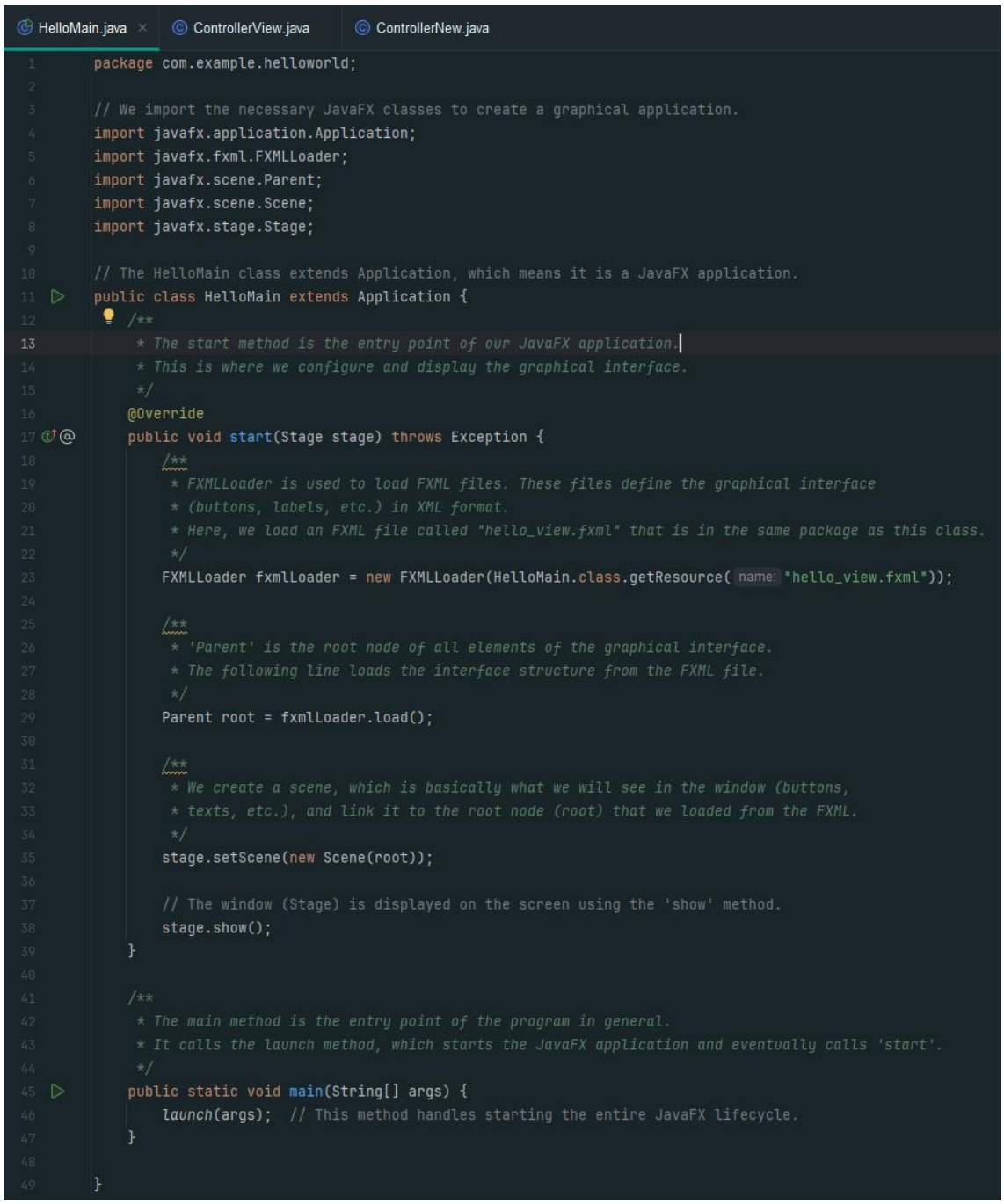
Solucion HelloWorld

1. [2/10]. After having been in class and reviewing the presentations, you should comment on the HelloMain.java, ControllerView.java and ControllerNew.java classes, which are located within src\main\java\com.example.helloworld. Put as many comments as you think appropriate so that you explain what each class is for, what each created object is and what each function does.



HelloMain.java | ControllerView.java | ControllerNew.java

```java
package com.example.helloworld;

// We import the necessary JavaFX classes to create a graphical application.
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

// The HelloMain class extends Application, which means it is a JavaFX application.
public class HelloMain extends Application {
    /**
     * The start method is the entry point of our JavaFX application.
     * This is where we configure and display the graphical interface.
     */
    @Override
    public void start(Stage stage) throws Exception {
        /**
         * FXMLLoader is used to load FXML files. These files define the graphical interface
         * (buttons, labels, etc.) in XML format.
         * Here, we load an FXML file called "hello_view.fxml" that is in the same package as this class.
         */
        FXMLLoader fxmlLoader = new FXMLLoader(HelloMain.class.getResource( name: "hello_view.fxml"));

        /**
         * 'Parent' is the root node of all elements of the graphical interface.
         * The following line loads the interface structure from the FXML file.
         */
        Parent root = fxmlLoader.load();

        /**
         * We create a scene, which is basically what we will see in the window (buttons,
         * texts, etc.), and link it to the root node (root) that we loaded from the FXML.
         */
        stage.setScene(new Scene(root));

        // The window (Stage) is displayed on the screen using the 'show' method.
        stage.show();
    }

    /**
     * The main method is the entry point of the program in general.
     * It calls the launch method, which starts the JavaFX application and eventually calls 'start'.
     */
    public static void main(String[] args) {
        launch(args);  // This method handles starting the entire JavaFX lifecycle.
    }

}
```

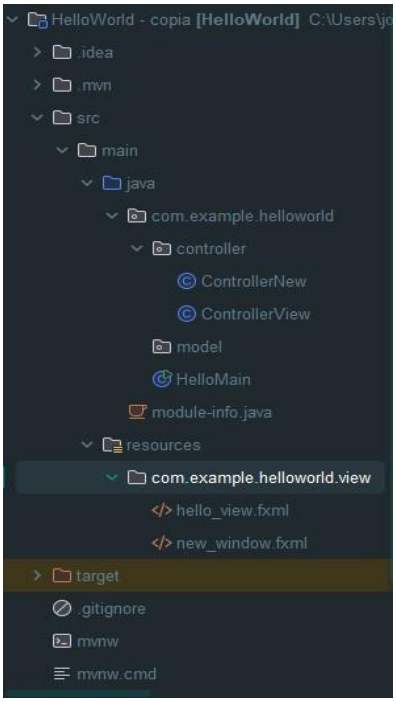
HelloMain.java | ControllerView.java | ControllerNew.java

```java
package com.example.helloworld;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.stage.Modality;
import javafx.stage.Stage;

import java.io.IOException;

public class ControllerView {

    /**
     * The button defined in the FXML file, associated with an ID of 'btn'.
     * When the user clicks this button, the 'showText' method is triggered.
     */
    @FXML
    private Button btn;

    /**
     * Label that can display some text in the user interface.
     * It is linked with an ID of 'lbl' in the FXML file.
     */
    @FXML  no usages
    private Label lbl;

    /**
     * TextField where the user can enter text.
     * The value of this field is obtained in the 'showText' method.
     */
    @FXML
    private TextField txt;

    /**
     * This method is triggered when the button is clicked. Its function is to open a new window.
     * First, it gets the text from the 'txt' text field, then it loads a new FXML file
     * called "new_window.fxml", and finally, shows that new window.
     */
    @FXML
    void showText(ActionEvent event) throws IOException {
        // Gets the text entered by the user in the 'txt' text field.
        String text = txt.getText();

        // We load the new view from the "new_window.fxml" file.
        FXMLLoader fxmlLoader = new FXMLLoader(HelloMain.class.getResource( name: "new_window.fxml"));

        // We load the FXML file and create the interface structure.
        Parent root = fxmlLoader.load();

        // We obtain the controller of the new window to pass the text to it.
        ControllerNew controller = fxmlLoader.getController();

        // We pass the text from the 'txt' text field to the controller of the new window.
        controller.seeText(text);

        // We create a new window (Stage) to show the new scene.
        Stage stage = new Stage();

        // We set the new scene that we loaded from the FXML.
        stage.setScene(new Scene(root));

        // We set the modality of the new window to make it modal and block the previous window.
        stage.initModality(Modality.WINDOW_MODAL);

        // We set the window that launched the event as the owner of this new window.
        stage.initOwner(((Node) (event.getSource())).getScene().getWindow());

        // We show the new window on the screen.
        stage.show();
    }
}
```

HelloMain.java | ControllerView.java | ControllerNew.java

```java
package com.example.helloworld;

import javafx.fxml.FXML;
import javafx.scene.control.Label;

public class ControllerNew {

    /**
     * Label that can display some text in the user interface.
     * It is linked with an ID of 'lbl' in the FXML file.
     */
    @FXML
    private Label lbl;

    /**
     * This method is used to update the text displayed in the 'lbl' label.
     * It receives a string message and sets it as the text of the 'lbl' label.
     */
    @FXML  1 usage
    public void seeText(String message) {
        // Set the text of the 'lbl' label to the provided message.
        lbl.setText(message);
    }

}
```

2. [3/10]. Starting from the example project, create a copy of it and make the appropriate changes so that:
   a. The project is organized with the Model – View – Controller.



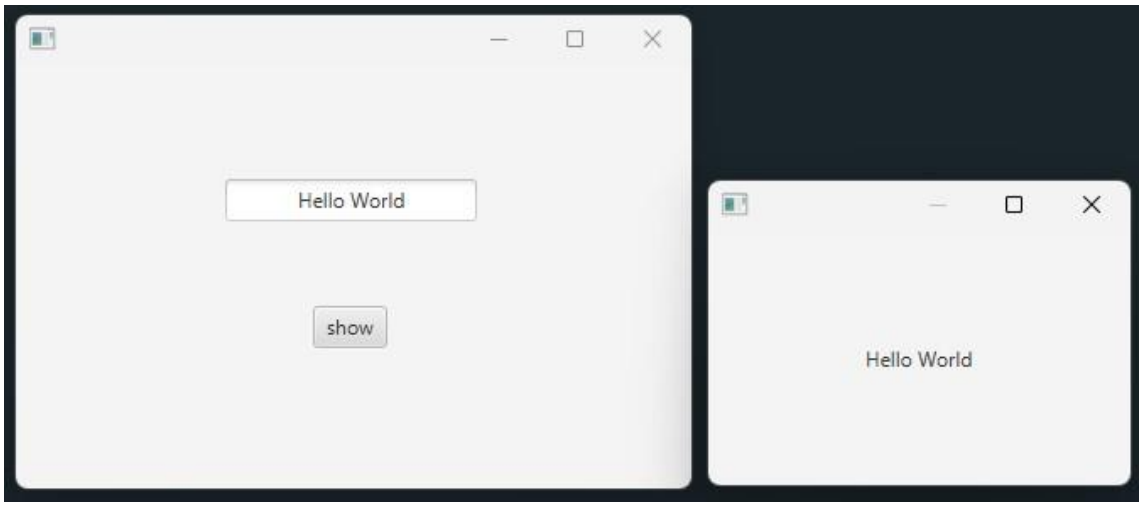
Rename main path so that it finds the hello_view and controller view so that it finds new_window

```
FXMLLoader fxmlLoader = new FXMLLoader(HelloMain.class.getResource( name: "view/hello_view.fxml"));
```

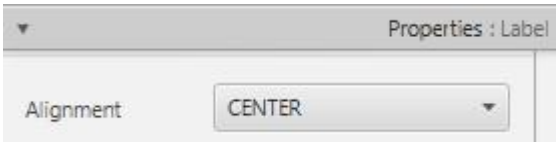
```
FXMLLoader fxmlLoader = new FXMLLoader(HelloMain.class.getResource( name: "view/new_window.fxml"));
```

b. All hello_view and new_window elements are centered. Also, rename both fxml files to my_view and my_new_Window

Center



Center the text showing the my_new_window label



Rename

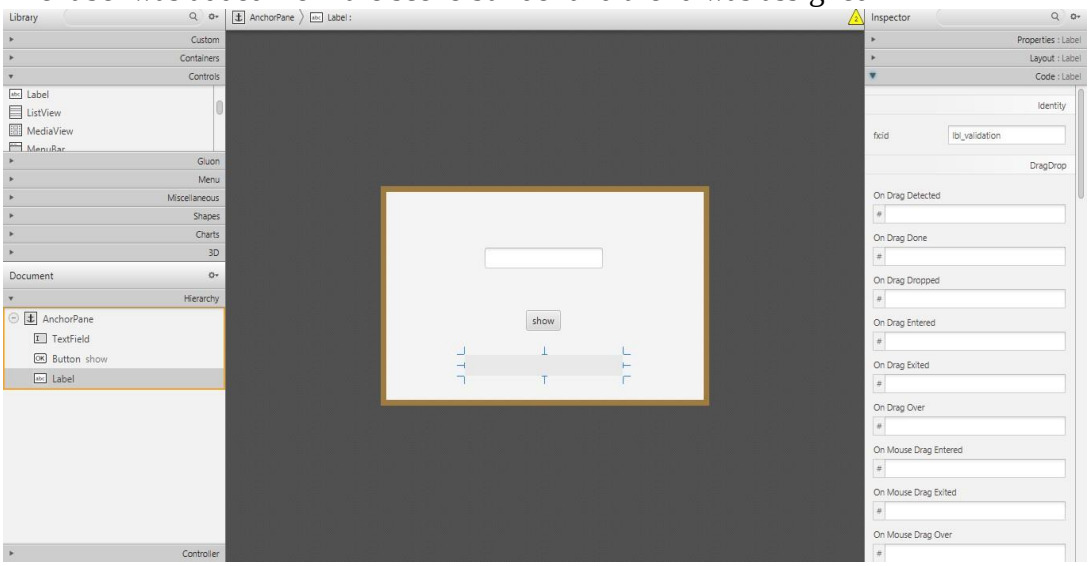
Update the controller and main where we pass the routes

```
FXMLLoader fxmlLoader = new FXMLLoader(HelloMain.class.getResource( name: "view/my_view.fxml"));
```

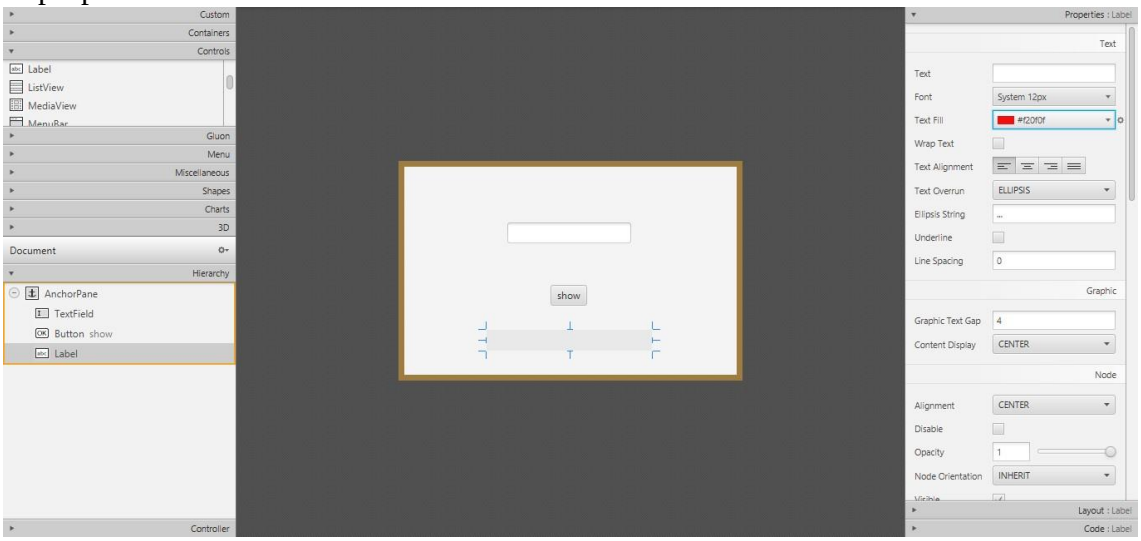
```
FXMLLoader fxmlLoader = new FXMLLoader(HelloMain.class.getResource( name: "view/my_new_window.fxml"));
```

In ControlallerView.java, a validation is done on the text variable so that, if it is empty (""), instead of opening my_new_window, it writes a message in a label (which you must add to the window) that says: “Please write something' and it will be displayed in red.

The label was added from the scene builder and the id was assigned



In properties it is set to show the text in red



Controller code

```java
public class ControllerView {

    @FXML
    private Button btn;

    @FXML  no usages
    private Label lbl;

    @FXML
    private Label lbl_validation;

    @FXML
    private TextField txt;

    @FXML
    void showText(ActionEvent event) throws IOException {
        String text = txt.getText();

        if (text.isEmpty()) {
            // If the text is empty, it displays the message in the Label
            lbl_validation.setText("Please write something");
        } else {
            lbl_validation.setText("");

            // If the text is not empty, it loads the new window
            FXMLLoader fxmlLoader = new FXMLLoader(HelloMain.class.getResource( name: "view/my_new_window.fxml")
            Parent root = fxmlLoader.load();

            ControllerNew controller = fxmlLoader.getController();
            controller.seeText(text);

            Stage stage = new Stage();
            stage.setScene(new Scene(root));
            stage.initModality(Modality.WINDOW_MODAL);
            stage.initOwner(((Node) (event.getSource()))).getScene().getWindow());
            stage.show();
        }
    }


}
```