into commands for the Model (DBModel.java). Specifically, the Controller loads an ObservableList with Earthquake data from the database. It then uses this ObservableList to conduct queries and perform filters on the data.

Here's what you'll need to complete for the Controller:

1. `[5 points] public ObservableList<String> getDistinctAreas()`

This method should:

a. Loop through the all earthquakes list, one earthquake at a time
b. If the areasList does not contain the earthquake's area (state or country), add it to the areasList
c. Sort the areasList after all have been added

2. **`[5 points] public boolean`** `removeEarthquake(Earthquake eq)`

   This method should first check to see if the earthquake is null. If b is null, return false.

   Otherwise, remove the earthquake from the ObservableList of all earthquakes and also invoke a call to deleteRecord(…) of DBModel to delete the earthquake from the database as well (permanent deletion). Return true if successful, false if there is an SQLException.

3. **`[5 points] public`** `ObservableList<Earthquake> filter(String area, double magnitude)`

   This method should:

   a. Loop through the all earthquakes list
   b. If an earthquake's area matches the area specified AND is greater than or equal to (>=) the magnitude specified, add it to the filteredEarthquakesList.
   c. Return the filtered list

Open ListEarthquakesScene.java

This Java class provides the view (the visualization) of nodes which present data from the Controller. Most of the design has been done for you already (since this is old material), but there are a few new tasks to complete, as follows:

Here’s what you’ll need to add: for the delete feature (new methods, do not exist yet):

1. **`[5 points] public void`** `initialize(URL location, ResourceBundle resources)`

   a. Set the items of the earthquakes list view, using the controller
   b. Set the items of the areas combo box using the controller
   c. Disable the remove earthquake button
   d. Set the text of the count label to the number of earthquakes displayed

2. **`[5 points] public Object`** `removeEarthquake() {`

   This method should be attached to the delete button (setOnAction).

   When the removeEarthquakeButton is clicked

   a. Get the selected earthquake from the earthquakesListView
   b. Use the controller to call the removeEarthquake(Earthquake eq) method you created earlier.
   c. Select -1 on the list view to clear any selections
   d. Disable the delete button (since nothing is selected)
   e. Update the test of the countLabel with the new number of earthquakes

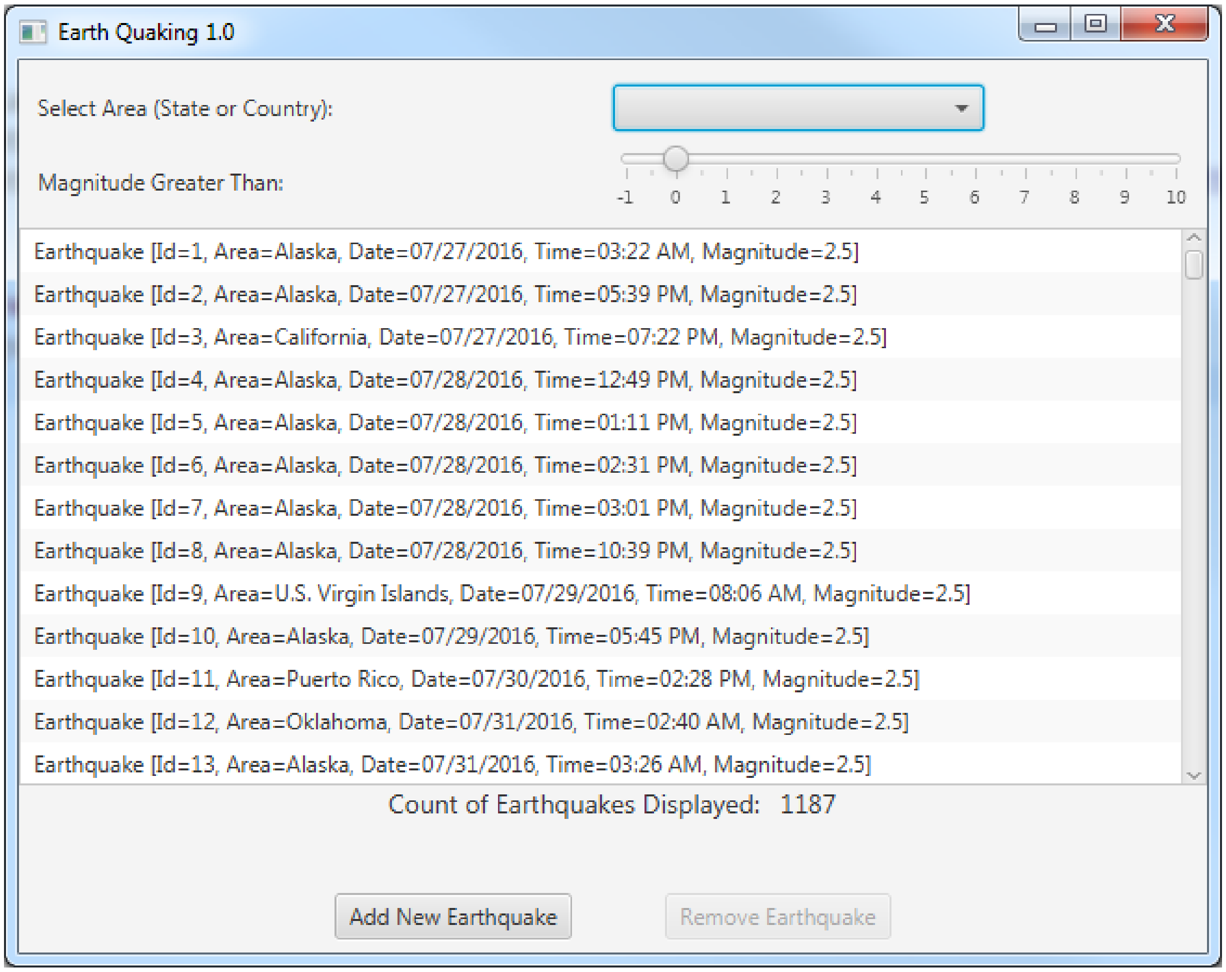
3. `[10 points] public Object filter() {`

This method should be attached to the areasCB (combo box) and magnitudeSlider.

When the combo box or slider values change, the list view should be filtered as follows:

a. Use the value of the areasComboBox and the magnitudeSlider to filter the list of what is displayed in the list view.
b. Update the countLabel to reflect the count of earthquakes in the filtered list.

All Earthquakes (remove disabled):

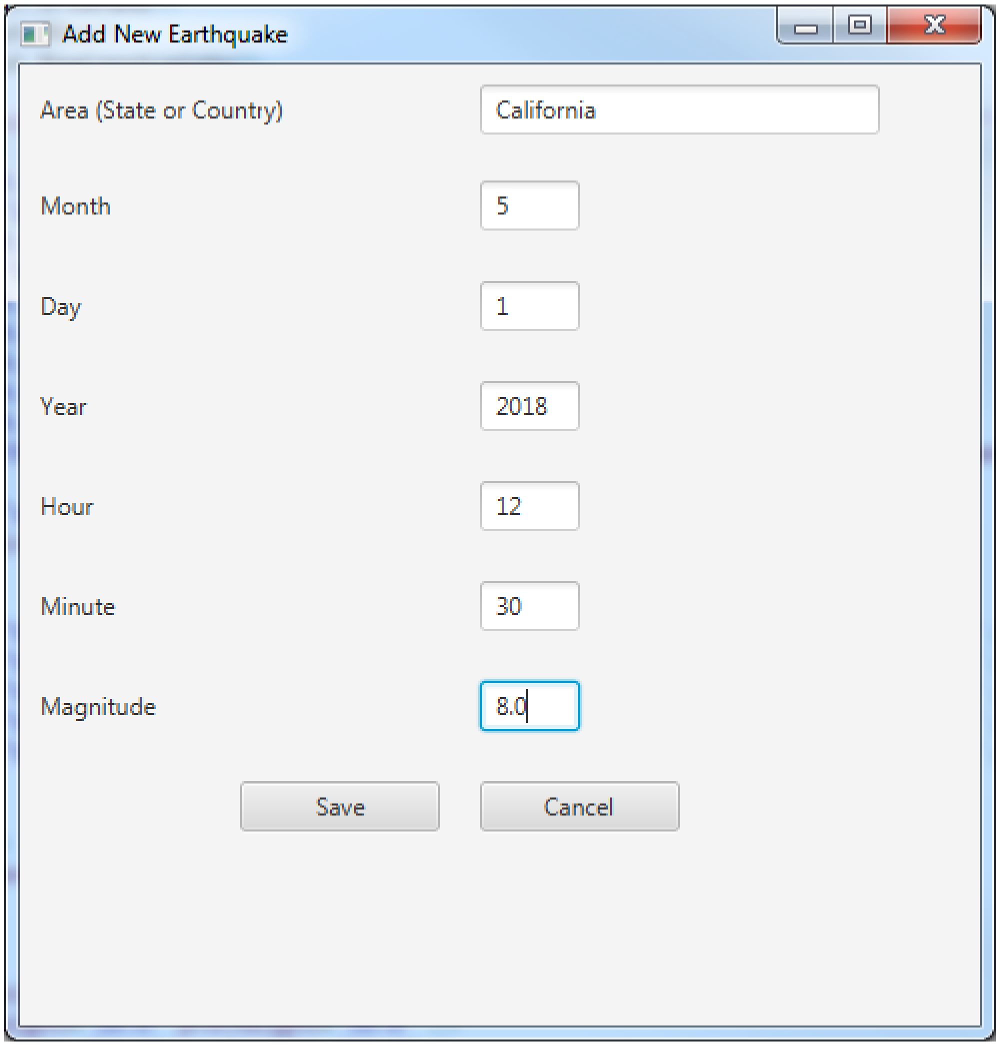


```
Earthquake [Id=1, Area=Alaska, Date=07/27/2016, Time=03:22 AM, Magnitude=2.5]
Earthquake [Id=2, Area=Alaska, Date=07/27/2016, Time=05:39 PM, Magnitude=2.5]
Earthquake [Id=3, Area=California, Date=07/27/2016, Time=07:22 PM, Magnitude=2.5]
Earthquake [Id=4, Area=Alaska, Date=07/28/2016, Time=12:49 PM, Magnitude=2.5]
Earthquake [Id=5, Area=Alaska, Date=07/28/2016, Time=01:11 PM, Magnitude=2.5]
Earthquake [Id=6, Area=Alaska, Date=07/28/2016, Time=02:31 PM, Magnitude=2.5]
Earthquake [Id=7, Area=Alaska, Date=07/28/2016, Time=03:01 PM, Magnitude=2.5]
Earthquake [Id=8, Area=Alaska, Date=07/28/2016, Time=10:39 PM, Magnitude=2.5]
Earthquake [Id=9, Area=U.S. Virgin Islands, Date=07/29/2016, Time=08:06 AM, Magnitude=2.5]
Earthquake [Id=10, Area=Alaska, Date=07/29/2016, Time=05:45 PM, Magnitude=2.5]
Earthquake [Id=11, Area=Puerto Rico, Date=07/30/2016, Time=02:28 PM, Magnitude=2.5]
Earthquake [Id=12, Area=Oklahoma, Date=07/31/2016, Time=02:40 AM, Magnitude=2.5]
Earthquake [Id=13, Area=Alaska, Date=07/31/2016, Time=03:26 AM, Magnitude=2.5]
```

Count of Earthquakes Displayed: 1187

Create new FXML design and Controller named AddEarthquakeScene.fxml and AddEarthquakeScene.java

1. **[10 points]** Create a Scene that resembles the following design. Make sure to give ids to each control (node) that will be needed to save a new earthquake in the FXML. Associate the FXML scene with its corresponding controller (AddEarthquakeScene.java).



Add New Earthquake

Area (State or Country)	California
Month	5
Day	1
Year	2018
Hour	12
Minute	30
Magnitude	8.0

2. **[8 points]** Implement a method to save a new earthquake (attach it to the saveButton), which will add a new earthquake to both the ObservableList (in the Controller) and the database (also done by the controller).
3. **[2 points]** Implement a method to cancel the operation (attach it to the cancelButton), that will use the ViewNavigator to take the user back to the ListEarthquakesScene.

Submitting Your Work on Canvas

When you are finished, upload a Word document (named LastNameFirstName_FinalExam.docx) and a zip file of your Java project to Canvas for credit. You should only have one Word document for all parts of this project, AND one zip file for the Eclipse project.

Copy the code from all classes above **(JUST THE CODE YOU EDITED)** into a Word document and label each part. Then take screenshots (images) of each phase, including:

- When app first loads
- After filtering with combo box
- After filtering with slider
- After removing an earthquake
- After adding a new earthquake

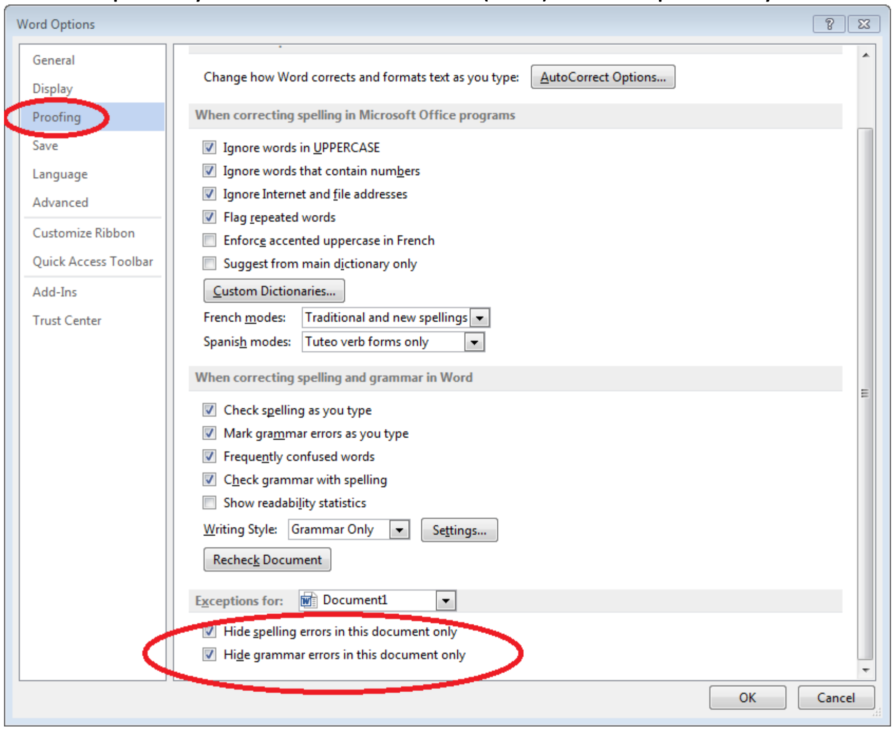
Please see guidelines for **turning off spelling and grammar checking** below:

Finally, create an export (zip file) of your Eclipse project. You should submit **1 Word document and 1 zip file** (name LastNameFirstName_FinalExam_Programming.zip) for full credit on this assignment.

Please **turn off** spelling and grammar checking for the Word document, by going to **File -> Options -> Proofing**. Check the 2 boxes at the bottom:

"Hide spelling errors in this document only" and "Hide grammar errors in this document only" (see screen shot below for details, if needed)

Please upload your Word document (first) and a zip file of your entire Java project (second) to Canvas.



Change how Word corrects and formats text as you type:

When correcting spelling in Microsoft Office programs

- [✓] Ignore words in UPPERCASE
- [✓] Ignore words that contain numbers
- [✓] Ignore Internet and file addresses
- [✓] Flag repeated words
- [ ] Enforce accented uppercase in French
- [ ] Suggest from main dictionary only

French modes: Traditional and new spellings
Spanish modes: Tuteo verb forms only

When correcting spelling and grammar in Word

- [✓] Check spelling as you type
- [✓] Mark grammar errors as you type
- [✓] Frequently confused words
- [✓] Check grammar with spelling
- [ ] Show readability statistics

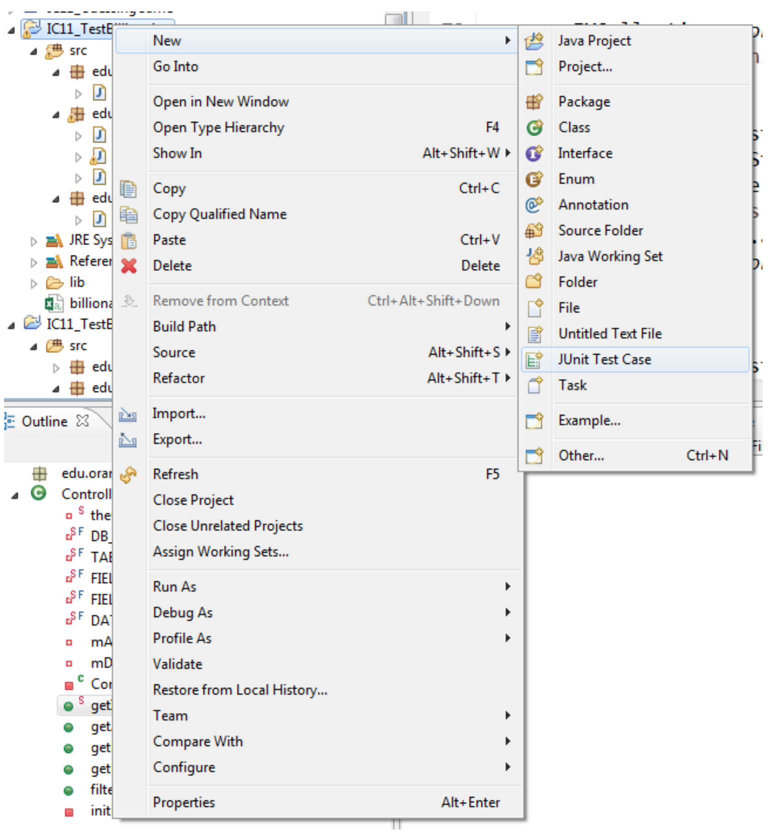
Writing Style: Grammar Only

Exceptions for: Document1

- [✓] Hide spelling errors in this document only
- [✓] Hide grammar errors in this document only

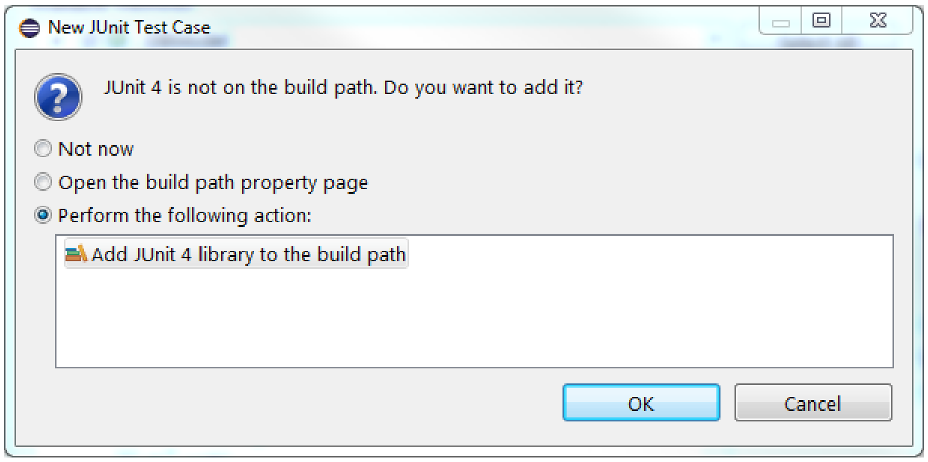
Extra Credit [+5 points] Create new JUnit Test Case named TestDBModel

Right-click on the FinalExam_Programming project, then choose New -> JUnit Test Case. Name the class TestDBModel.



Next, choose all method stubs (see screen shot below) and browse for the "Class under test", which should be edu.orangecoastcollege.cs272.finalexam.model.DBModel

Select all the methods under DBModel (other than the constructor DBModel(String, String[], String[]), then click the Finish Button. When it asks you to add JUnit to the build path, please click OK.


New JUnit Test Case

JUnit 4 is not on the build path. Do you want to add it?

- ( ) Not now
- ( ) Open the build path property page
- [✓ Perform the following action:]

Add JUnit 4 library to the build path

Complete the JUnit testing for all methods in order to earn the +5 points extra credit.