GUI with JavaFX

Lab #7

COMP3021 2022 Spring

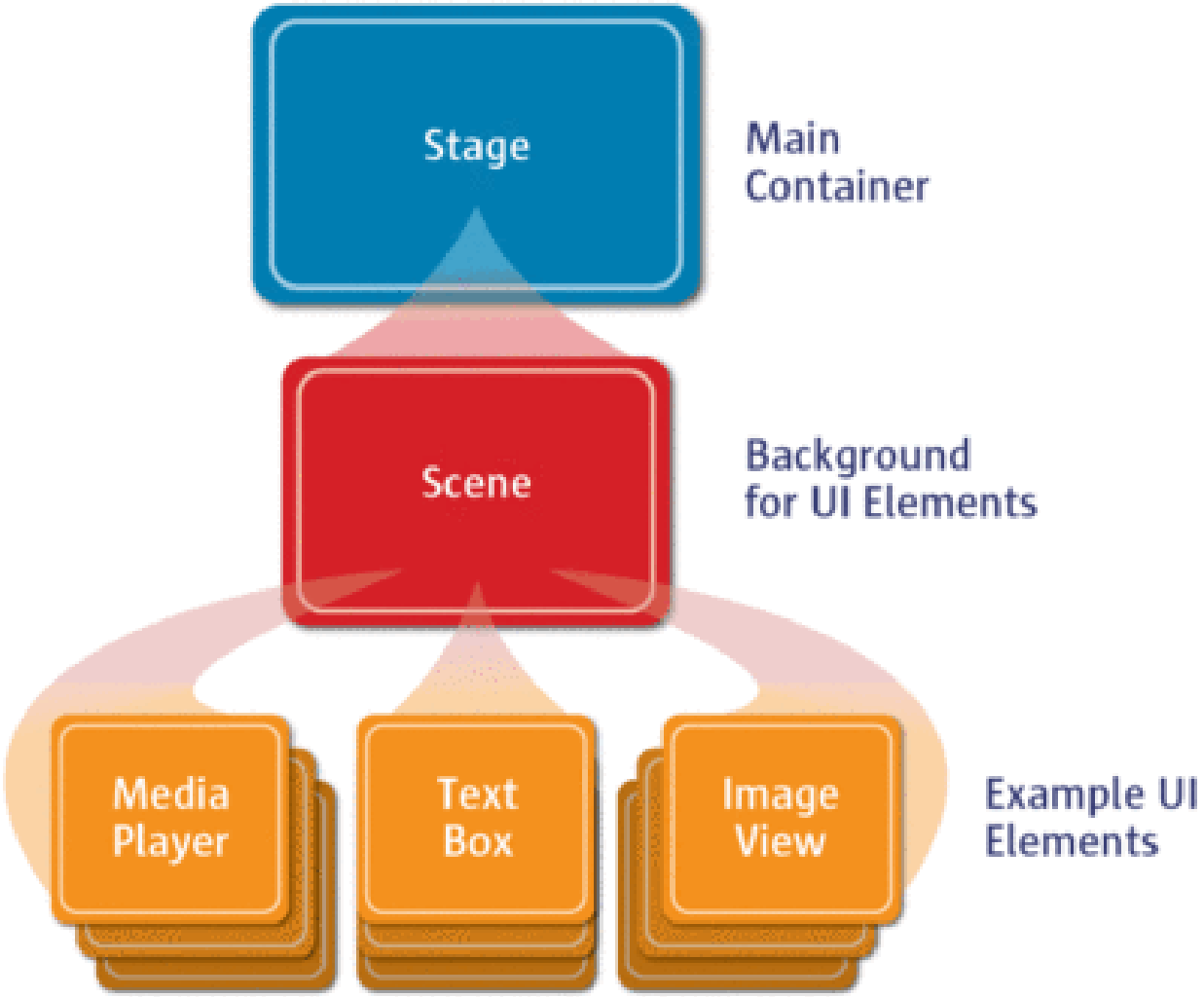
ChengPeng Wang(cwangch@cse.ust.hk)
Yiyuan Guo(yguoaz@cse.ust.hk)
Bowen Zhang(bzhangbr@cse.ust.hk)
Heqing Huang(hhuangaz@cse.ust.hk)

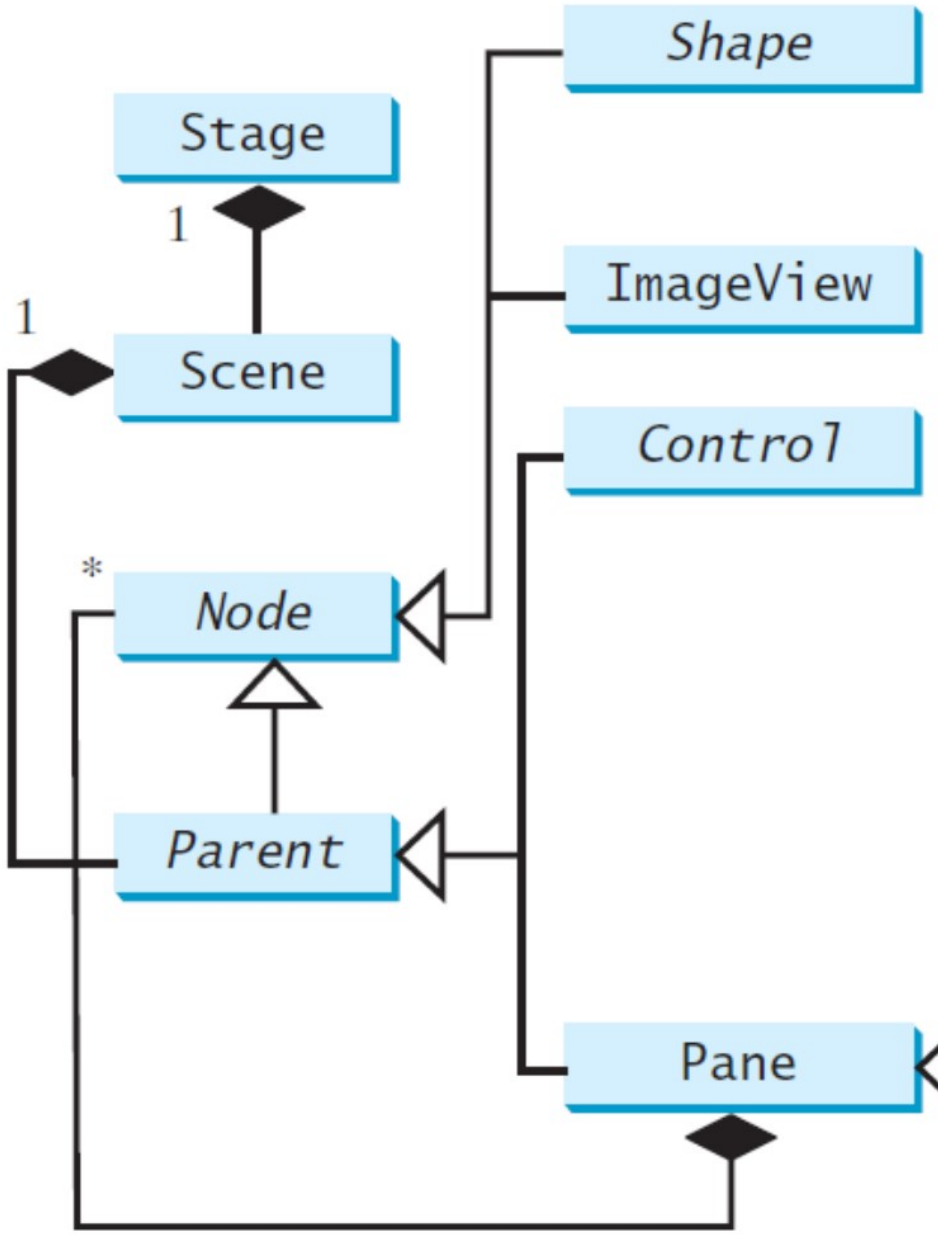
- **Objectives of this lab**

  Learn How to Create GUI with JavaFX

  Create the GUI for your notebook

  Create event listeners

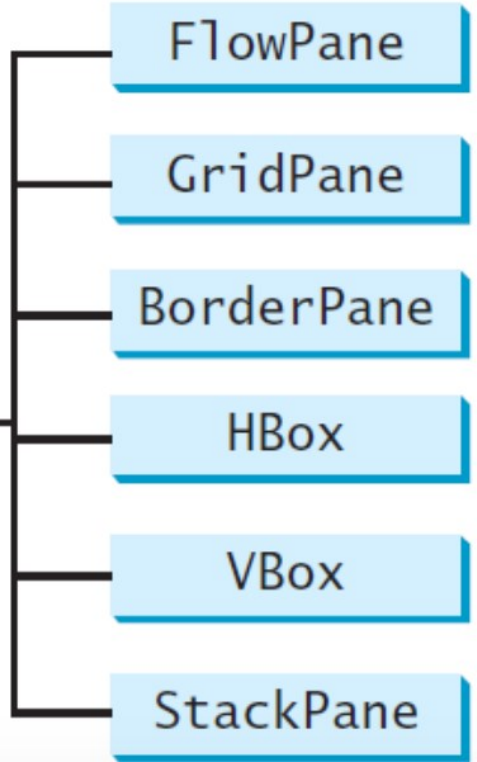

Stage
Main Container
Scene
Background for UI Elements
Media Player
Text Box
Image View
Example UI Elements



Shapes such as Line, Circle, Ellipse, Rectangle, Path, Polygon, Polyline, and Text are subclasses of Shape.

For displaying an image.

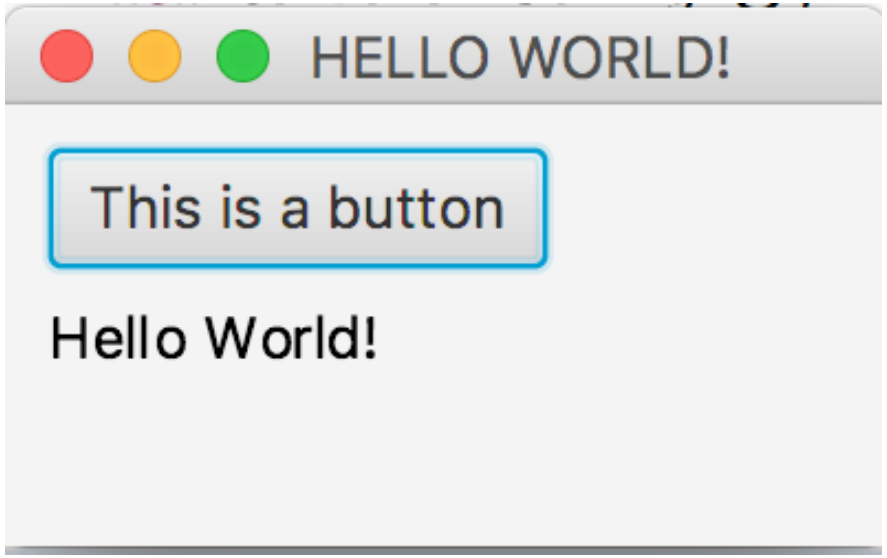
UI controls such as Label, TextField, Button, CheckBox, RadioButton, and TextArea are subclasses of Control.



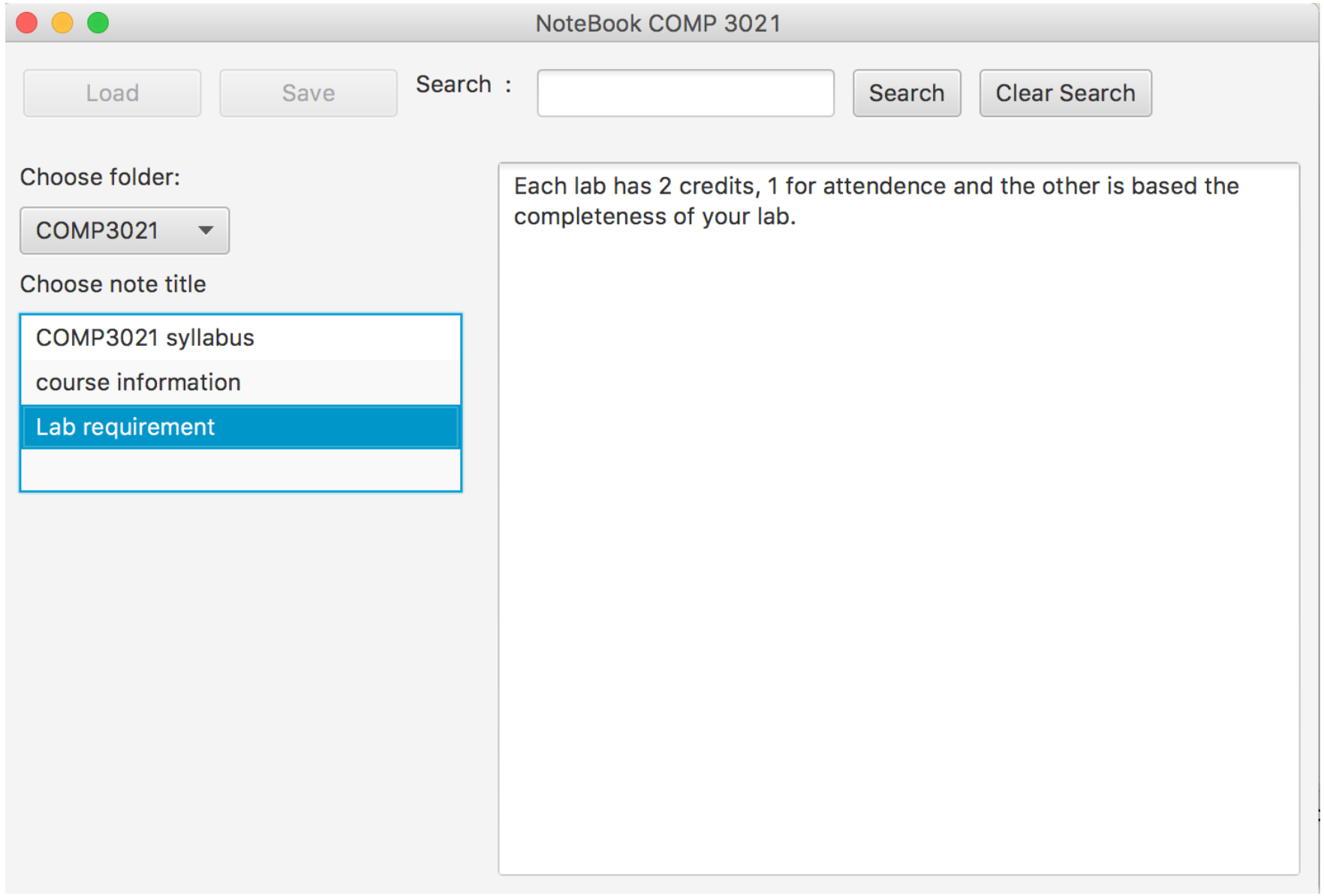
Class	*Description*
`Pane`	Base class for layout panes. It contains the `getChildren()` method for returning a list of nodes in the pane.
`StackPane`	Places the nodes on top of each other in the center of the pane.
`FlowPane`	Places the nodes row-by-row horizontally or column-by-column vertically.
`GridPane`	Places the nodes in the cells in a two-dimensional grid.
`BorderPane`	Places the nodes in the top, right, bottom, left, and center regions.
`HBox`	Places the nodes in a single row.
`VBox`	Places the nodes in a single column.

- **HelloWorld in JavaFX**

```
public class HelloWorld extends Application {
   public static void main(String[] args) {
            launch(args);
      }
   @Override
   public void start(Stage stage) {
      stage.setTitle("HELLO WORLD!");
      VBox vbox = new VBox(); //create pane
            vbox.setPadding(new Insets(10)); //add space
            vbox.setSpacing(8); //vertical space between nodes
             // add two nodes
            vbox.getChildren().add(new Button("This is a button"));
             vbox.getChildren().add(new Text("Hello World!"));
            //create scene
            Scene scene = new Scene(vbox, 200, 100);
      stage.setScene(scene);
            stage.show();
      }
}
```



• LAB #6 Final Result



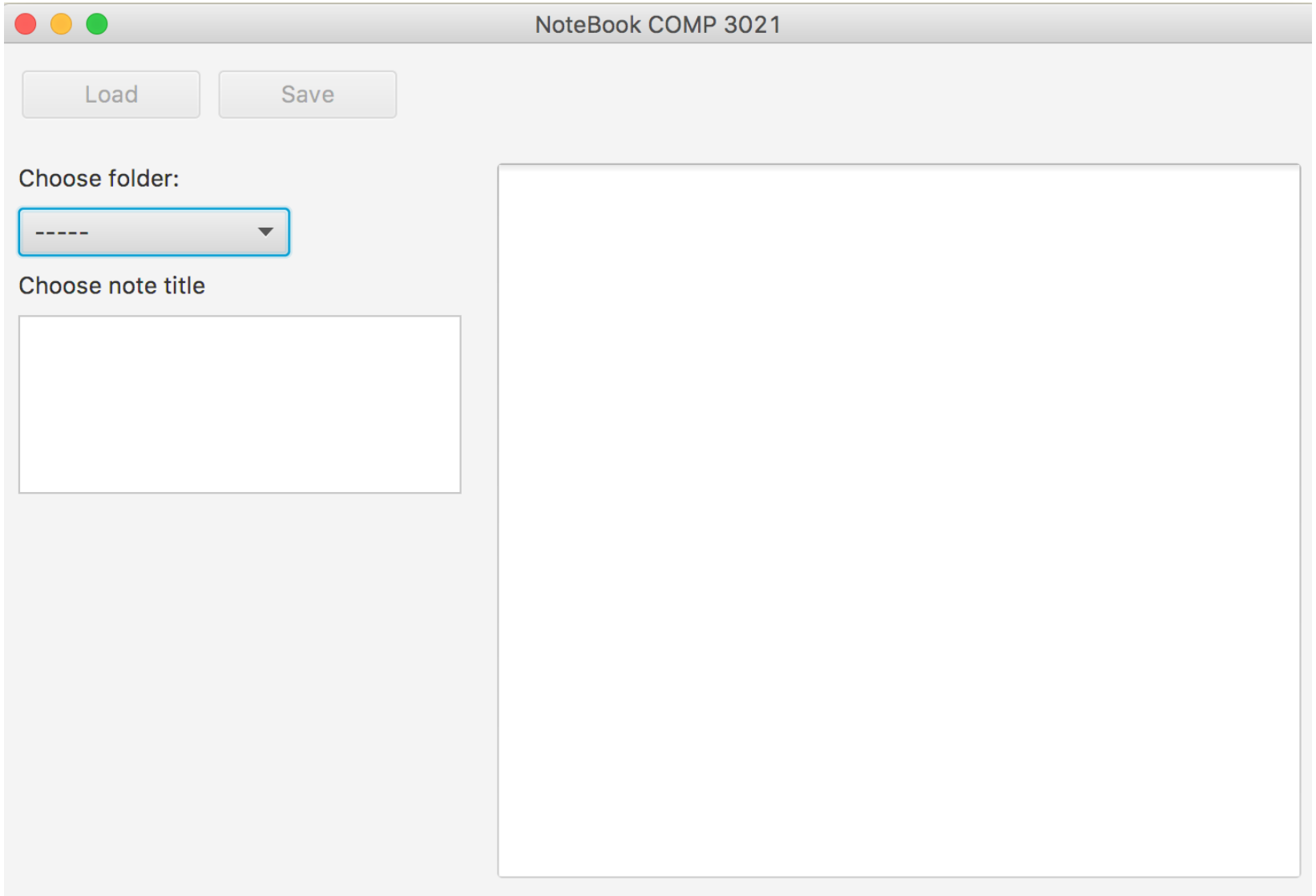
- **TASK 1: Import the initial JavaFX class**

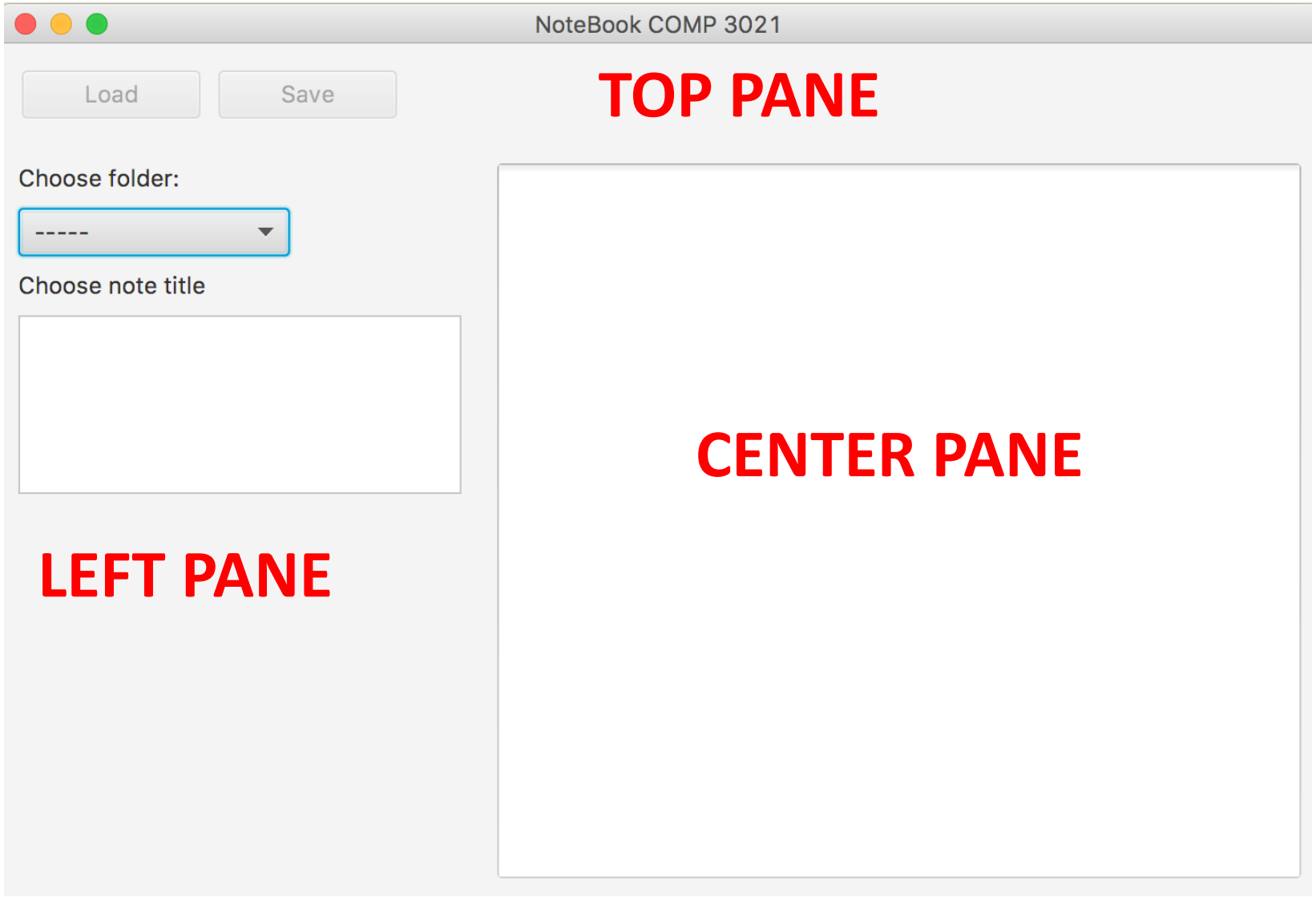
1. Download the `NoteBookWindow.java` class from Canvas.

2. Copy this file under **base** package, where we implemented lab 2 – 4.

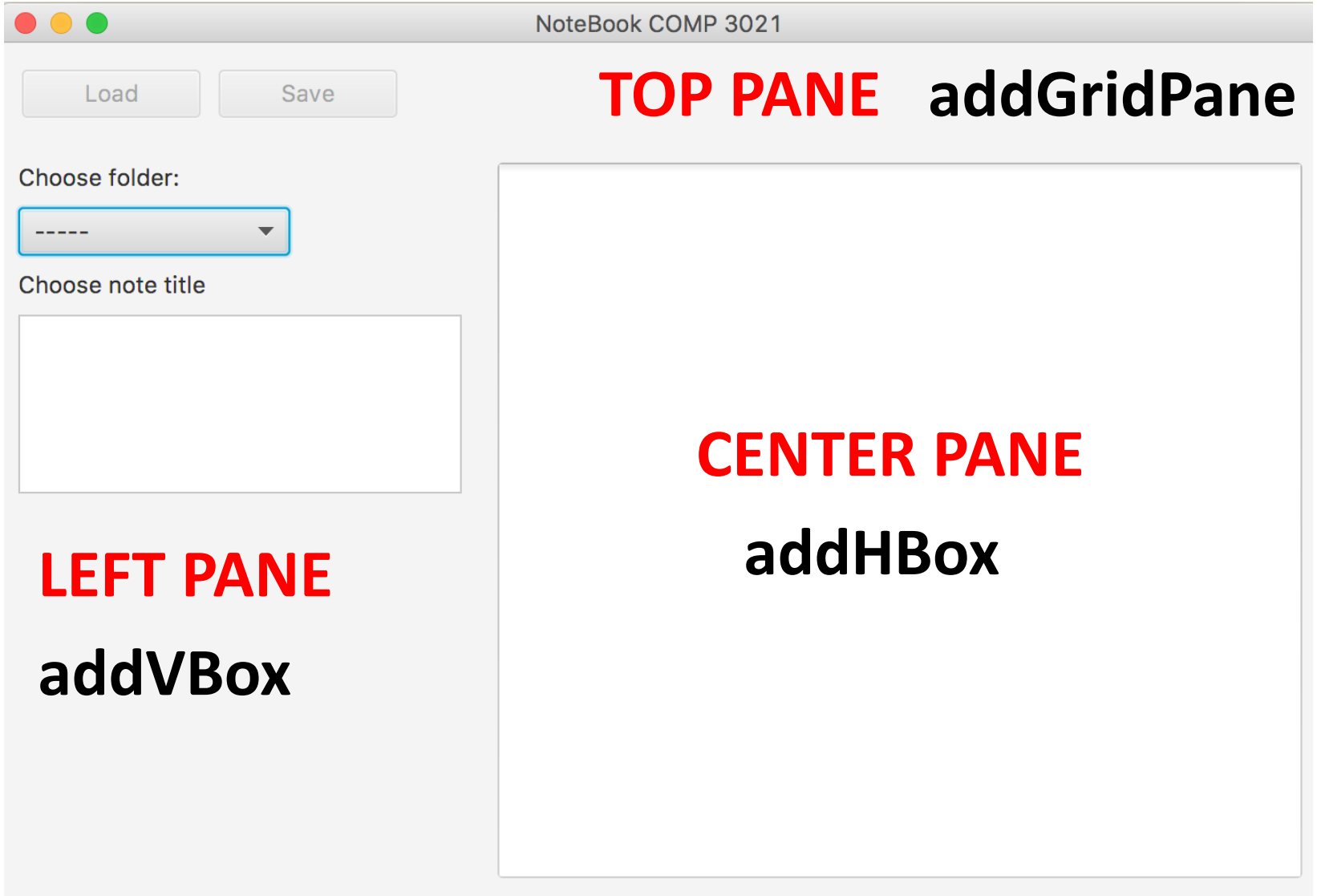
3. Get familiar with the class methods and fields

- **If you run the class you should see this**



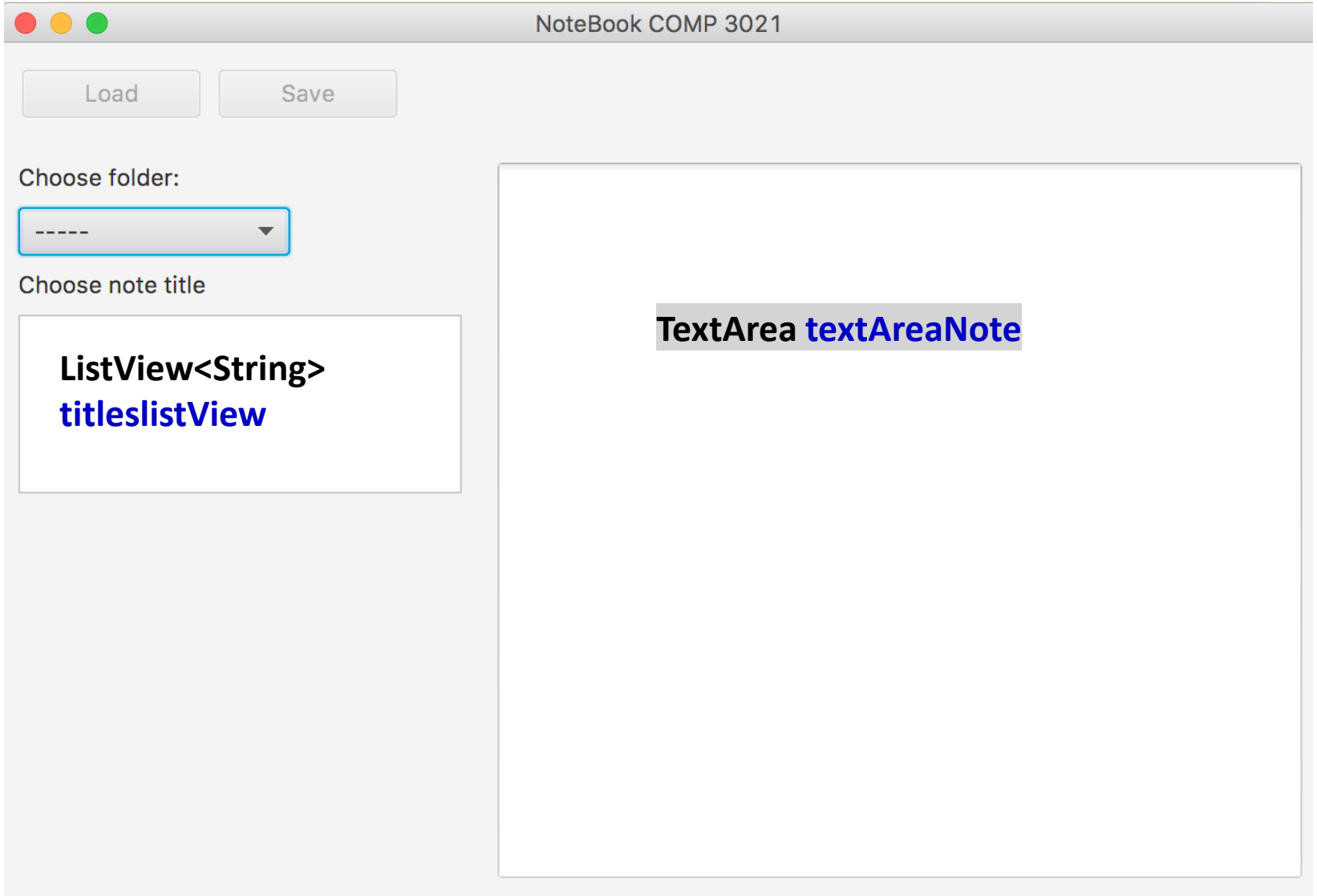

NoteBook COMP 3021
Load
Save
TOP PANE
Choose folder:

Choose note title
CENTER PANE
LEFT PANE


NoteBook COMP 3021
Load
Save
TOP PANE
addGridPane
Choose folder:

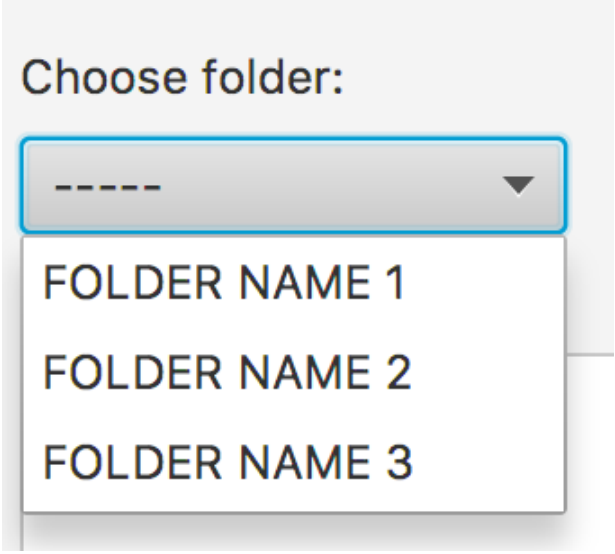
Choose note title
CENTER PANE
addHBox
LEFT PANE
addVBox

ComboBox<String>
foldersComboBox

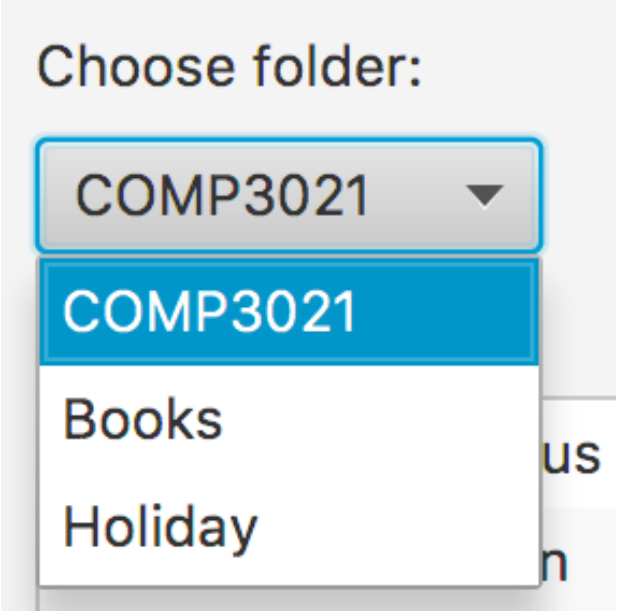


- **TASK 2: Populate the foldersComboBox**

The **foldersComboBox** shows:



You have to load the folder names from the **noteBook** object.
Your comboBox should be like this



- **TASK 2: Populate the foldersComboBox**

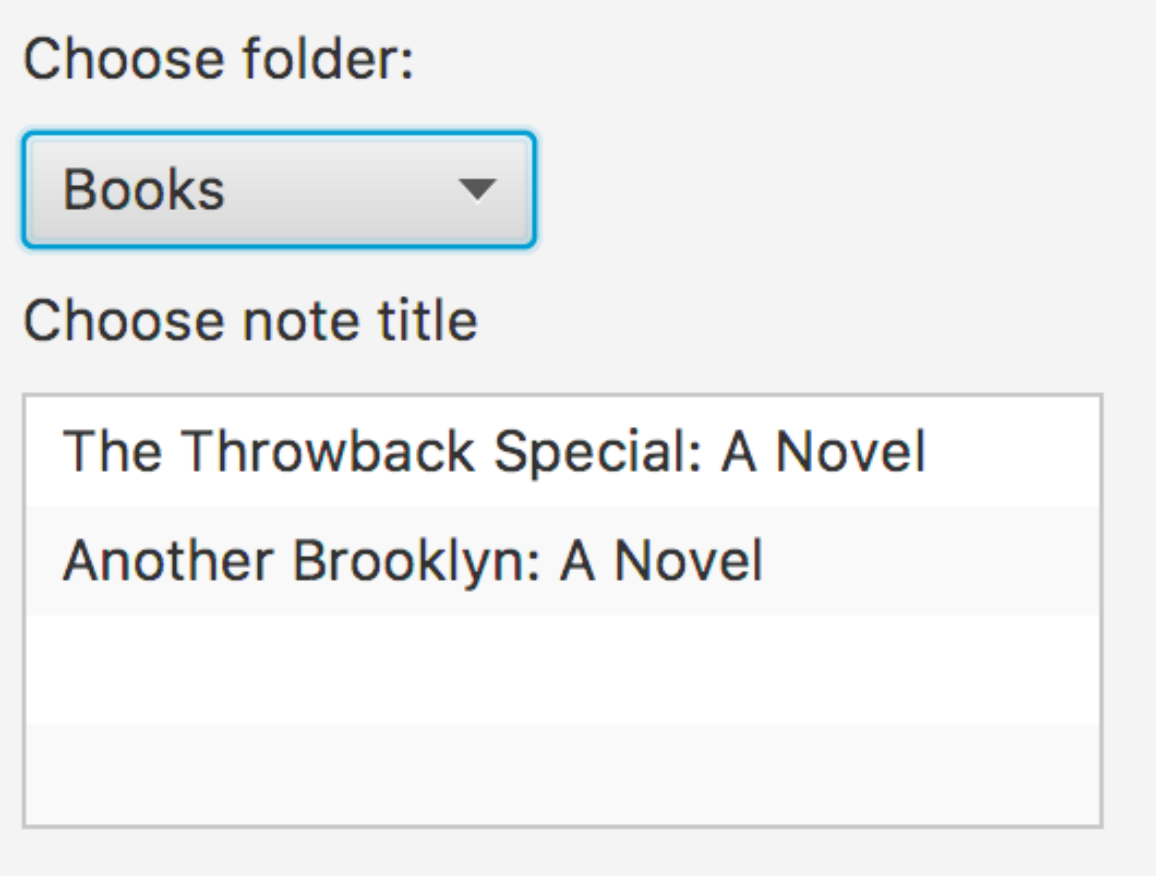
To achieve this replace this statement in the function **addVBox** with appropriate code

```
foldersComboBox.getItems().addAll("FOLDER NAME 1", "FOLDER NAME 2", "FOLDER NAME 3");
```

HINT: access the **noteBook** object and get all folder names.

- **TASK 3: Populate the titlesListView**

Once the user selects a folder from the **comboBox**, the **titlesListVIew** has to be updated with the titles of ALL TextNote objects in the selected folder



- **TASK 3: Populate the titlesListView**

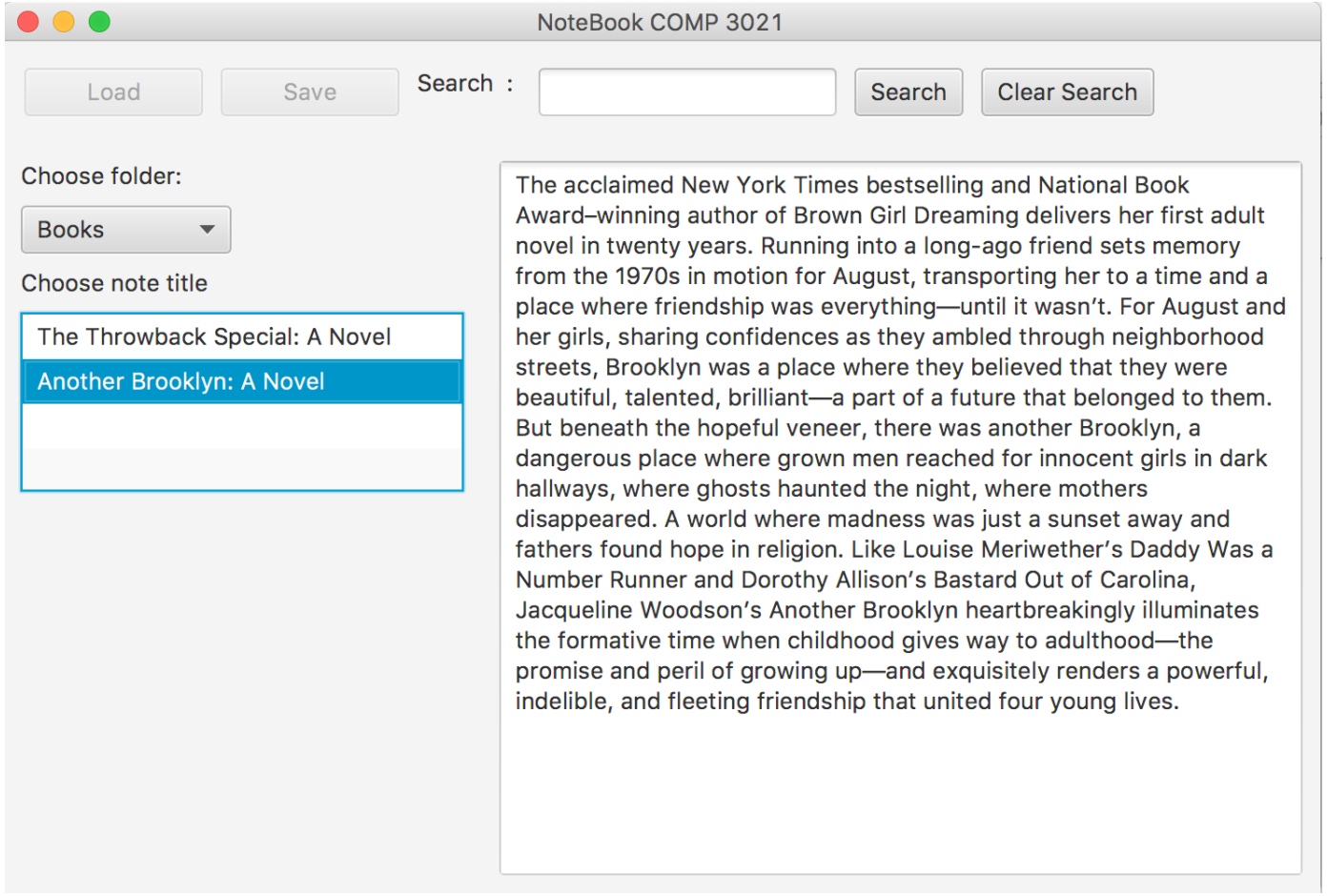
To achieve this, you have to add code to the function **updateListView**, invoked from the change listener of the foldersComboBox

This will be called every time the selection of foldersComboBox changes!

```
foldersComboBox.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Object>() {
                @Override
                public void changed(ObservableValue ov, Object t, Object t1) {
                        currentFolder = t1.toString();
                        // this contains the name of the folder selected
                        // TODO updatelistview
                        updateListView();

                }

        });
```

- **TASK 4: Populate the textAreaNote**

Once the user selects a Note title from the **titleslistView,** the TextArea **textAreaNote** has to be loaded with the content of the selected note



- **TASK 4: Populate the textAreaNote**

  To achieve, this complete the code invoked from the change listener of the titlelistView

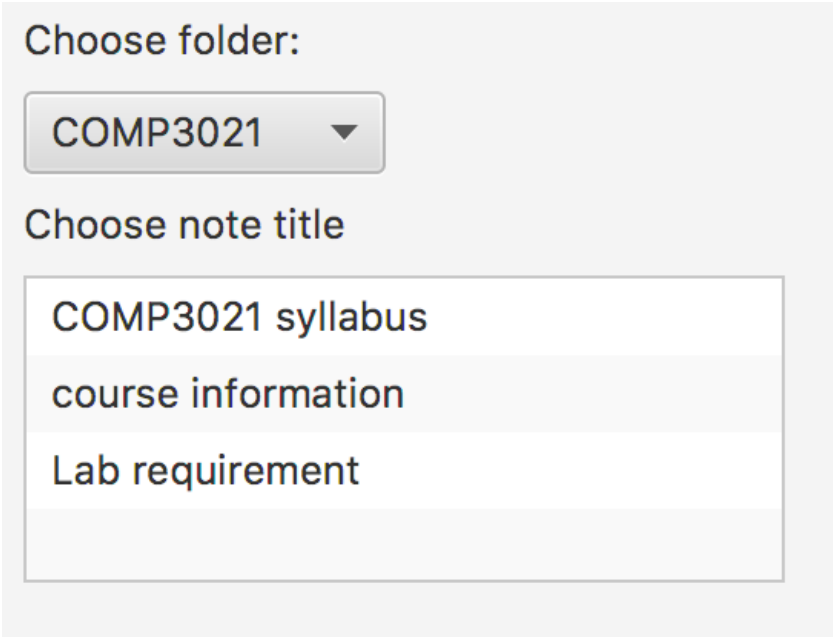
  This will be called every time the selection of titleslistView changes!

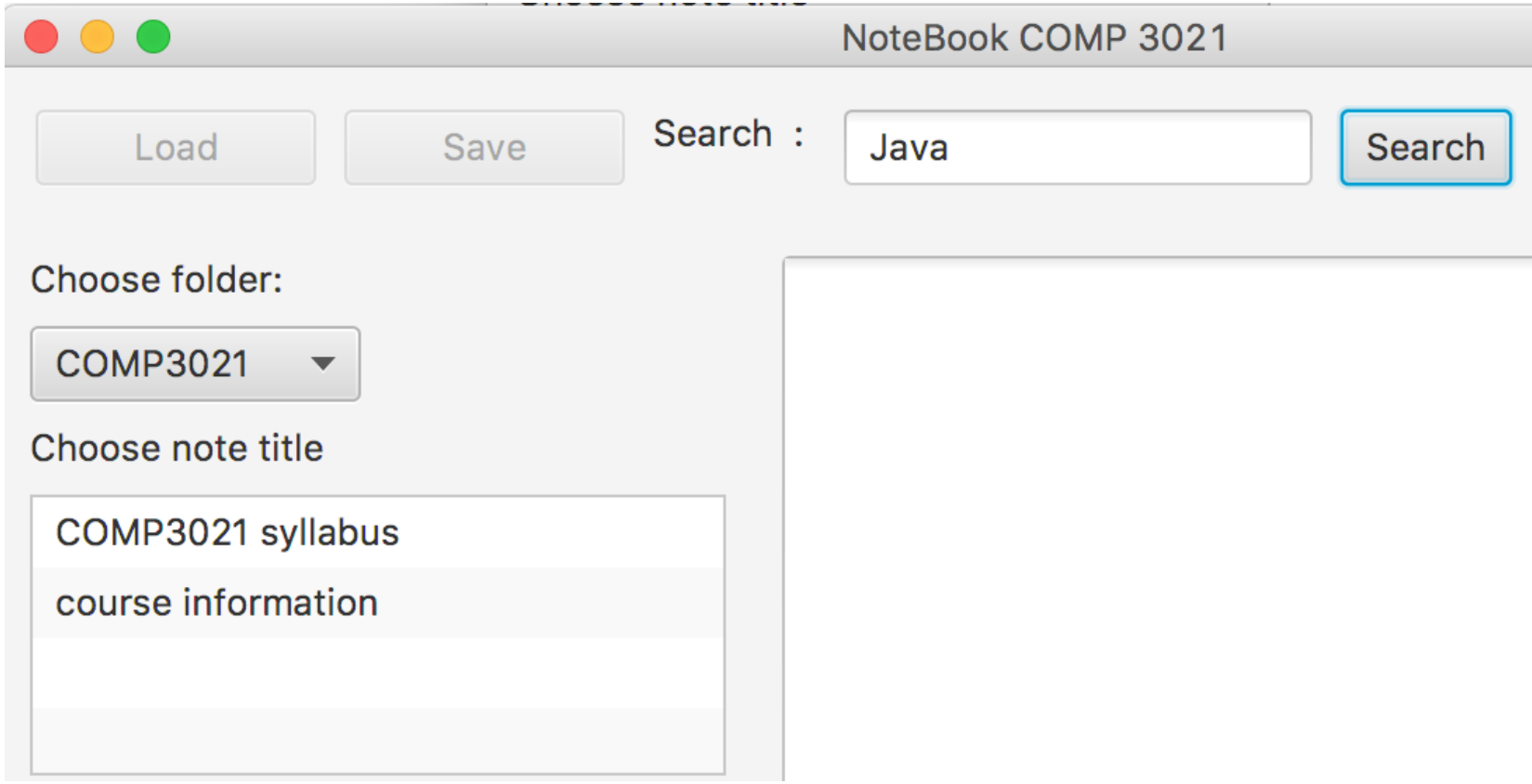
```
titleslistView.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Object>() {
                    @Override
                    public void changed(ObservableValue ov, Object t, Object t1) {
                        if  (t1  ==  null)
                            return;
                        String title = t1.toString();
                        // This is the selected title
                        // TODO load the content of the selected note in
                        // textAreNote
                        String content = "";
                        textAreaNote.setText(content);

                    }
            });
```

- **TASK 5: Invoke the search function that you implemented in LAB3**

If the user search ”Java” show in titleslistView ONLY those titles in the selected Folder that are returned by the function

```
List<Note> notes = note.searchNotes(currentSearch);
```

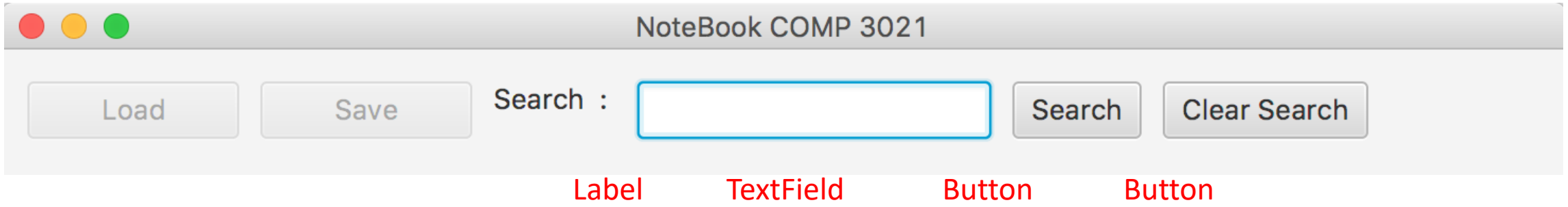




“Lab requirement” does not contain the word Java

- **TASK 5: Invoke the search function that you implemented in LAB3**

  Create one Label, TextField and two Button objects, and add them in the top pane in the function **addHBox**, as show below



  Remember to add them to the **hbox** object

```
hbox.getChildren().addAll(.........
```

- **TASK 5: Invoke the search function that you implemented in LAB3**

Add the listener for the click button of “Search”

```
Button buttonSearch = new Button("Search");
buttonSearch.setOnAction(new EventHandler<ActionEvent>() {
        @Override
public void handle(ActionEvent event) {
 currentSearch = textSearch.getText();
 textAreNote.setText("");
 TODO
        }
  });
```

HINT can we modify the method **updateListView** and invoke it?

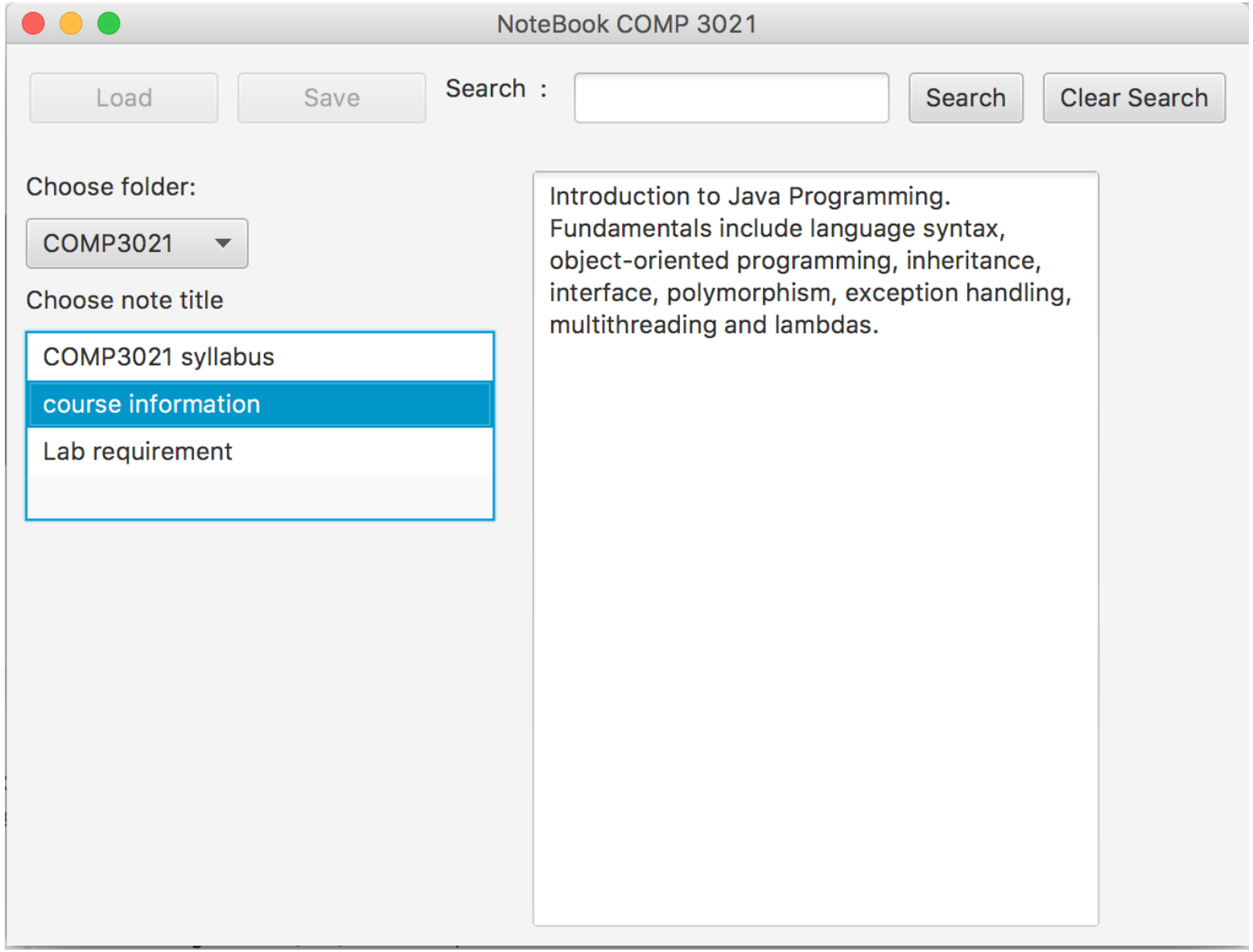
- **TASK 5: Invoke the search function that you implemented in LAB3**

Add the listener for the click button of “Clear Search”

```
Button buttonRemove = new Button("Clear Search");
buttonRemove.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent event) {
currentSearch  =  "";
textSearch.setText("");
textAreNote.setText("");
TODO
}
});
```

If a user clicks the button we want to shows ALL the titles of the selected folder!

- **Show a demo of your program to the TAs**



END OF LAB #7

Don’t forget to commit and push your code.

GitHub