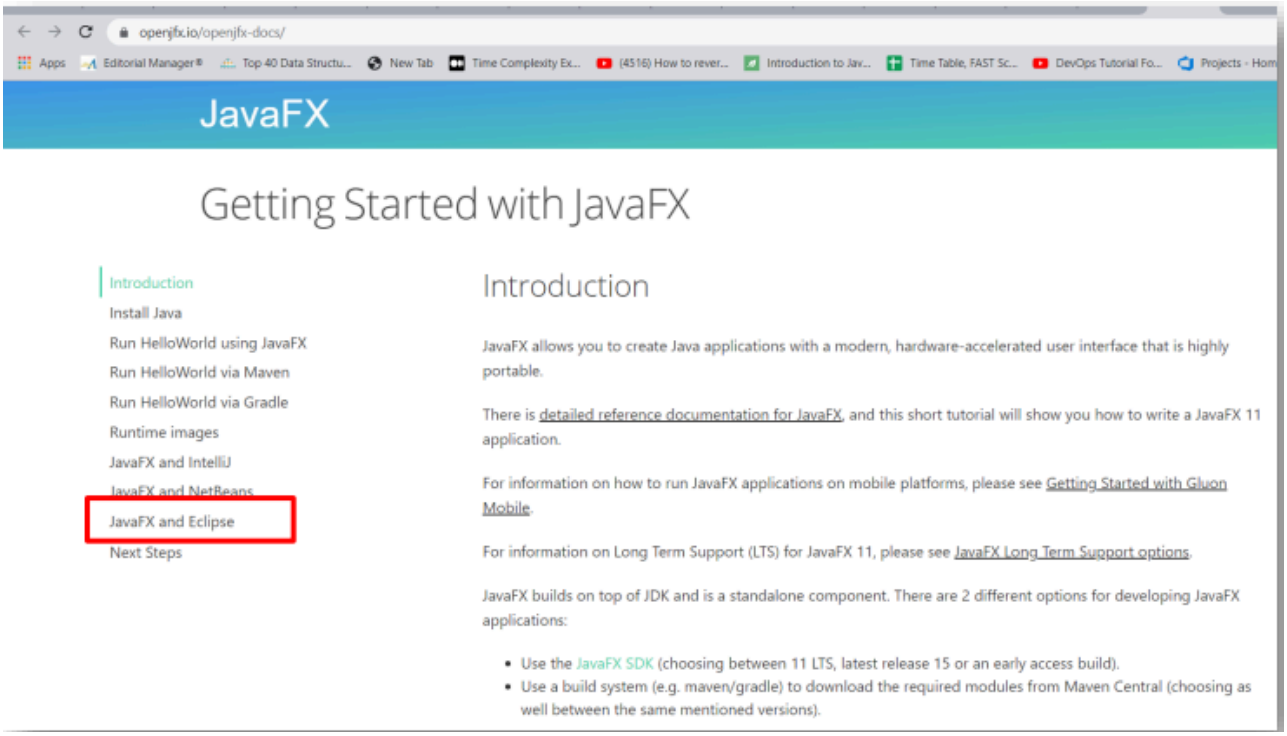
Fast-NU-ISB SPRING 2022

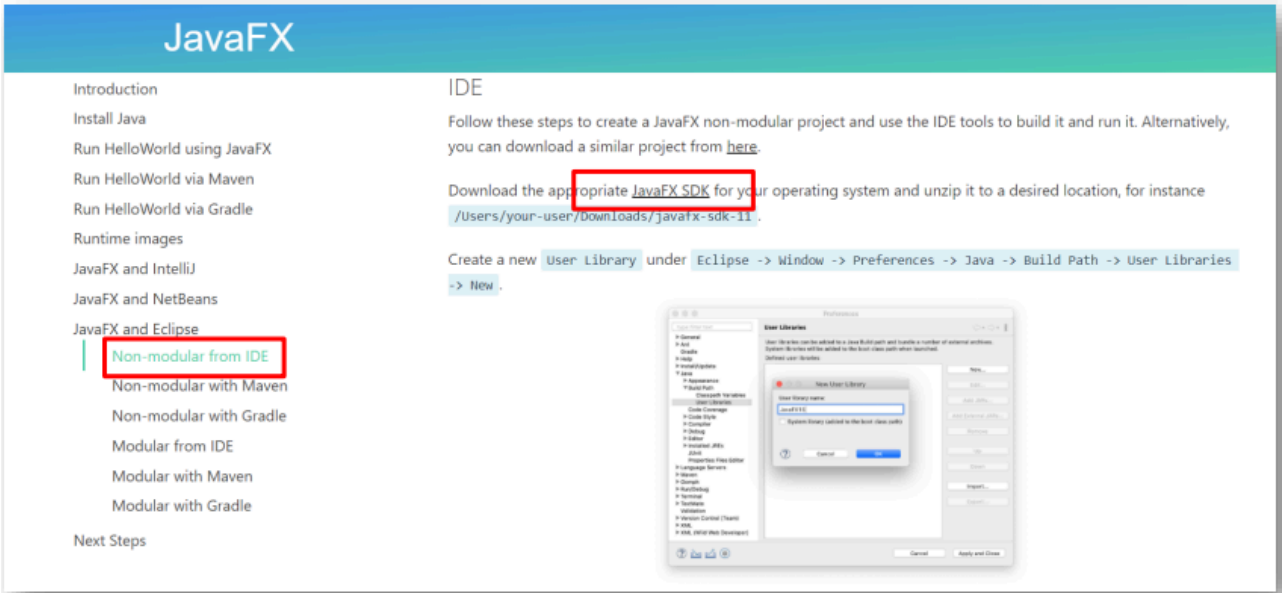
Lab 02–GUI with JavaFX and Event Handling

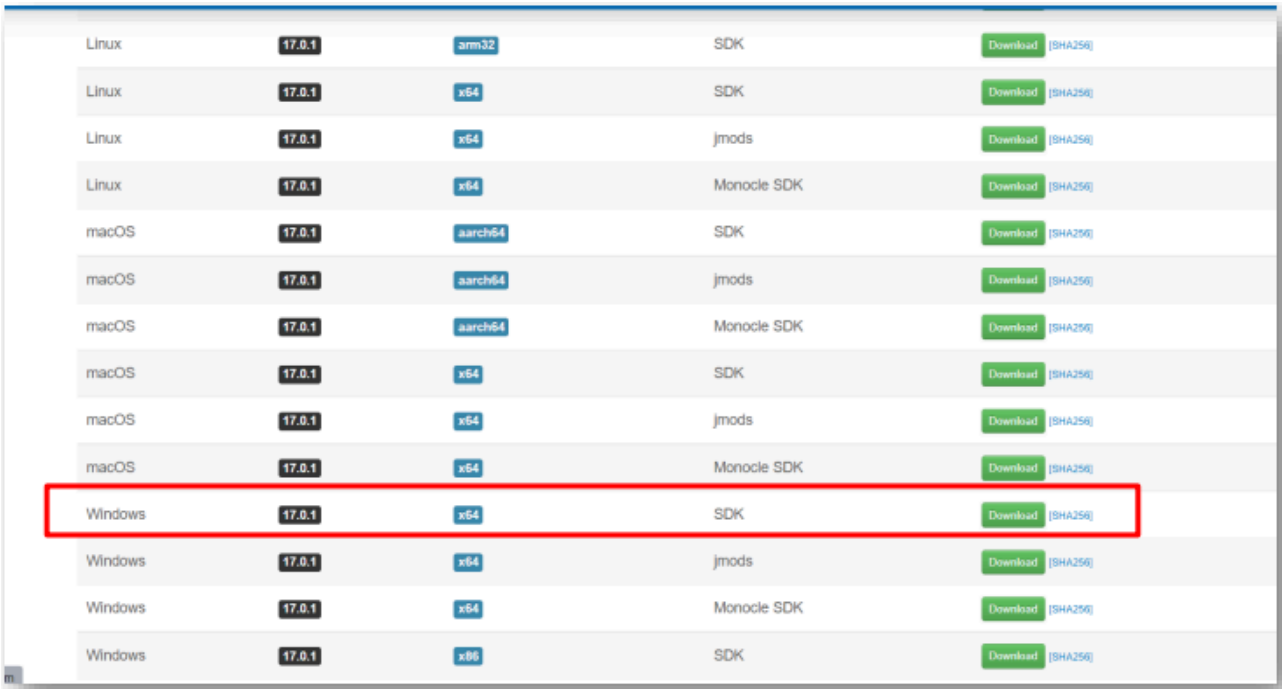
JavaFX is the latest graphical user interface framework. It is a platform for making amazing GUI application.

The applications developed using JavaFX can run on various devices such as Desktop Computers, Mobile Phones, TVs, Tablets, etc.

Step#1: Download JavaFX SDK





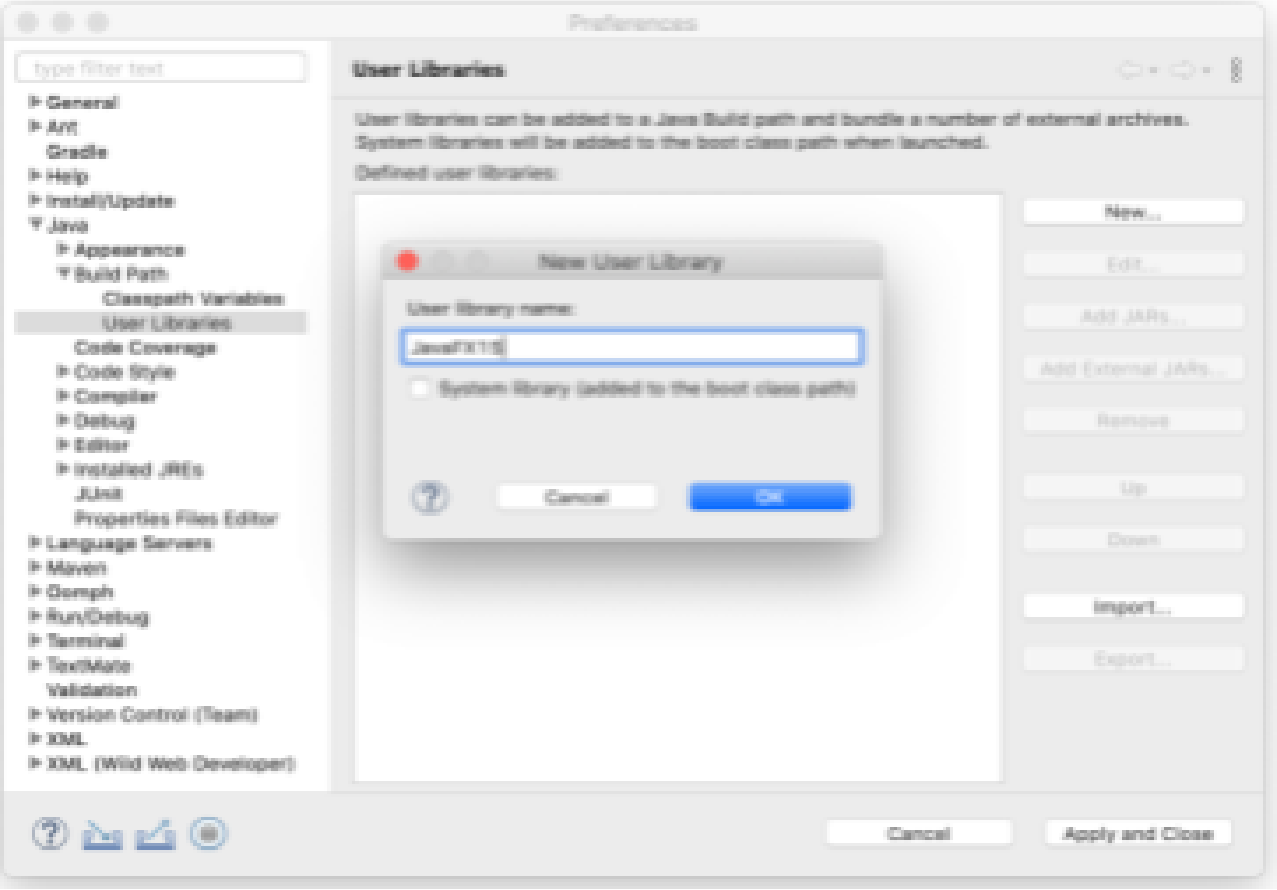


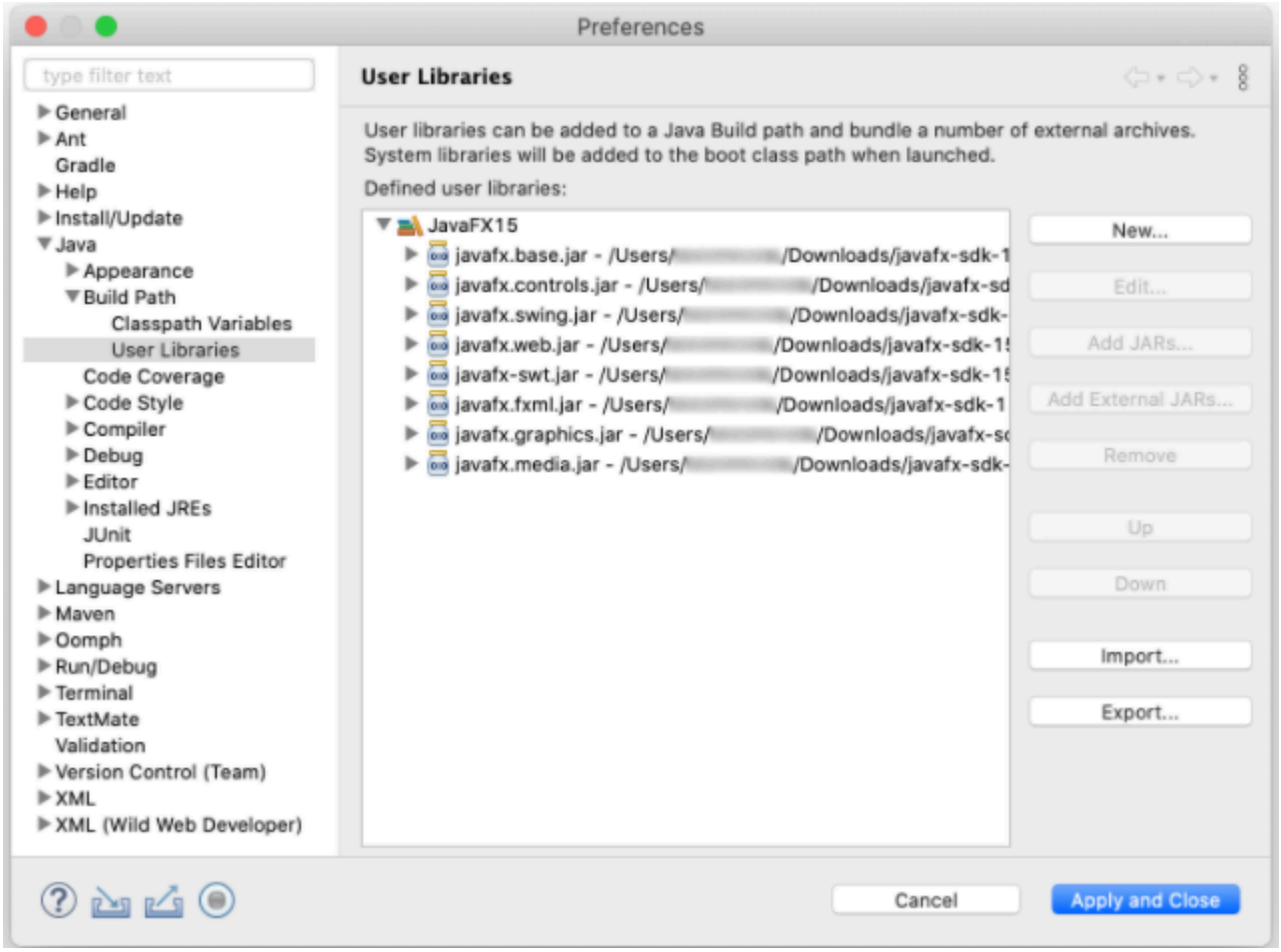
Linux	17.0.1	arm32	SDK	Download [SHA256]
Linux	17.0.1	x64	SDK	Download [SHA256]
Linux	17.0.1	x64	jmods	Download [SHA256]
Linux	17.0.1	x64	Monocle SDK	Download [SHA256]
macOS	17.0.1	aarch64	SDK	Download [SHA256]
macOS	17.0.1	aarch64	jmods	Download [SHA256]
macOS	17.0.1	aarch64	Monocle SDK	Download [SHA256]
macOS	17.0.1	x64	SDK	Download [SHA256]
macOS	17.0.1	x64	jmods	Download [SHA256]
macOS	17.0.1	x64	Monocle SDK	Download [SHA256]
Windows	17.0.1	x64	SDK	Download [SHA256]
Windows	17.0.1	x64	jmods	Download [SHA256]
Windows	17.0.1	x64	Monocle SDK	Download [SHA256]
Windows	17.0.1	x86	SDK	Download [SHA256]

Step#2: Open Eclipse and create a new `User Library` under

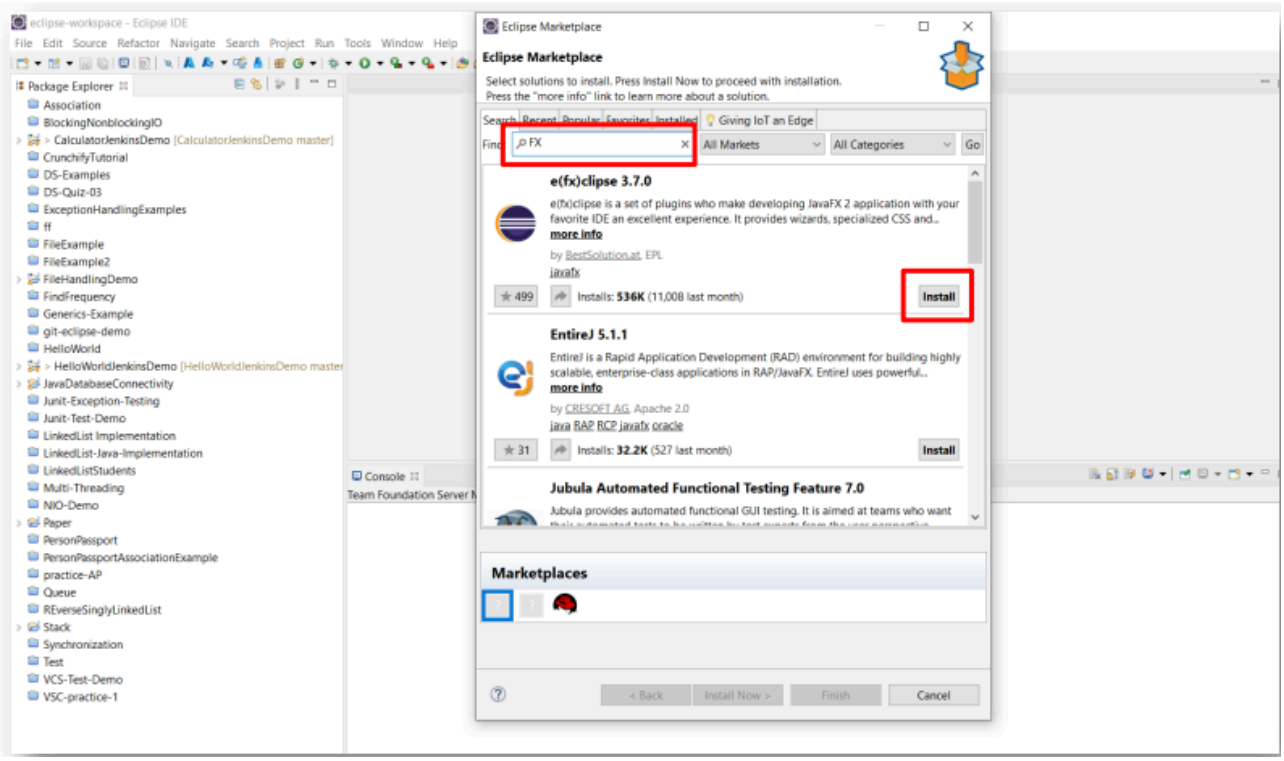
```
Window -> Preferences -> Java -> Build Path -> User Libraries -> New
```

Name it `JavaFX17` and Add External JARS under the `lib` folder from JavaFX 17.


Preferences
type filter text
General
Ant
Gradle
Help
Install/Update
Java
Appearance
Build Path
Classpath Variables
User Libraries
Code Coverage
Code Style
Compiler
Debug
Editor
Installed JREs
JUnit
Properties Files Editor
Language Servers
Maven
Oomph
Run/Debug
Terminal
TextMate
Validation
Version Control (Team)
XML
XML (Wild Web Developer)
User Libraries
User libraries can be added to a Java Build path and bundle a number of external archives.
System libraries will be added to the boot class path when launched.
Defined user libraries:
New User Library
User library name:
JavaFX11S
System library (added to the boot class path)
Cancel
OK
New...
Edit...
Add JARs...
Add External JARs...
Remove
Up
Down
Import...
Export...
Cancel
Apply and Close

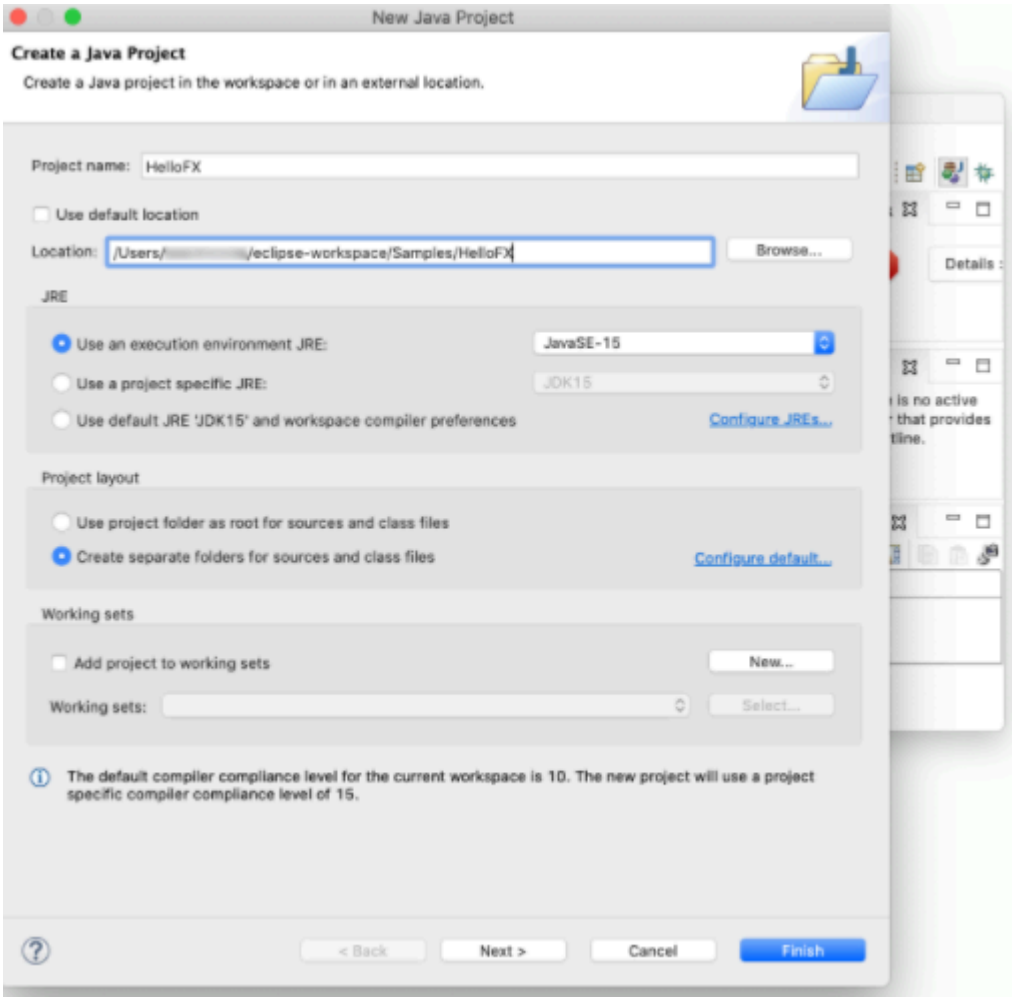


Step#3: Install JavaFX



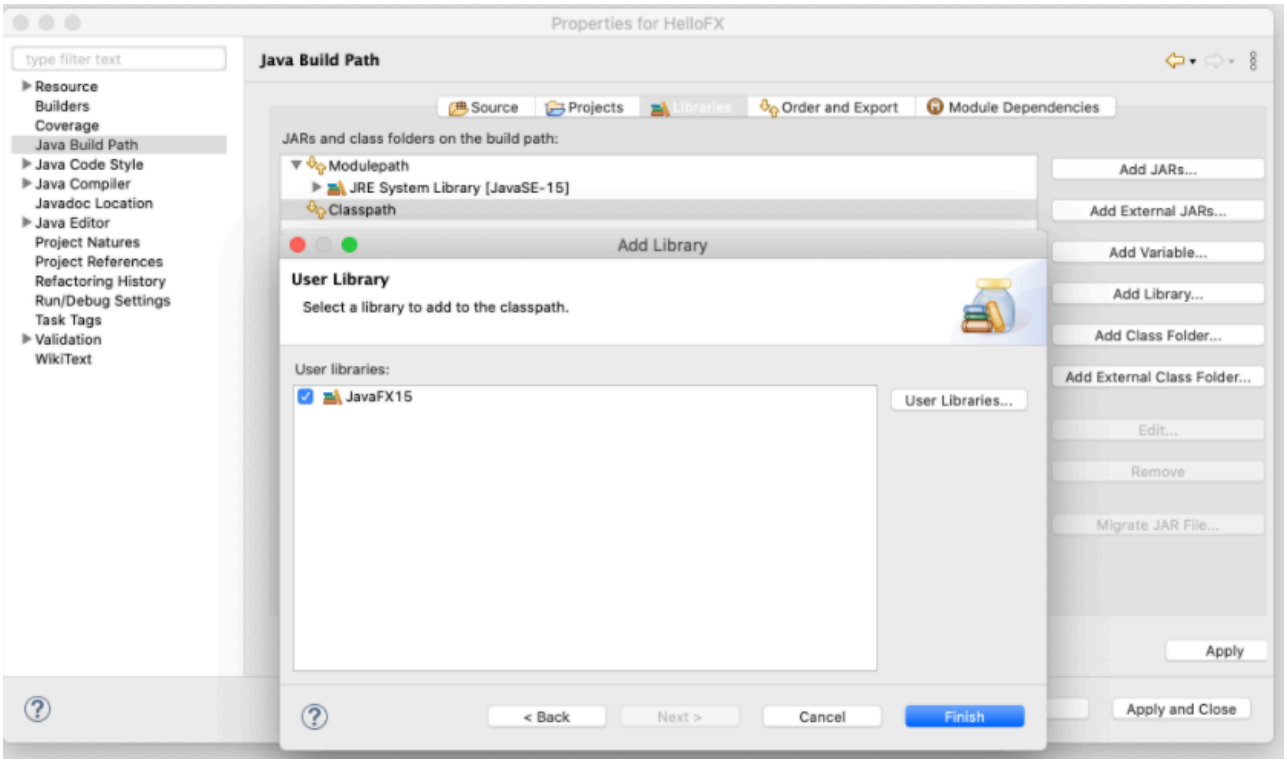
Create a new Java Project

Select `File -> New -> Java Project`, and provide a name to the project, like `HelloFX`, and a location.



Add the `JavaFX17` library into the classpath.

Right click on project -> Build Path -> Add Library -> User Library -> Next ->JavaFX17



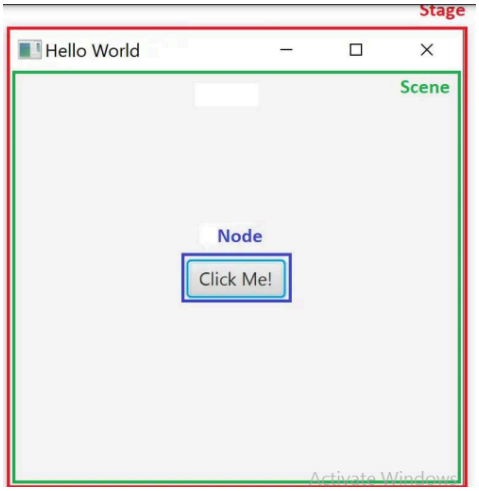
Click apply and close the dialog.

Run the Project

Click `Run -> Run As -> Java Application -> Main - hellofx` to run the project.

Steps to create a JavaFX Program

1. Extend Application
2. Override start (Stage)
3. Create Nodes (e.g., Button)
4. Place the Nodes in the Scene
5. Place the Scene on Stage
6. Show Stage



User Interface Code

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.StackPane;


public class Main extends Application {
    //Inside the main() method, we can launch our application using Application.launch().
    public static void main(String[] args) {
        launch(args);
        }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("My First JavaFX GUI"); //add some nice caption to our window.
        Button btnHello= new Button("Hello"); //Create GUI Elements

        StackPane layout= new StackPane();
        layout.getChildren().add(btnHello);

        Scene scene1= new Scene(layout, 300, 250); //create
        primaryStage.setScene(scene1);

        primaryStage.show(); // It is hidden by default.
        }
        }
```

Main Frame

GUI Widgets

Select Layout

Container

Add scene in a primary stage

Developing Age Group Application

Creating Grid Pane

```java
public class Main extends Application implements EventHandler<ActionEvent>{
    Label groupLabel;
    Button showGroupBtn,shiftBtn;
    @Override
    public void start(Stage primaryStage) {
        try {
            GridPane root = new GridPane();
            root.setPadding(new Insets(10,10,10,10));
            root.setVgap(8);
            root.setHgap(10);
```

Creating Nodes

```java
        Label titleLabel=new Label("Age Group Guide");
        titleLabel.setAlignment(Pos.CENTER);
        GridPane.setConstraints(titleLabel,2,0);

        Label ageLabel=new Label("Age");
        GridPane.setConstraints(ageLabel,1,2);

        TextField ageInput=new TextField("20");
        GridPane.setConstraints(ageInput,2,2);


        showGroupBtn=new Button("Show My Age Group");
        GridPane.setConstraints(showGroupBtn,2,4);
        showGroupBtn.setOnAction(this);

        //shiftBtn=new Button("Show Second Window");
        //GridPane.setConstraints(shiftBtn,2,2);
        //shiftBtn.setOnAction(this);

        groupLabel=new Label();
        GridPane.setConstraints(groupLabel,2,8);
```

Adding Nodes to the Layout (Pane)

```
        GridPane.setConstraints(ageInput,2,2);


        showGroupBtn=new Button("Show My Age Group");
        GridPane.setConstraints(showGroupBtn,2,4);
        showGroupBtn.setOnAction(this);

        //shiftBtn=new Button("Show Second Window");
        //GridPane.setConstraints(shiftBtn,2,2);
        //shiftBtn.setOnAction(this);

        groupLabel=new Label();
        GridPane.setConstraints(groupLabel,2,8);

        root.getChildren().addAll(titleLabel, ageLabel, ageInput, showGroupBtn, groupLabel);
        Scene scene = new Scene(root,400,400);
        //scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
        primaryStage.setTitle("Age Group");
        primaryStage.setScene(scene);
        primaryStage.show();
    } catch(Exception e) {
        e.printStackTrace();
    }
```

Creating Scene and Adding Layout to the Scene

```
        root.getChildren().addAll(titleLabel, ageLabel, ageInput, showGroupBtn, groupLabel);
        Scene scene = new Scene(root,400,400);
        //scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
        primaryStage.setTitle("Age Group");
        primaryStage.setScene(scene);
        primaryStage.show();
    } catch(Exception e) {
        e.printStackTrace();
    }
```

Implementing Event Handler

```java
public class Main extends Application implements EventHandler<ActionEvent>{
    Label groupLabel;
    Button showGroupBtn,shiftBtn;
    @Override
    public void start(Stage primaryStage) {
        try {
            GridPane root = new GridPane();
            root.setPadding(new Insets(10,10,10,10));
            root.setVgap(8);
            root.setHgap(10);


            Label titleLabel=new Label("Age Group Guide");
            titleLabel.setAlignment(Pos.CENTER);
            GridPane.setConstraints(titleLabel,2,0);

            Label ageLabel=new Label("Age");
            GridPane.setConstraints(ageLabel,1,2);

            TextField ageInput=new TextField("20");
            GridPane.setConstraints(ageInput,2,2);


            showGroupBtn=new Button("Show My Age Group");
            GridPane.setConstraints(showGroupBtn,2,4);
            showGroupBtn.setOnAction(this);
```

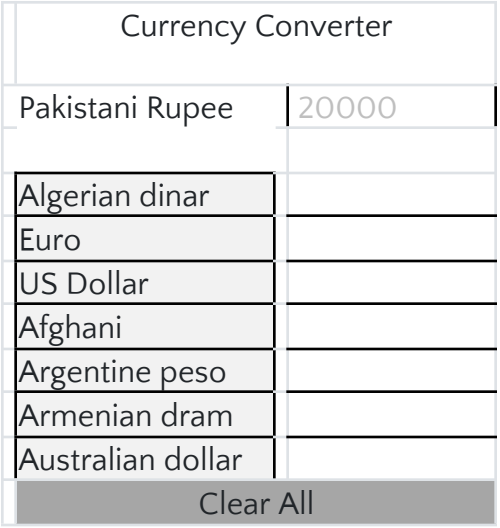
```java
    @Override
    public void handle(ActionEvent event) {
        if(event.getSource()==showGroupBtn) {
            groupLabel.setText("You are senir Citizen");
        }
        else if(event.getSource()==shiftBtn) {
            SecondWindow.display();
        }
    }
```

Creating Connection between different windows

- Create another class File -> New -> Class and name it as SecondWindow.java
- Create a function e.g. display()
- Edit the function to add Stage, Pane, Nodes, Scene etc. and show the stage
- Now call SecondWindow.display inside event handler of some node (button) of the first window.

Task#01

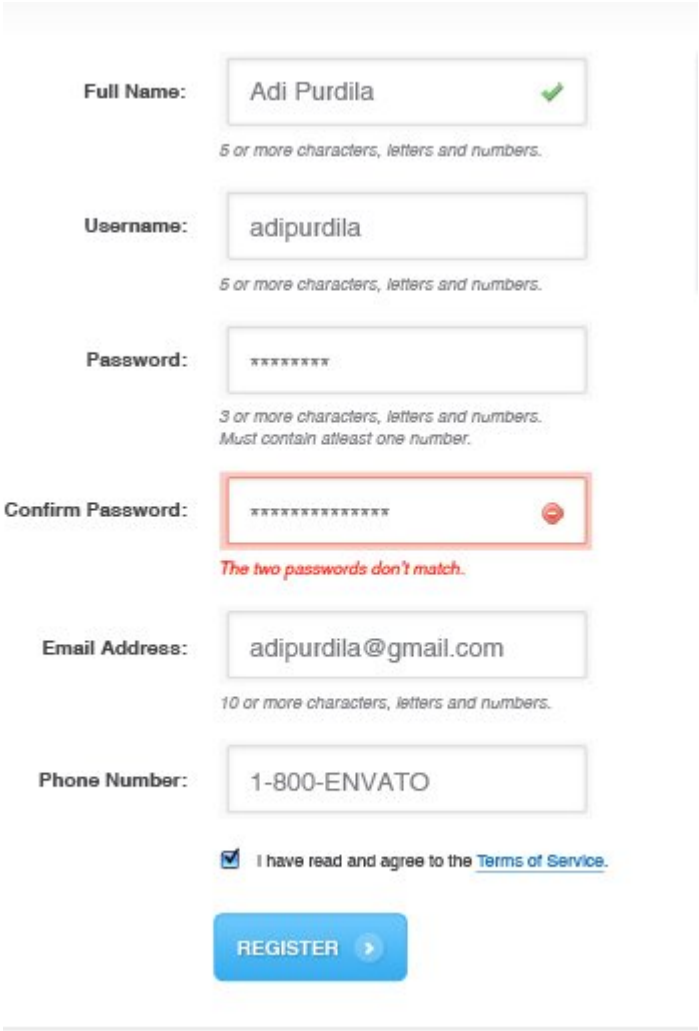
Create a currency converter in which a user enters money in Pakistani Rupee and selects an appropriate button for conversion to any other currency. The currency converter should display the new (converted) currency. The clear all button removes all data.



Task#02

Create two different windows for login and signup as shown below. Create a connection between two windows such that if a user is new he should move to signup window. Similarly after successful registration the user should be directed to login window. Add validation on data entered in login and signup page using event handlers. Following user input validations are important:

1. The name must not contain any numbers
2. The username must not contain any spaces
3. Password should contain at least 8 characters
4. Password and confirm password field must match
5. Email address must not have any spaces
6. Phone number must contain numbers



Submission Instructions:

- Save all program files with your roll no and task number e.g. i20XXXX_Problem01
- Now create a new folder with name ROLLNO_LAB02 e.g. i22XXXX_LAB02
- Move all of your program files to this newly created folder and compress it into .zip file.
- Now you have to submit this zipped file on Google Classroom.