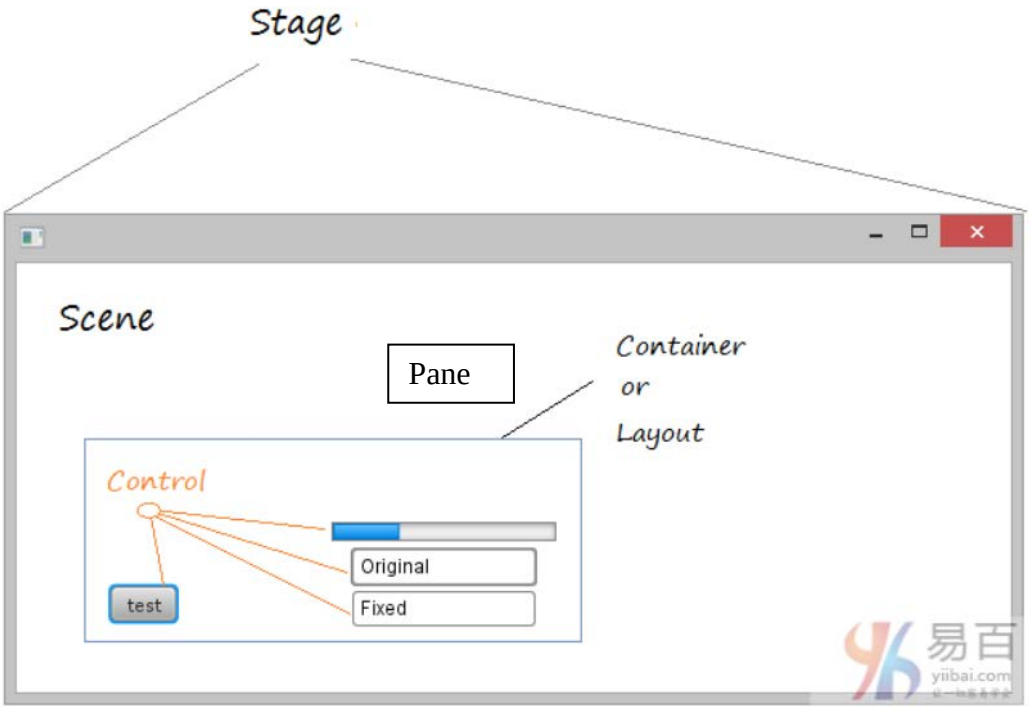
Java2 Lab 9
(JavaFx)

[Experimental Objective]

1. Learn to understand Swing
2. Master the basic method of JAVA graphical interface design by writing and debugging programs

[Javafx composition]

The following image shows the relationship between the Stage, the Scene, the Container, the Layout, and the Controls in
JavaFx:



Javafx some simple class names

1, Scene: create a scene
2, Stage: create a stage
3, Pane: the base class of the panel
4, stackPane: panel, the node is placed in the center of the panel
5, FlowPane: nodes are placed line by line, or column by column
6, GridPane: the node is placed in a cell of a two-dimensional network
7, BorderPane: the node is placed around and central
8, HBox: nodes are placed in a single line
9, VBox: nodes are placed in a single column
10, Color: define the color
11, Font: define the font size, format, thickness, etc.

[Javafx install]

Install **e(fx)clipse:**

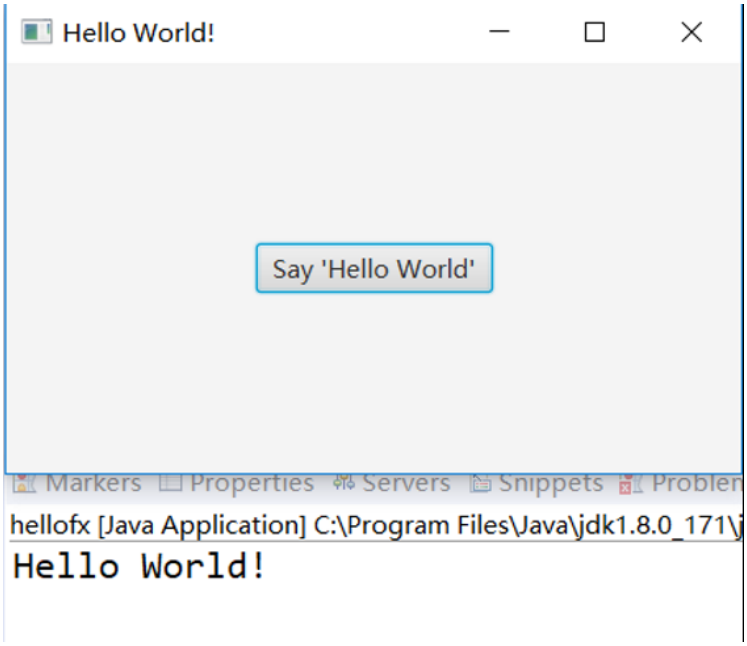
See tutorial:
https://www.yiibai.com/javafx/install-efxclipse-into-eclipse.html

input the text in 'install new software ':

Name: e(fx)clipse
Location: http://download.eclipse.org/efxclipse/updates-released/2.3.0/site

[Exercise]

Exercise 1:
Refer to Exercise 1 of lab8, write the hello world program using JavaFx, and display the window and button contents as hello world. Click the button to output hello world in the console.



prompt:
Set the button event and assemble the scene, button, stage and other parts in turn.

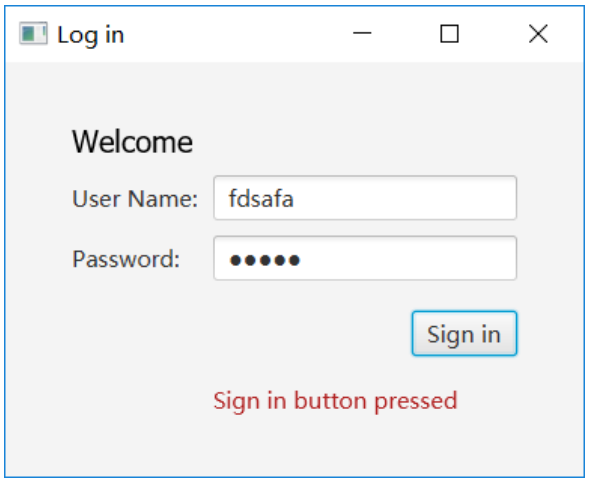
```
    public void start(Stage primaryStage) {
        try {
            Button button = new Button();
            button.setText("hello world");
            button.setOnAction(new EventHandler<ActionEvent>() {
                @Override
                public void handle(ActionEvent event) {
                    // TODO Auto-generated method stub
                    System.out.println("hello world");
                }
            });
            StackPane stackPane = new StackPane();
            stackPane.getChildren().add(button);
            Scene scene = new Scene(stackPane, 350, 200);

            primaryStage.setTitle("hello world");
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Note: StackPane: It is a card layout similar to the previous CardLayout, that is, the content behind it will be displayed on the previous content.

Exercise 2:



Idea hint:

On the basis of Exercise 1, follow the idea of the last lab8 experiment 2, put the text, label, textfield, password field and other components into the GridPane (because the GridPane can create a two-dimensional row and column grid), then the gridPane After resizing, add it to the scene and set the scene to stage.

```java
@Override
public void start(Stage primaryStage) {
        primaryStage.setTitle("Log in");

        GridPane grid = new GridPane();
        grid.setAlignment(Pos.CENTER);
        grid.setHgap(10);
        grid.setVgap(10);
        grid.setPadding(new Insets(25, 25, 25, 25));

        placeComponents(grid);

        Scene scene = new Scene(grid, 400, 375);
        primaryStage.setScene(scene);
        scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
        primaryStage.show();
}
```

Place other components in turn into the GridPane:

```java
public static void placeComponents(GridPane grid) {
    Text text = new Text("Welcome");
    text.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
    grid.add(text, 0, 0, 2, 1);

    Label userName = new Label("User Name:");
    grid.add(userName, 0, 1);
    TextField userTextField = new TextField();
    grid.add(userTextField, 1, 1);

    Label pw = new Label("Password:");
    grid.add(pw, 0, 2);
    PasswordField pwBox = new PasswordField();
    grid.add(pwBox, 1, 2);

    Button btn = new Button("Sign in");
    HBox hbBtn = new HBox(10);
    hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
    hbBtn.getChildren().add(btn);
    grid.add(hbBtn, 1, 4);

    final Text actiontarget = new Text();
    grid.add(actiontarget, 1, 6);

        btn.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent e) {
                actiontarget.setFill(Color.FIREBRICK);
                actiontarget.setText("Sign in button pressed");
            }
        });


    }

    public static void main(String[] args) {
        launch(args);
    }
```

[Optional] Exercise 3:

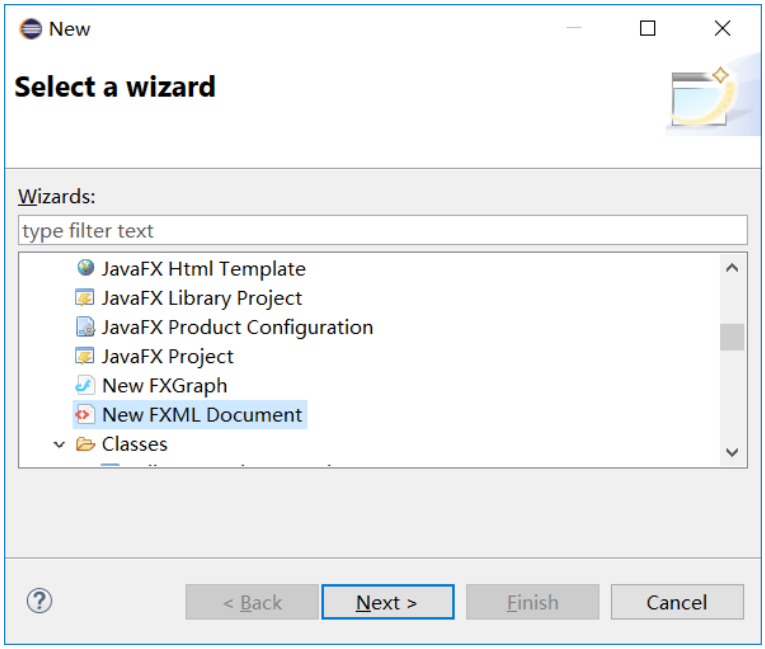
On the basis of Experiment 2, install JavaFX Scene Builder, implement Experiment 2 in graphical coding, and compare the advantages and disadvantages of graphical coding and coding.

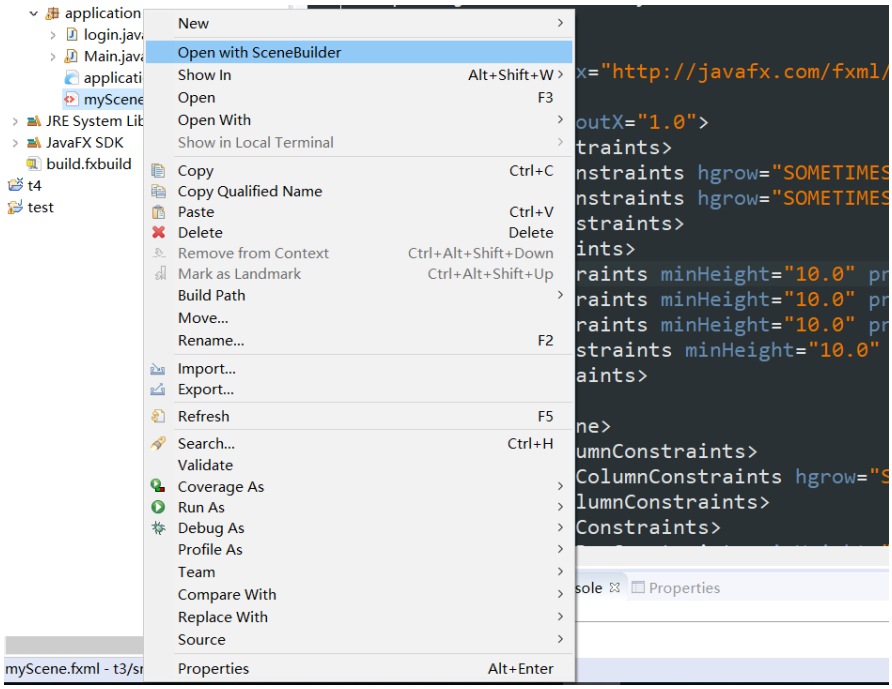
See the tutorial for specific installation steps:

https://www.yiibai.com/javafx/install-javafx-scene-builder-into-eclip

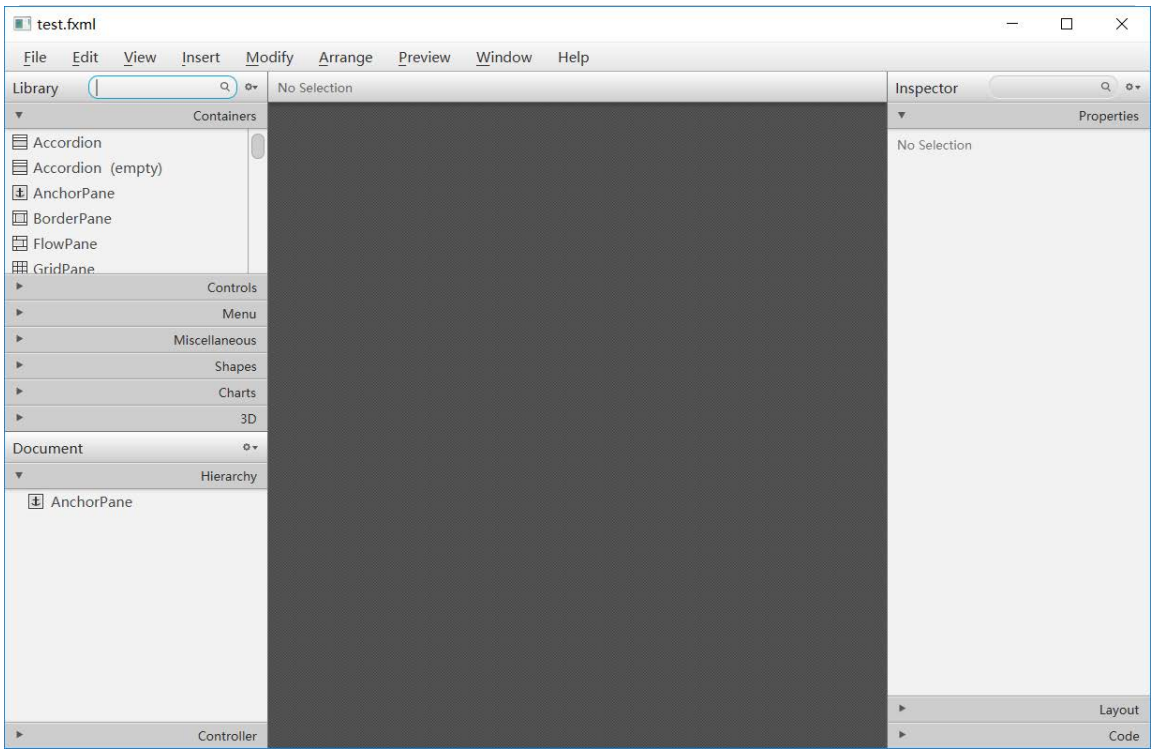
se.html

Create a new fxml document in the javaFx project, and right-click on Open with Scene Builder in the fxml file. This will open the graphical editing interface.





Simply add each component in turn and assemble it.



11849062@mail.sustc.edu.cn