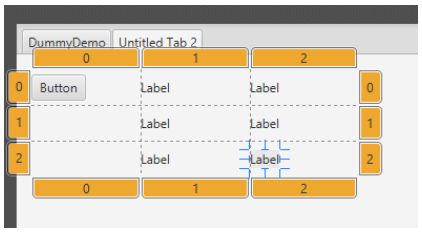
JAVAFX

1. **Set up the environment (If SceneBuilder is not installed)**
    a. Download the SceneBuilder installer file from the link: http://gluonhq.com/products/scene-builder/
    Unfortunately Oracle does not provide the latest versions' executables this is why the above link is to be used.
    b. Install SceneBuiler on the computer (In case of authentication problems try to use the .jar version although that cannot be bind to the NetBeans IDE). Memorize the install location.
    c. Start NetBeans and without creating a new project go to Tools->Options->Java->JavaFx. Set the Scene Builder Home location to the location you have specified during the installation. (Ok).
2. **Start a new JavaFX project**
    a. Download the start project from the page of the class
    b. Examine the project's structure and files.
3. **Refactor the starting project**
    a. Create a subpackage under the main package named model.
    b. Create another subpackage and name it view.
    c. Add a new Empty FXML file under the resources.fxml package. (Right click on the package -> New…->Other->JavaFX->Empty FXML). Name the file FXMLStudentsScene. During the creation of the file set "Use Java Controller" checkbox and use the default controller class name. Do not set a formatting file.
    d. Change the body of the MainApp#start method to the following:
```
FXMLLoader loader = new FXMLLoader(MainApp.class.getResource("/fxml/FXMLStudentsScene.fxml"));
Scene scene = new Scene(loader.load());
stage.setTitle("Students Register");
stage.setScene(scene);
stage.show();
```
    e. Double click (or right click->Open) on the FXMLStudentsScene.fxml file. In Scene Builder drag a new Button to the Middle of the window. Run the App…
4. **Make a button work**
    a. Open the FXMLStudentsSceneController.java file and add a new method to it named handleButtonPushed(). Make this method print out "Button pushed" to the standard output.
    b. Use the @FXML annotation on the method (so that you can link it to the Button in Scene Builder)
    c. Go back to Scene Builder. Select the button and at the right pane expand the "Code : Button" option. Set the "On action" field to "handleButtonPushed". Save everything and run the App..
5. **Make a button interact with a label**
    a. In SceneBuilder add a new label to the Scene.
    b. Open the FXMLStudentsSceneController.java and add a new field of type "Label" and name "helloLabel" to the class. Set it private and use the @FXML annotation on it.
    c. Add the following line to the handleButtonPushed method:
```
hellolabel.setText("Button pushed");
```
    d. Go back to Scene Builder and select the label. Set the fx:id property (under Code : Label at the right side) to helloLabel

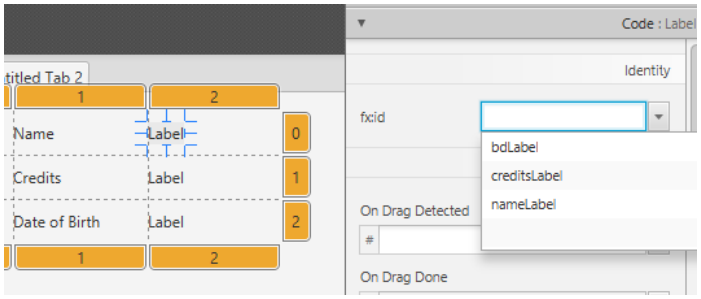
6. **Load data from the model**
    a. Go to Scene Builder and reorganize the Scene. Add a TabPane to the root AnchorPane, and move the previously created Button and Label to the first tab. Name this tab DummyDemo
    b. Select the other tab and name it LoadDemo.
    c. Right Click on the TabPane in the left menu and select Fit to Parent.
    d. Add a GridPane to the LoadDemoTab and fill it according to this picture (you can add a new column, by right clicking on the column header):



    e. Set the text of the Button to “Load” and the texts of the labels in the first column “Name”, “Credits”, and “Date of birth” respectively from the top to the bottom.
    f. Add three new Label fields to the FXMLStudentsSceneController named “nameLabel”, “creditsLabel” and “bdLabel”. (Use the @FXML annotation here too)

    g. In the model package create a class that can represent a Student with three properties: name, credits, and date of birth. Generate getters setters and a constructor to set all there fields.
    h. Add a new class named Model into the model package. Make this class have a Student typed field. Generate getters and setters for this field. Add a default constructor to the class that sets the fields value to a new Student: (“Robert Smith”, 55, “1991-12-03”)
    i. Add a new field to StudentsSceneController named model (type: Model), and add a setter for this field
    j. In the start method of the MainApp class add `((FXMLStudentsSceneController)loader.getController()).setModel(new Model());` after initializing the Scene.

    k. Now that the Model and the view are connected you can add the handleLoad method to the StudentsSceneController class that sets the respective labels to the values read from the model’s student field.
    l. Do not forget to set the fx:id properties of the labels in the third column in the gridPane to the @FXML annotated labels…



    m. Set the “On Action” property of the Load button to “handleLoad”
    n. Save everything and run the App…

7. **What if data changes?**
    a. Add a new button to the Tab and name it “Change Name”.
    b. Add a handler method to the StudentsSceneController class that changes the name off the student of the model to “Tom Smith”
    c. Connect the button and the method and run the App. (If you did it well the name does not changes.)
    d. Modify the Student class so that it uses a StringProperty field for the name instead of the simple String. (Tip: use ALT+INSERT to add a new JavaFX Property). Handle the appearing errors…
    e. In the handleLoad method of the StudentsSceneController class change the line that sets the name label to the following:
    ```
    nameLabel.textProperty().bind(model.getStudent().nameProperty());
    ```
    f. Run the App. (Now if you push the “Change name” button, the name label automatically updates.)
8. **How to add dialogs?**
    a. Add the following lines to the handleChangeName method:
    ```
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Button pushed");
    alert.setHeaderText("You have pushed the Change Name button");
    alert.setContentText("This shows that you have pushed a button...");
    alert.showAndWait();
    ```
    b. More about dialogs: http://code.makery.ch/blog/javafx-dialogs-official/

A much more detailed JavaFX tutorial: http://code.makery.ch/library/javafx-8-tutorial/

Seralization: https://www.tutorialspoint.com/java/java_serialization.htm