Creating a stand-alone Java application using post Java 10 with JavaFX

JP_5_3_Practice: Creating a JavaFX application

In this tutorial you will learn how to create a Java FX application in eclipse.

The application will be a simple voting application that you can create a JAR file of so that it can be ran outside of the development environment.

From Java 11 onwards JavaFX does not come included in the Java JDK but has to be added and configured separately in Eclipse.

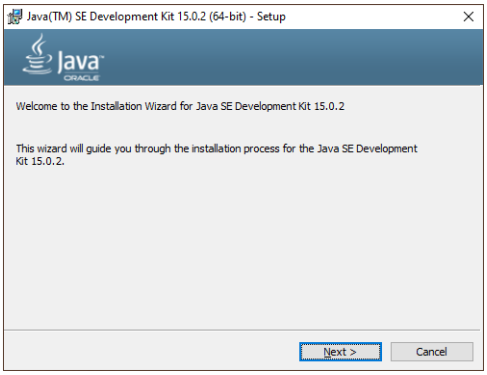
Install the latest Java Development Kit

1. You need to download and install the latest Java Platform, Standard Edition (Java SE) that lets you develop and deploy Java applications on desktops and servers

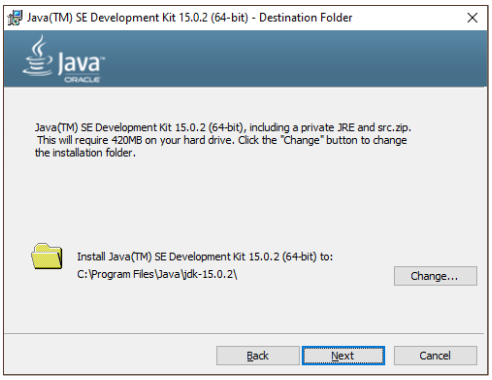
    a. Go to https://www.oracle.com/java/technologies/java-se-glance.html and download the latest JDK available

    b. Double click the downloaded .exe file to start the install of the JDK

    c. Click Next to start the setup process



    d. Click Next to install to the default location



    e. Click Close to complete the JDK install

    f. When you install the latest version of a JDK it is worth removing all previous versions of Java from your system to avoid any internal conflicts

Install the latest version of Eclipse

2. You need to download and install the latest version of Eclipse

   a. Go to https://www.eclipse.org/downloads/ and download the latest version of Eclipse.

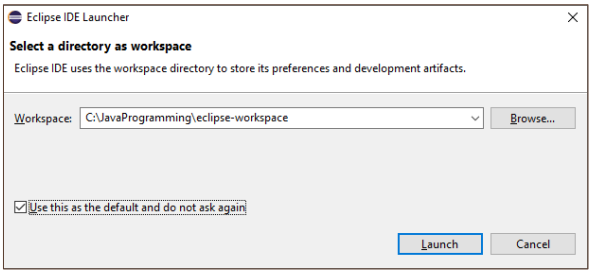
   b. Double click the downloaded .exe file to start the install of Eclipse

   c. Select the Eclipse IDE for Java Developers Option from the install menu

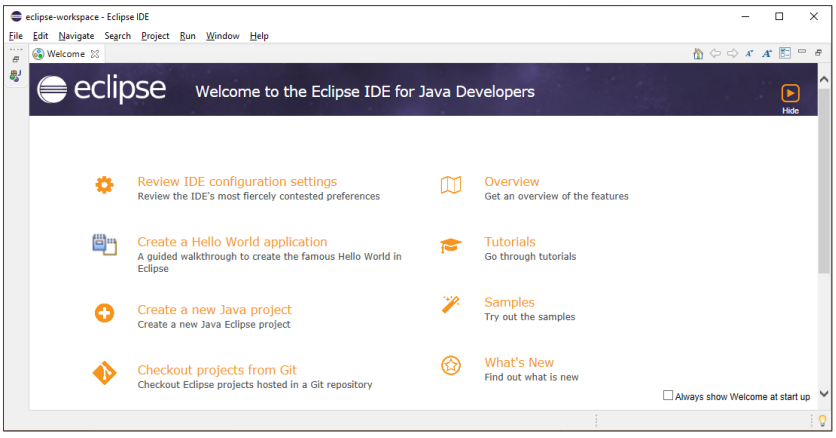
   d. Click install to install Eclipse to the default location

   e. Click the Launch button to start Eclipse

   f. Select the workspace that will contain the code you write (Use a new workspace)

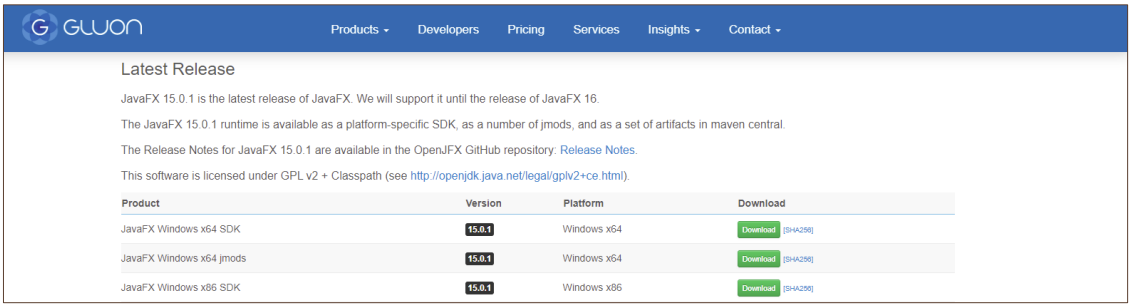


   g. Eclipse will launch with the welcome screen

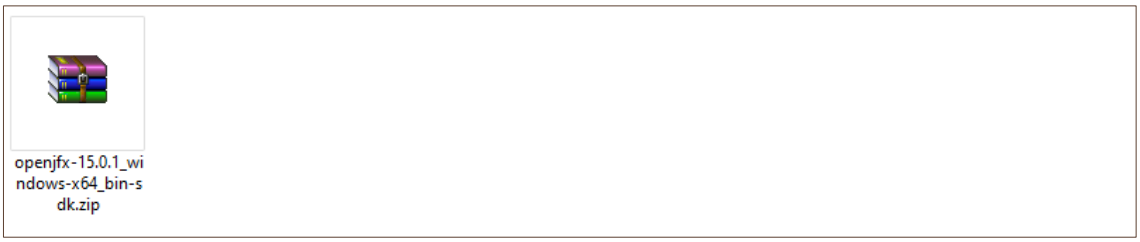


Downloading the Java FX library.

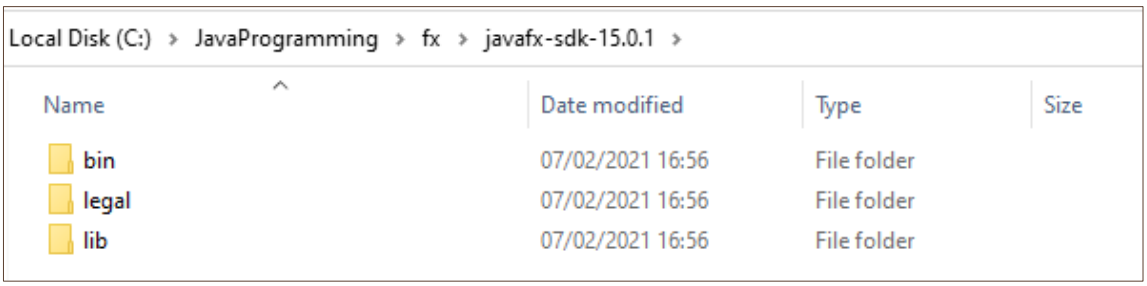
3. FX is not included by default in the Java JDK so must be downloaded.
    a. Go to https://gluonhq.com/products/javafx/ and download the latest (at the time of writing this was JavaFX(15)) suitable version of JavaFX for your Operating System



    b. Extract the downloaded files to a location of your choice. You will use this location when setting up JavaFX in Eclipse so remember where you extract it to

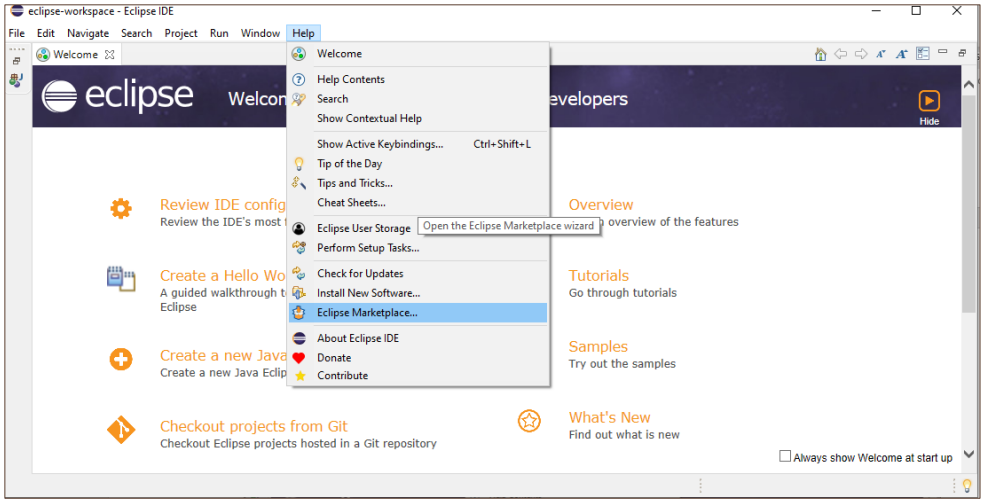


    c. You will end up with the following folder structure for JavaFX

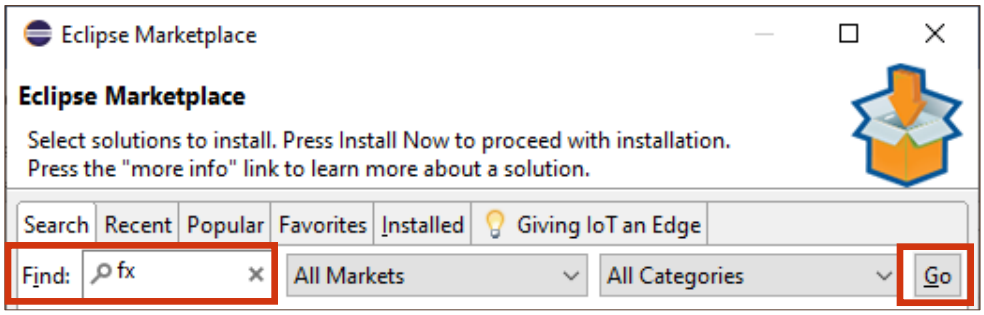


Installing Java FX in Eclipse.

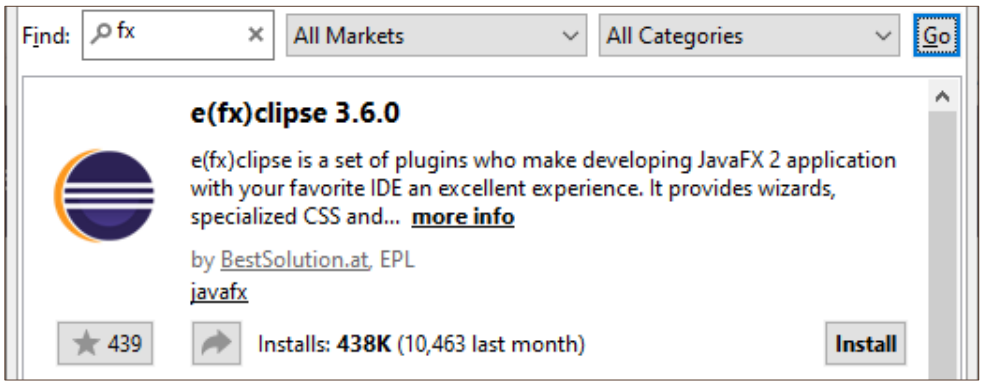
4. FX is not installed by default in Eclipse so must be added.
    a. Start Eclipse and select help from the menu
    b. Select Eclipse Marketplace from the drop-down menu



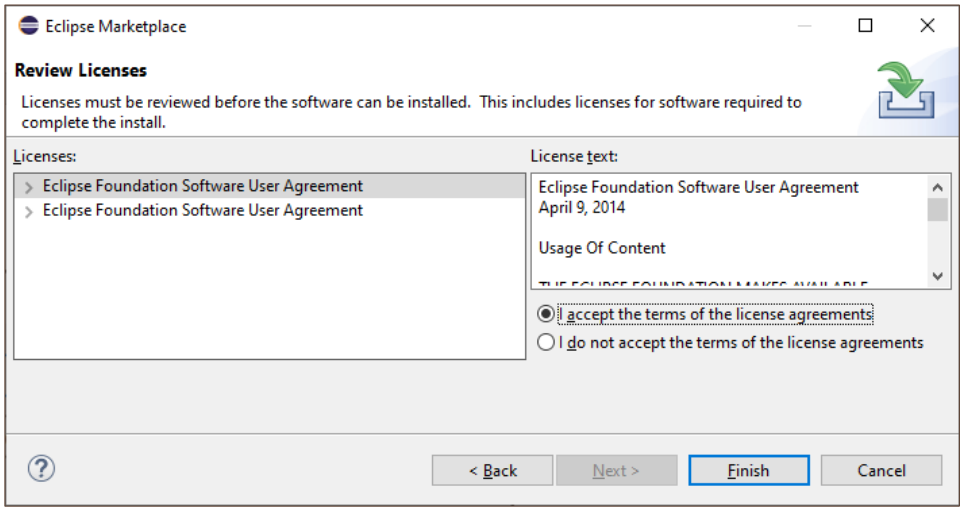
    c. Type fx in the search bar and click the go button



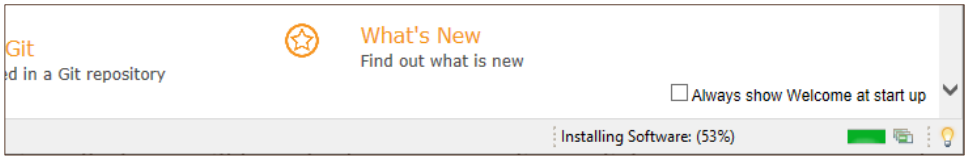
    d. Select the latest version of JavFX and click on the Install button



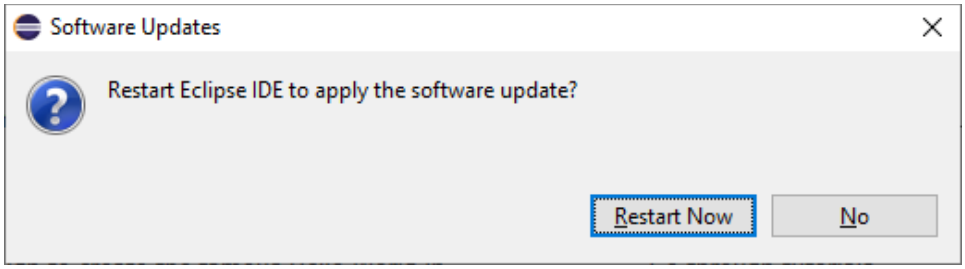
e. Agree to the terms of the license agreement and click Finish to complete the install



f. Eclipse will now install the selected software (this may take a few minutes)



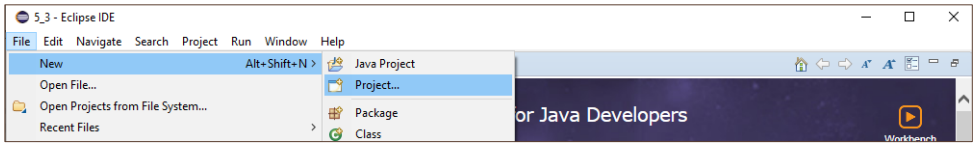
g. When the software is installed you will be asked to restart Eclipse, click Restart now to complete the install



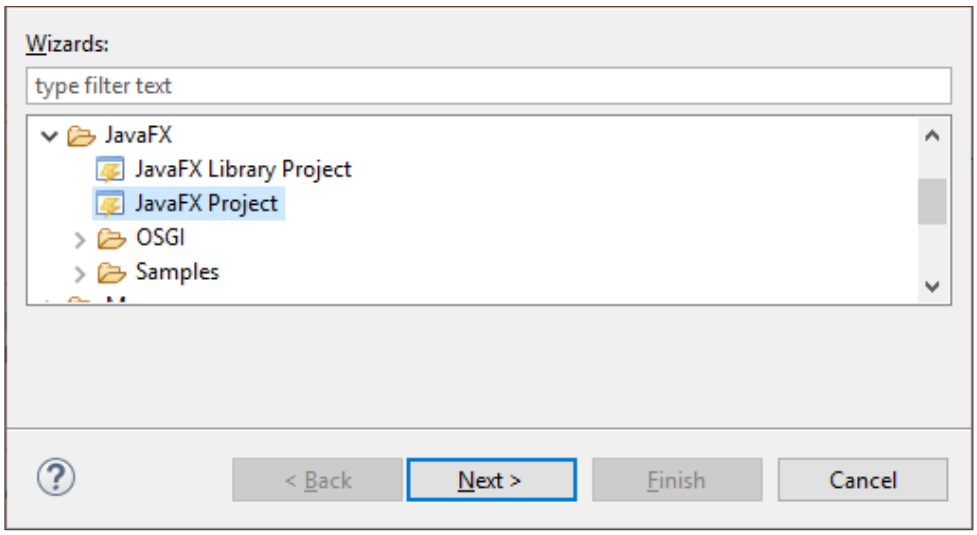
Creating a JavaFX application in Eclipse.

5. To create a JavaFX application in Eclipse you need to create a JavaFx project.

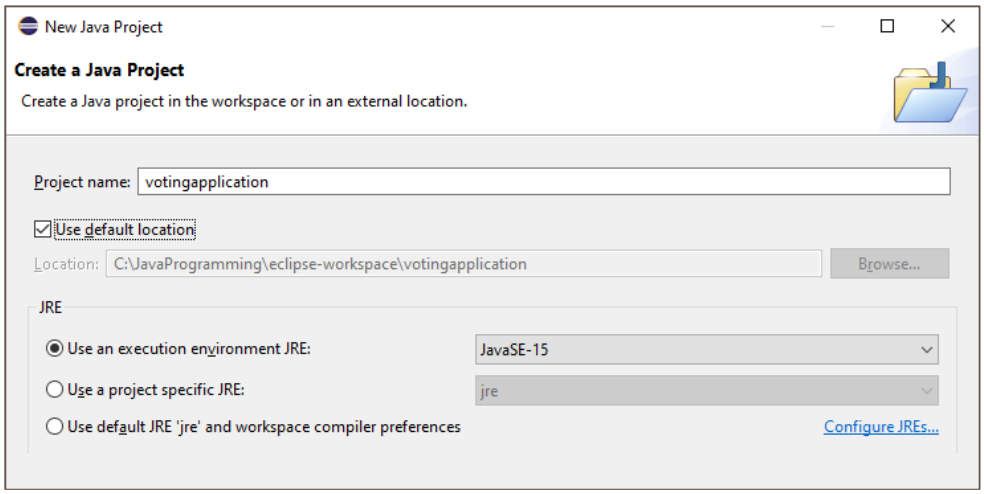
   a. Select File, New and Project from the menu from the restarted Eclipse IDE



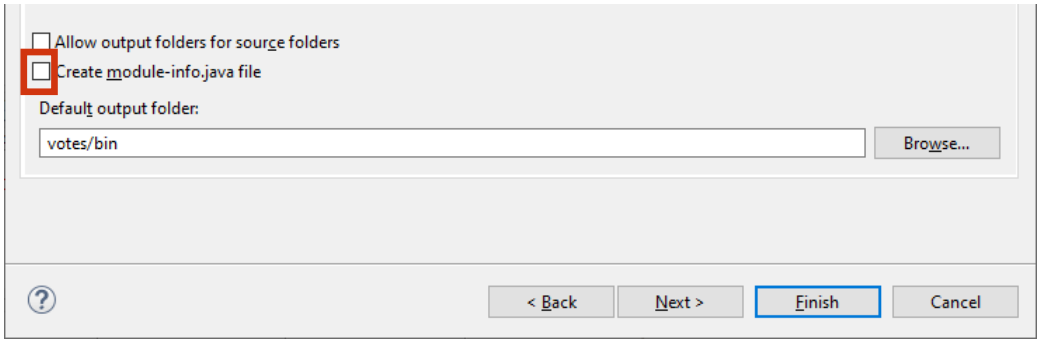
   b. Select JavaFX and then JavaFX Project from the drop-down list and click Next>



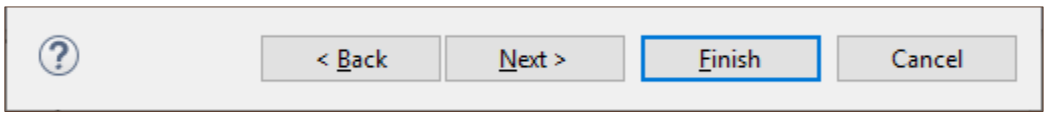
   c. Call the project votingapplication and select a workspace location or tick the box to use the default location



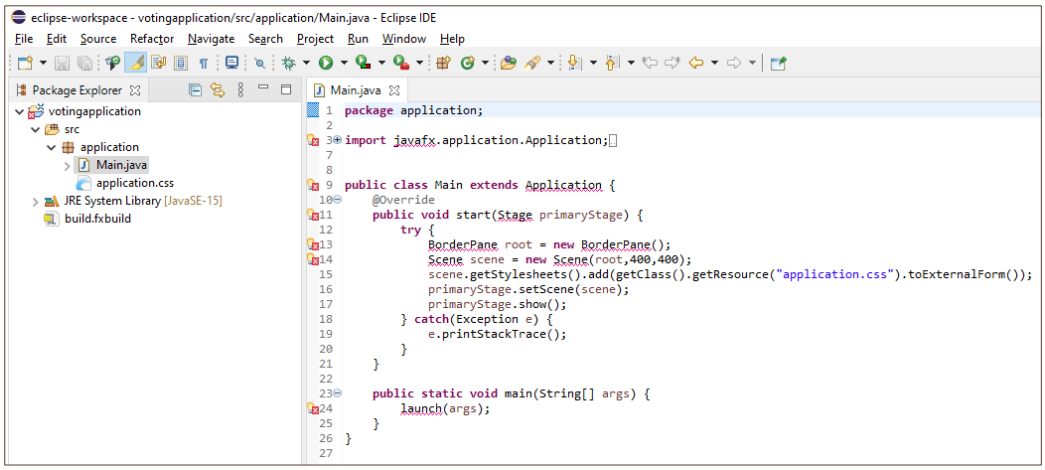
d. Leave the settings as their default and click Next, untick the box to make create a module file



e. Click Finish to create the JavaFX project



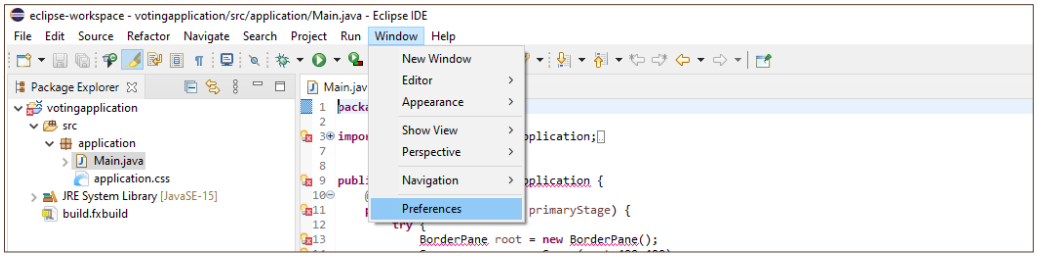
f. Once the building workspace process is complete you can open the Main.java file to see the base FX code that is provided for you to build on



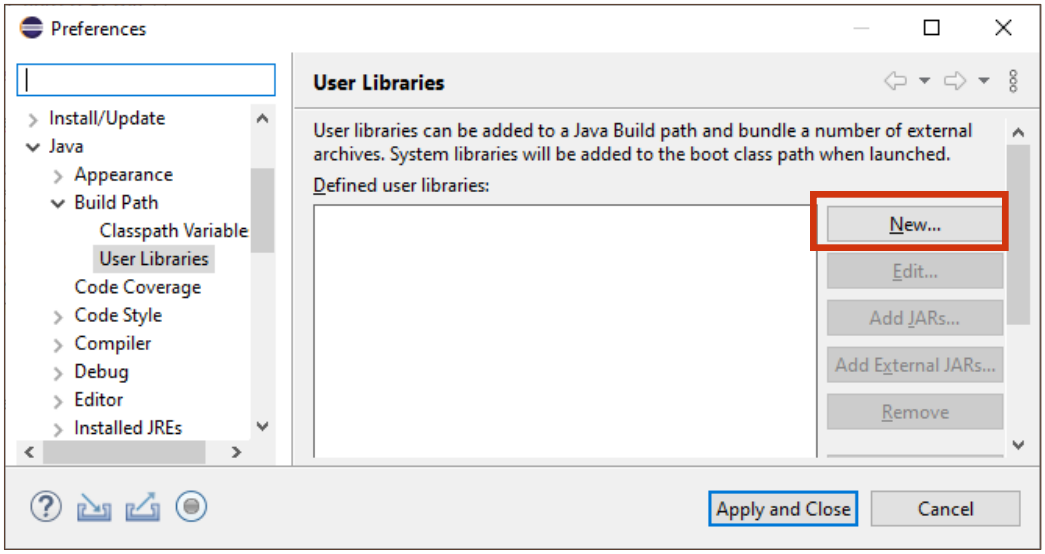
g. As you can see there are errors in the code, this is because you need to configure JavaFX within your project

Configuring Eclipse to use JavaFX.

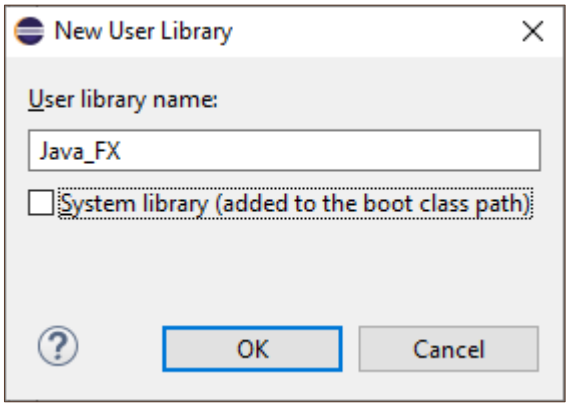
6. To create a JavaFX application in Eclipse you need to configure the project settings.
   a. Click on **Windows** and then select **preferences** from the drop down



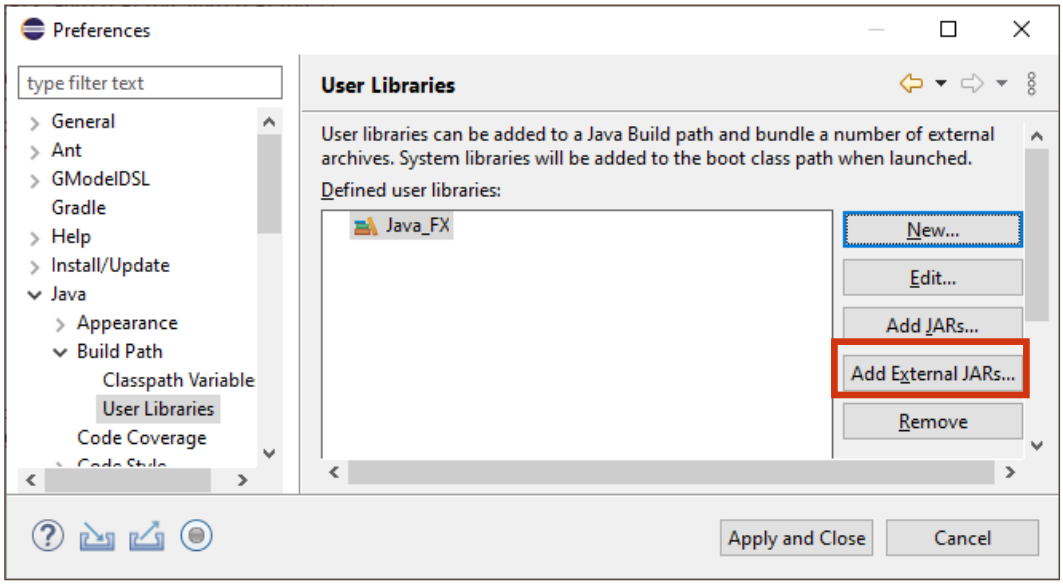
   b. From the options select: **Java – Build Path – User Libraries – click New…**



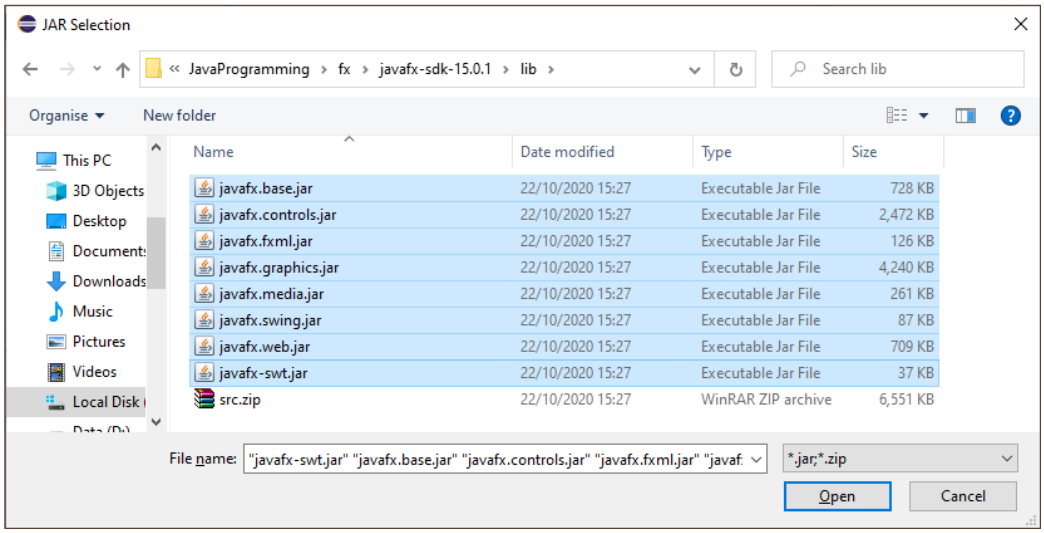
   c. Add Java(FX) as the library name and click OK



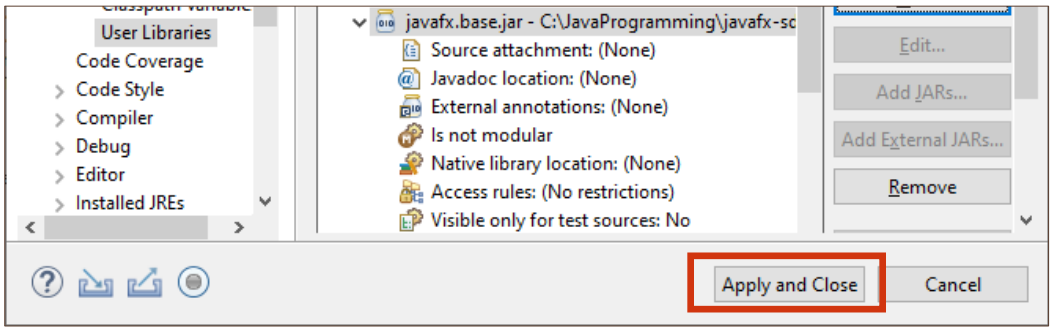
d. Click on Add External JARs



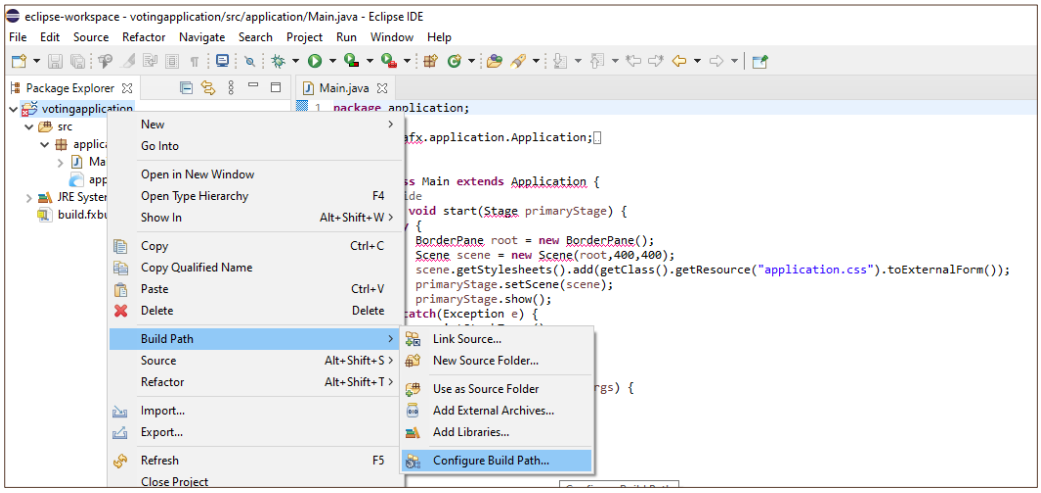
e. Browse to where you extracted the JavaFX files, go to the lib folder, select all the jar files and click Open



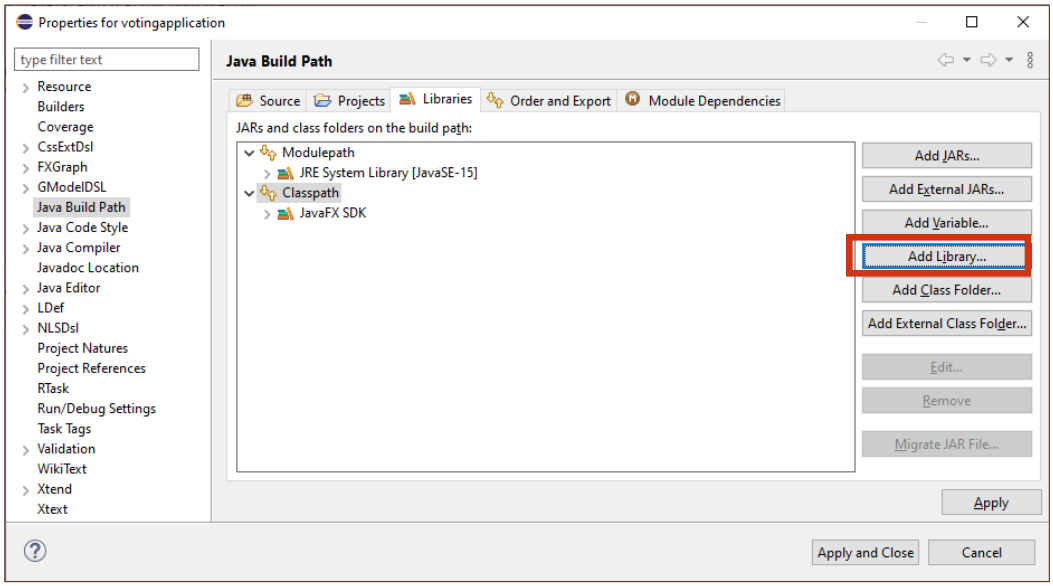
f. Click Apply and Close



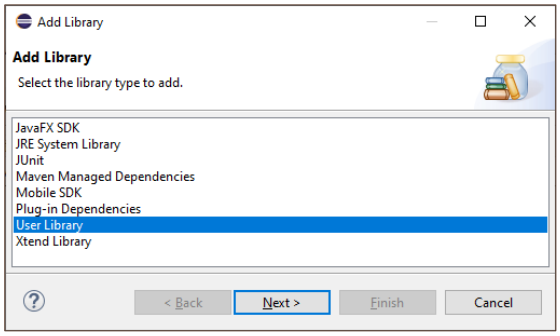
g. Right click the project file and select **Build Path** – **Configure Build Path** from the drop-down menu



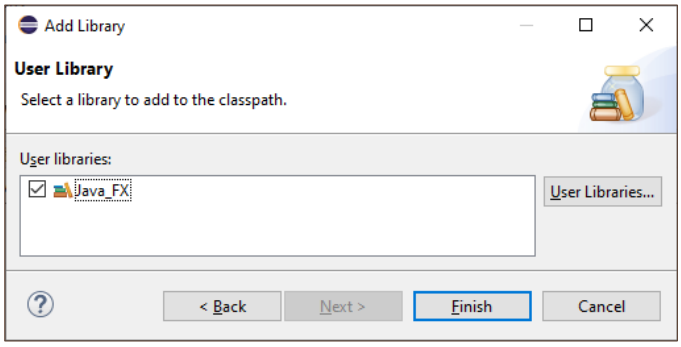
h. Click the **libraries** tab, then select **Classpath** and the click Add **Library**



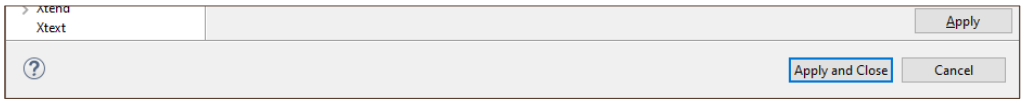
i. Select **User Library** from the list and click Next



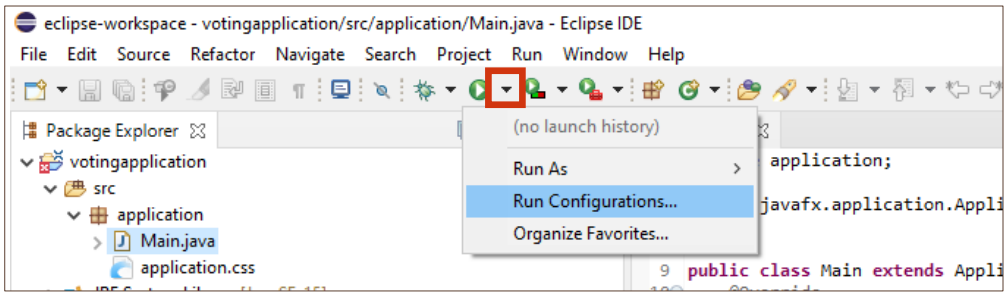
j. Select the java(FX) user library that you set up earlier click Finish



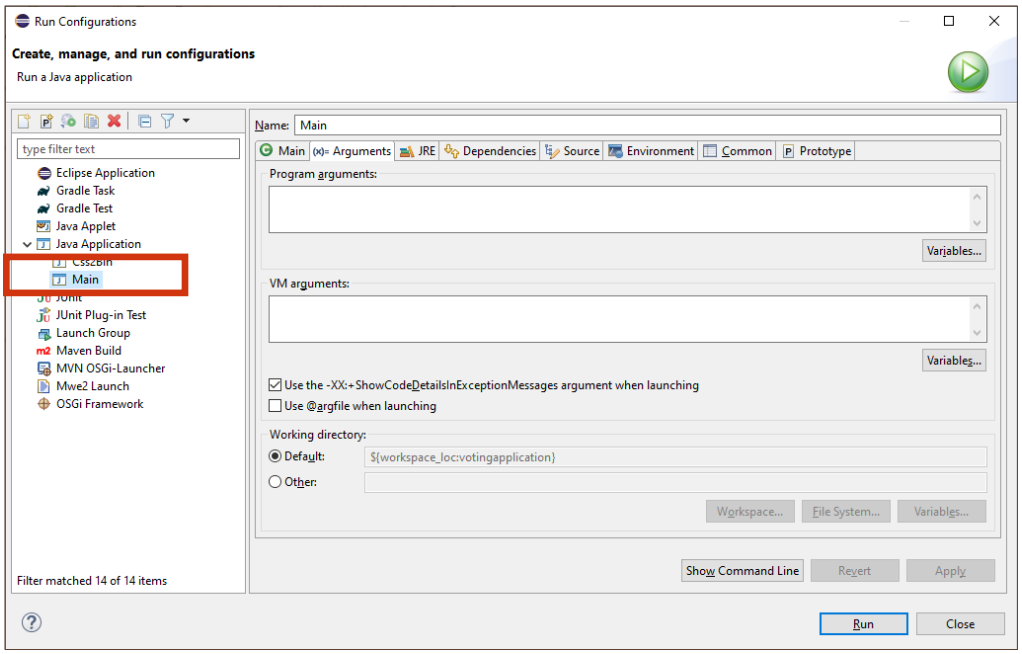
k. Click Apply and Close



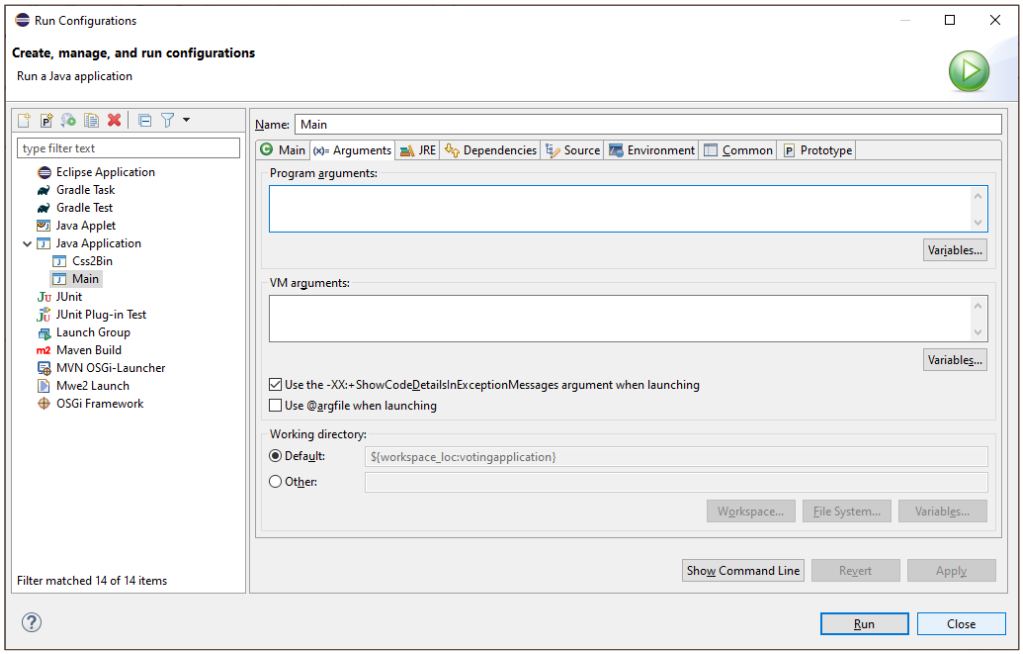
l. Double click on Main.java to select it and then the down arrow beside the **Run** button and select **Run Configurations** from the drop-down menu



m. Select **Java Application** and then **Main** from the menu on the left



n. Click the **Arguments** tab and click in the **VM Arguments** text area

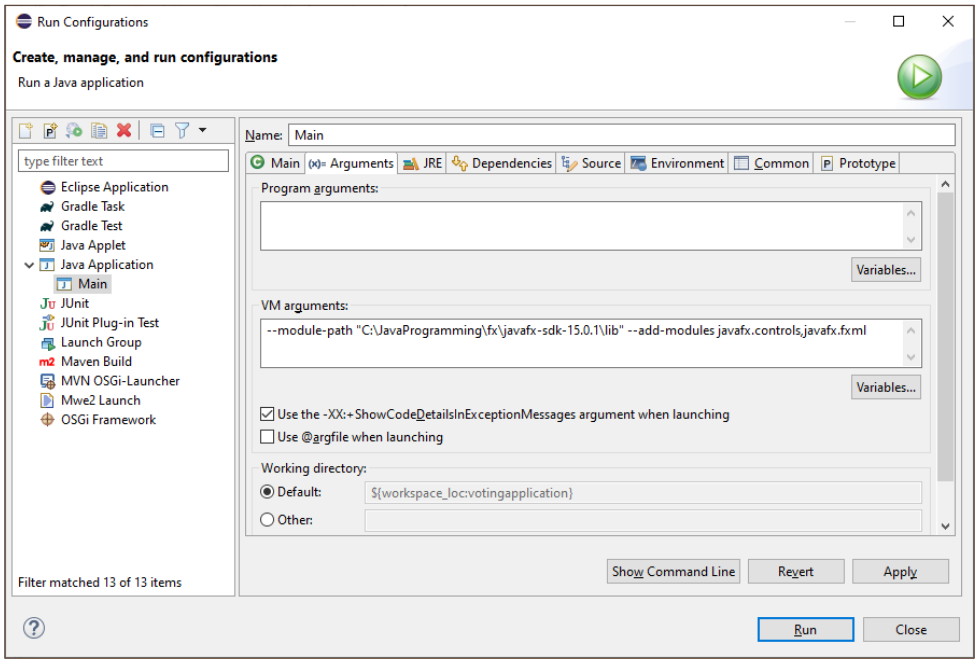


o. Here you will need to set up the path to your JavaFX controls, the following example shows where your path to the lib folder in your JavaFX directory needs to be added to the command:

```
--module-path "your directory path\lib" --add-modules javafx.controls,javafx.fxml
```

This example has used the following path to the JavaFX controls:

```
--module-path "C:\JavaProgramming\fx\javafx-sdk-15.0.1\lib" --add-modules
javafx.controls,javafx.fxml
```



p. Click Run to complete the JavaFX configuration

Coding a JavaFX application

7. When you create a JavaFX project you combine components and action listeners to create an interactive GUI application. The following application will allow users to vote for either Java or SQL as their favourite topic.

   a. Create the following instance fields above the start() method in the code. These will be used to keep the score and for the content of a label

```
public class Main extends Application {
   int cat1Score=0;
   String cat1 = "Java Team";
```

   b. Create a label that will display the category for the vote using the following code at the start of the main method (above the line that identifies the BorderPane):

```
public void start(Stage primaryStage) {
      //create and position the Java category label
      Label cat1Lbl = new Label(cat1);
      cat1Lbl.setLayoutX(110);
      cat1Lbl.setLayoutY(10);
      cat1Lbl.setTextFill(Color.RED);
```

Label cat1Lbl = new Label(cat1)- creates a new label that displays the value of cat1.

cat1Lbl.setLayoutX(110)- sets the X co-ordinate of the label.

cat1Lbl.setLayoutY(10)- sets the Y co-ordinate of the label.

cat1Lbl.setTextFill(Color.RED)- Sets the text color of the label.

   c. You will have to import libraries to use these components. Always make sure you import the javaFX library.

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.paint.Color;
```

d. You will now create another label that will display the number of votes cast. This will be displayed under the previous label and centered to the position of cat1Lbl.

```
//create and position the category 1 scores label
Label score1Lbl  = new Label(""+cat1Score);
score1Lbl.setLayoutX(130);
score1Lbl.setLayoutY(40);
score1Lbl.setTextFill(Color.RED);
```

e. A button is needed that the user can click to cast their vote. This will be positioned under the previous two labels. You will have to import the Button library, remember to choose the fx option.

```
//Create and position the category 1 button
Button cat1Btn = new Button("Vote " + cat1);
cat1Btn.setLayoutX(90);
cat1Btn.setLayoutY(80);
```

f. To make a component interactive you need to add an event handler to the component. Use the drop-down list of content prompts in eclipse to create the following action listener that uses a Lambda expression:

```
cat1Btn.setOnAction((ActionEvent event) -> {

}); //end lambda expression
```

You will need to import the javaFX ActionEvent.

g. The code inside the handle method is fired whenever the button is clicked. Add the code to increment the number of votes (stored as an instance field) and update the text in the label to show the new value.

```
cat1Btn.setOnAction((ActionEvent event) -> {
      cat1Score++;
      score1Lbl.setText(""+cat1Score);
}); //end lambda expression
```

h. Although you have created the components you have not yet added them to your application so if you were to run your code now you would not see them. To add the components you have to add them to an FX layout manager. Currently the application uses a BorderPane layout (you can learn more about these through the Java API) but you are going to use a Group layout that allows the placing of components using X and Y co-ordinates.
Update the existing code:

```
BorderPane root = new BorderPane();
```

to

```
Group root = new Group();
```

You will have to import the javafx.scene.Group library (you can delete the BorderPane library import as you won't be using it).

i. Add each component individually to the root node using the getChildren().add method:

```
Group root = new Group();
root.getChildren().add(cat1Lbl);
root.getChildren().add(score1Lbl);
root.getChildren().add(cat1Btn);
```

As you are placing them with their co-ordinates the order that you add them is not important (this is not always case).

j. Update the size of the scene (this is what you will add the root to in order to display the components in the primary stage (Window) to have a width of 300 and a height of 150.

```
Scene scene = new Scene(root, 400, 150);
```

k. You will not be using the stylesheets in this exercise so you can remove the line that starts with scene.getStyleSheets().

l. Add the following code that sets a title on the window and sets the width and height of the window equal to the dimensions of the scene.

```
primaryStage.setTitle("Voting Panel!");
primaryStage.setMinWidth(scene.getWidth());
primaryStage.setMinHeight(scene.getHeight());
primaryStage.setScene(scene);
```

m. You can now run and test that your code works.

8. Use the following information to create a second set of labels and buttons:

a. An integer instance field named cat2Score that is initialized to 0.

b. A String instance filed named cat2 that is initialized to "SQL Team"

c. Create a cat2Lbl at position 230, 10 that displays the text in blue.

d. Create a score2Lbl at position 250, 40 that displays the score in blue.

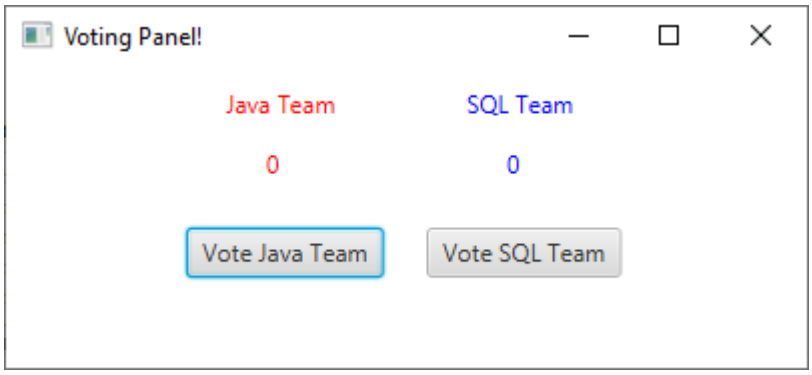
e. Create a button named cat2Btn that displays vote plus the value of cat2. Its position should be 210, 80.

f. Add an event handler for the button that increases cat2Score by one and updates the text value.

g. Add the components to the Group layout manager.

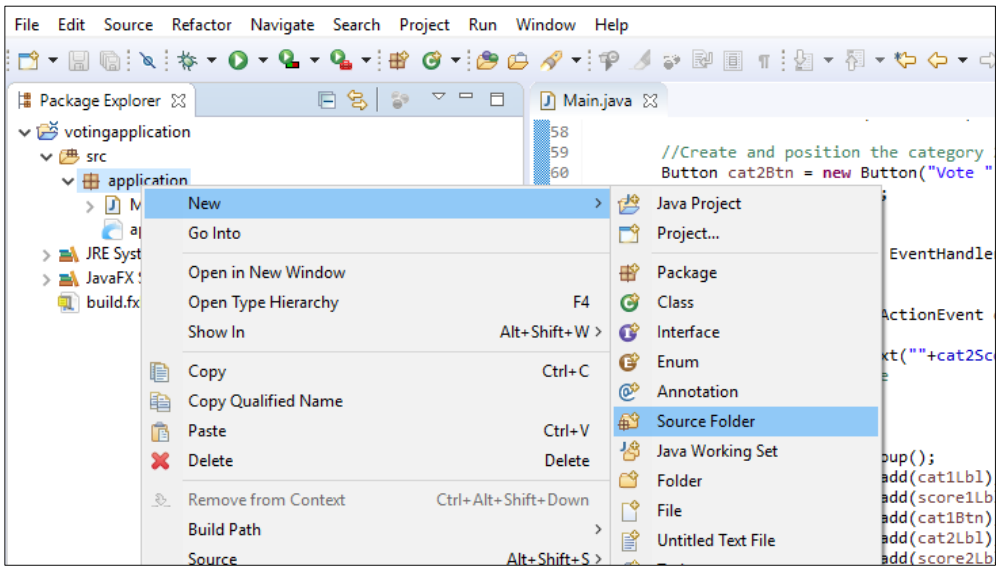
h. Run and test that your code works.

i. Your application should look like this:

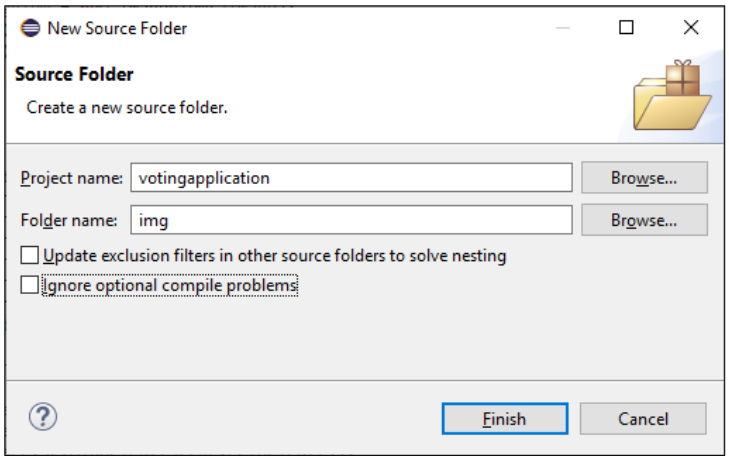


9. You will now add images to the application by including them in a resources folder with the application.

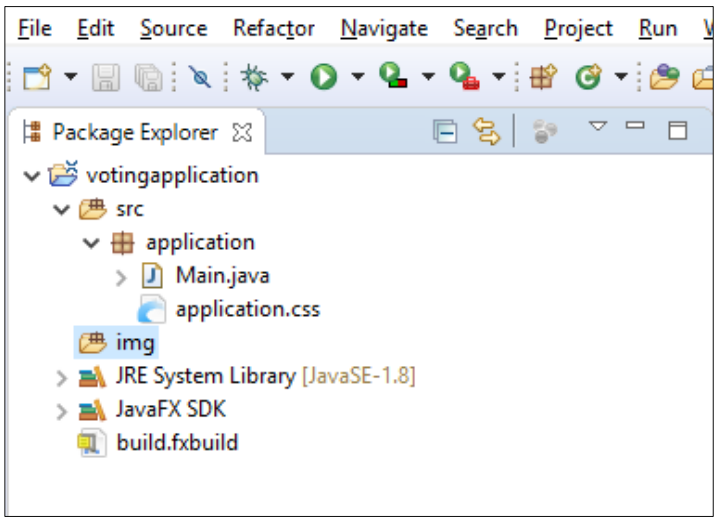
   a. In Eclipse, right click on the application package and select New, Source Folder.



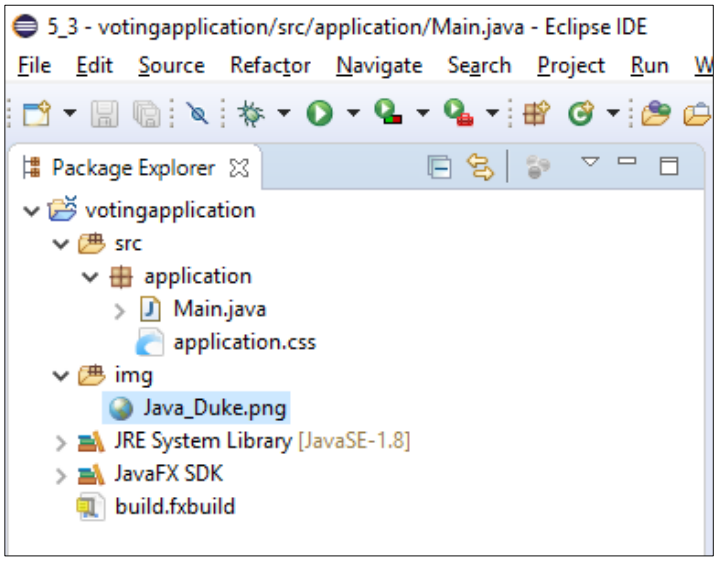
   b. Name the source folder img and then click Finish.



c. The img folder is displayed in your project directory.



d. Drag the image file Java_Duke.png from your local directory into the img folder. Choose copy link and then OK to complete the process.



e. Above the try catch block in your code add the following line of code that creates a reference (jImage) to the Java_Duke image in the img folder. You will need to import the fx image library.

```
//create a link to the Java image
Image jImage =
new Image(Main.class.getResourceAsStream("/Java_Duke.png"));
```

f. To display the image in the application you have to place it in an ImageView object that you can then use to set the display co-ordinates. Add this

```
//Setting the image view and place the image
ImageView jImgView = new ImageView(jImage);
jImgView.setX(10);
jImgView.setY(30);
```

10. Use the following information to add an image for the SQL Team vote.

    a. Drag the Datagirl.png image file into the img source folder.

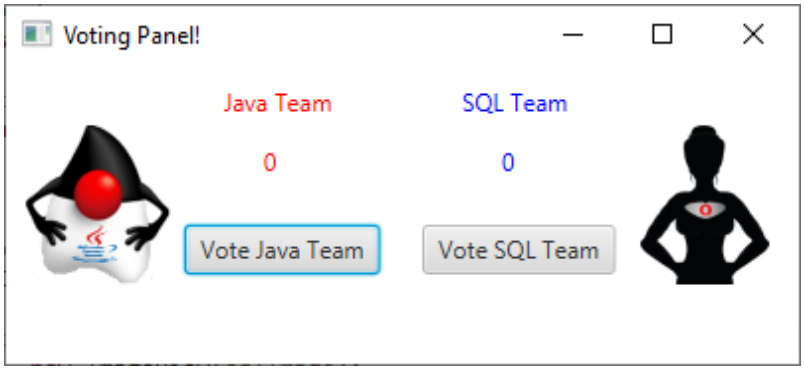
    b. Create a reference to the Datagirl image named sqlImage.

    c. Create a sqlImgView using the sqlImage reference.

    d. Place the sql image at the following co-ordinates: X- 320, Y-30.

    e. Add the image view to the Group layout m,anager.

    f. Your application should look like this:



11. You now have a working app that allows votes to be cast, however the user experience can be improved with a few tweaks.

    a. The vote count should not be displayed on screen until the vote is over.

        - Under the setTextFill() method for each score label add the following code so that the labels are not displayed to screen.

```
score1Lbl.setTextFill(Color.RED);
score1Lbl.setVisible(false);

score2Lbl.setTextFill(Color.BLUE);
score2Lbl.setVisible(false);
```

- Create a private method named showVotes that accepts no parameters and returns no values. This method will set the value of the scores as the text in the score labels and then display the labels to screen. Create the method above the main method at the bottom of the code.

```
private void showVotes() {
      score1Lbl.setText(""+cat1Score);
      score2Lbl.setText(""+cat2Score);
      score1Lbl.setVisible(true);
      score2Lbl.setVisible(true);
}//end method showVotes
```

- You get errors as the compiler cannot recognise the scoreLbl components. This is because you created them locally in the start method. To fix this you can create the reference to the component as an instance field and then create the component object locally.

  Add the following instance fields to the top of your class:

```
public class Main extends Application {
      int cat1Score=0, cat2Score = 0;
      String cat1 = "Java Team", cat2 = "SQL Team";
      private Label score1Lbl, score2Lbl;
```

- Update the code in the start menu to instantiate the object referenced as an instance field instead of creating a second one. For the score labels update the code by not declaring a type at the beginning of the statement so that it does this:

```
//create and position the category 1 scores label
score1Lbl = new Label(""+cat1Score);

//create and position the category 2 scores label
score2Lbl = new Label(""+cat1Score);
```

  Any components/objects that are required to be used in multiple methods should have a reference created as an instance field.

  The errors that were attached to the showVotes() method should now have been resolved.

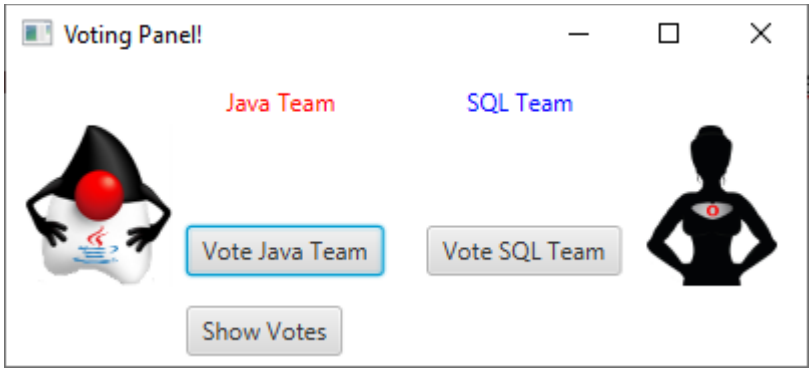
- Comment out the lines in the EventHandlers that display the values of the scores to the labels.

```
cat1Btn.setOnAction((ActionEvent event) -> {
      cat1Score++;
      //score1Lbl.setText(""+cat1Score);
});//end lambda expression
```

Do the same for the cat2Btn event handler.

b. To view the votes cast,

- create a new button named showBtn that will display the text "Show Votes". Display the button at the following co-ordinates: X-90, Y-120 and add it to the Group layout manager.
- Call the showVotes() method from its event handler.
- Run and test your program
- Your application should look like this:



c. You can see that the size of the buttons is different and doesn't look quite right. You can fix this by setting a preferred width for the buttons.

- Create an unchangeable value at the beginning of the start method:

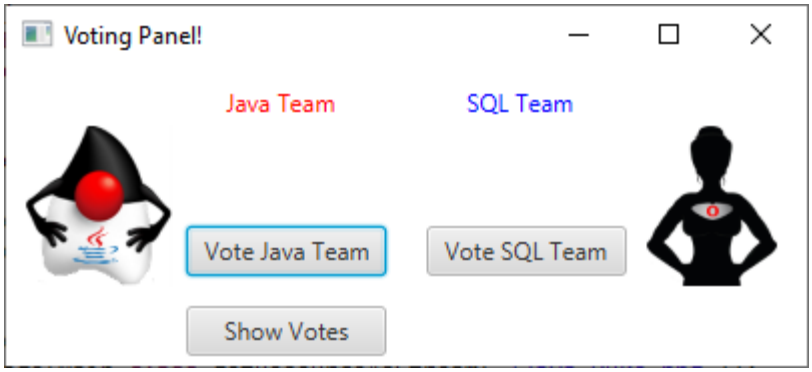
```
public void start(Stage primaryStage) {
        //unchangeable value for the width of the buttons.
final double BTN_WIDTH = 100;
```

- In the properties list for all of the buttons use the setPrefWidth() method to set the width of the button equal to the BTN_WIDTH value.

```
        showBtn.setLayoutX(90);
        showBtn.setLayoutY(120);
showBtn.setPrefWidth(BTN_WIDTH);
```

Do the same for the cat1Btn and the cat2Btn buttons.

- Run your program
- Your application should look like this:

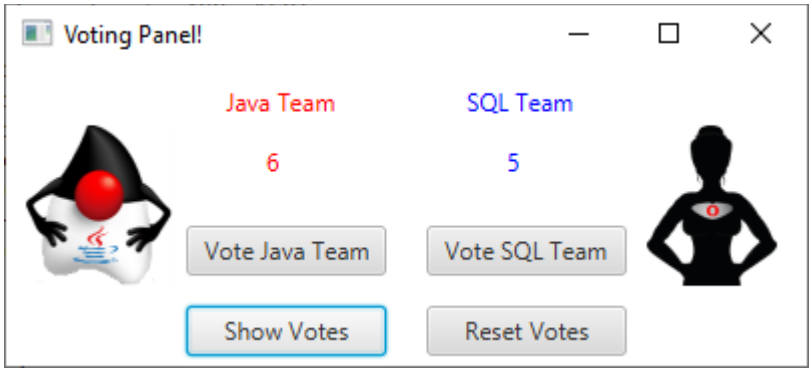


d. You will now add functionality that will reset the votes to zero so that voting can begin again.

- Create a private method that will set the values of the scores to zero and hide the scores labels.

```
private void resetVotes() {
      cat1Score=0;
      cat2Score=0;
      score1Lbl.setVisible(false);
      score2Lbl.setVisible(false);
}//end method resetVotes
```

- Create a button named resetBtn that displays "Reset Votes" as its text, it's positioned at 210, 120 and uses the preferred button width.
- Call the resetVotes() method from its event handler.
- Run and test your program
- Your application should look like this:



e. There are a few issues with the application:

- After showing the vote values you can click the vote buttons to add to the totals.
- You can reset the votes at any time.
- The votes are not reset after they have been viewed.

f. Create a private void method named setBtnUse that accepts 4 boolean parameters that will be used to control the disabled state of the buttons. You will have to create button references as instance fields for this to work (remember to remove the Button type from the start method)

```
private void setBtnUse(boolean cat1, boolean cat2,
                       boolean show, boolean reset) {
     cat1Btn.setDisable(cat1);
     cat2Btn.setDisable(cat2);
     showBtn.setDisable(show);
     resetBtn.setDisable(reset);
}//end method setBtnUse
```

g. You can fix these issues by simply controlling access to the buttons.

- The reset button should be disabled until the showVote button is clicked. When the reset button is created set it to disabled:

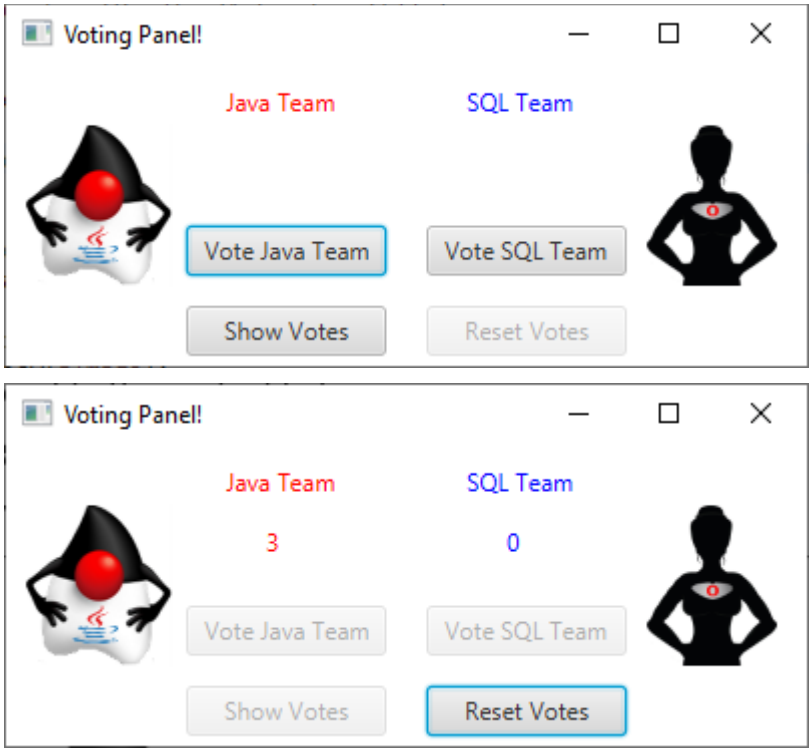
```
resetBtn.setPrefWidth(BTN_WIDTH); resetBtn.setDisable(true);
```

- In the event handler for the showBtn call the setBtnUse method after the showVotes() call passing the following values:

```
public void handle(ActionEvent event) {
     showVotes();
     setBtnUse(true, true, true, false);
}//end method handle
```

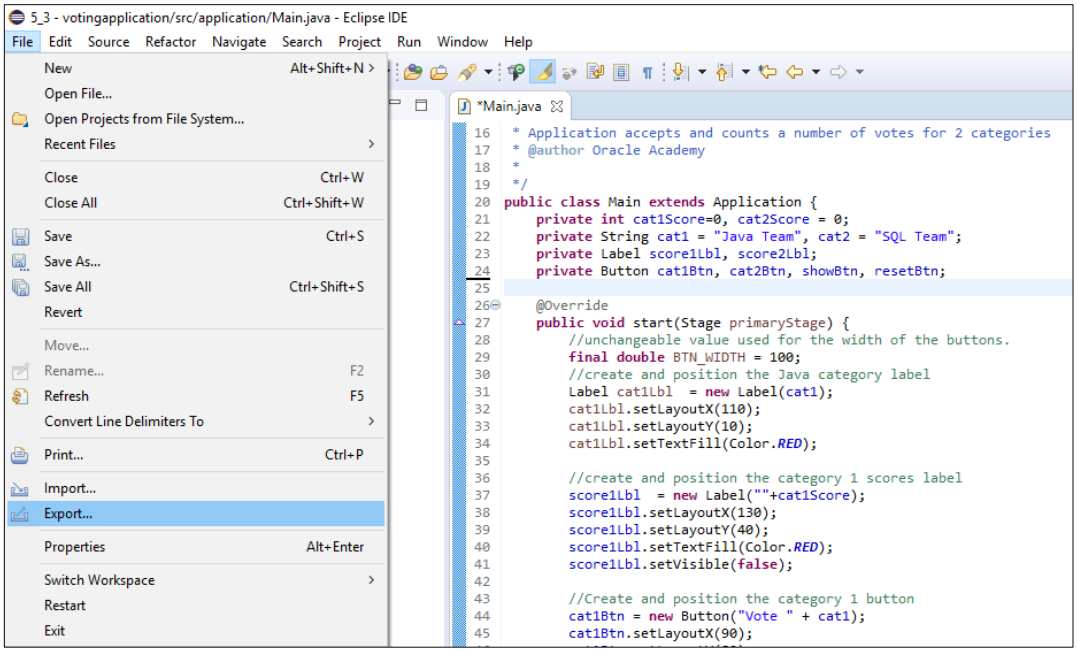
This will disable the two vote buttons and the show button and enable the reset votes button

- In the event handler for the resetBtn call the setBtnUse method passing values so that the two vote buttons and the show votes button are enabled, and the reset button is disabled.
- Run and test your program
- Your application should look like this:

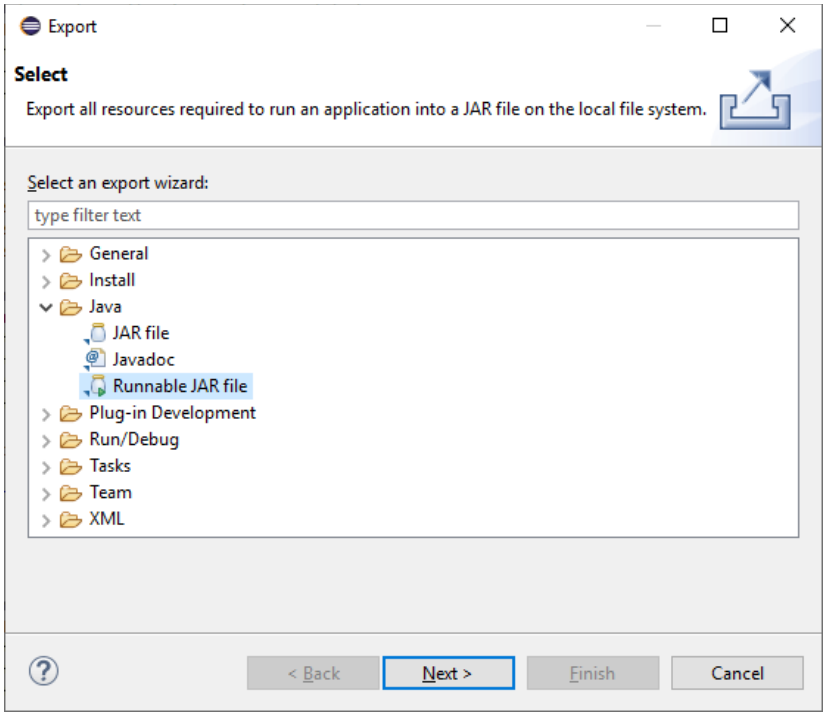


Creating a JAR file for your application.

12. To be able to run the application outside of the Eclipse IDE you need to create a runnable JAR file.
    a. In Eclipse select File, Export.

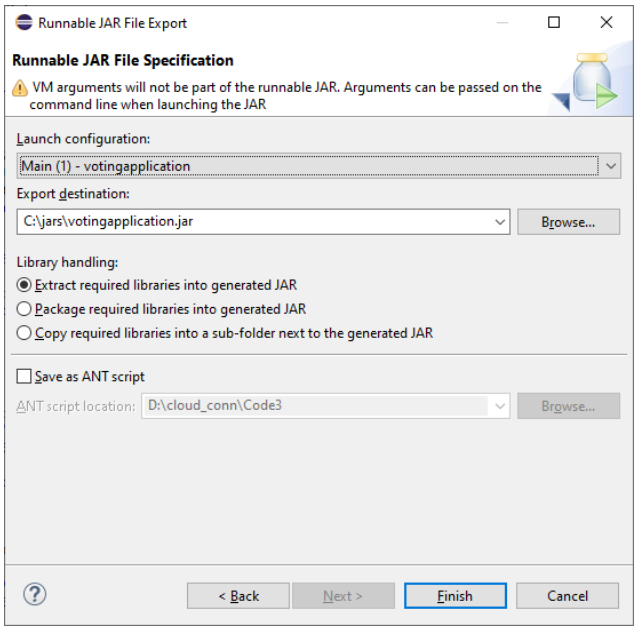


    b. From the Export window select Java and then Runnable JAR file, click Next.

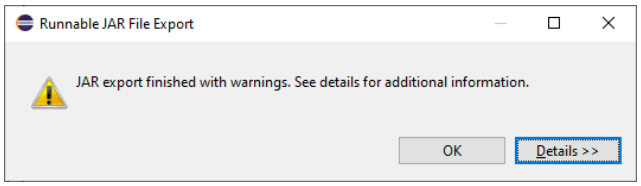


c. In the Launch Configuration box you will see a warning that the VM arguments you added to the run configuration profile in Eclipse will not be added to the JAR file. This will have to be done when running the JAR.
To create the JAR file follow these steps:

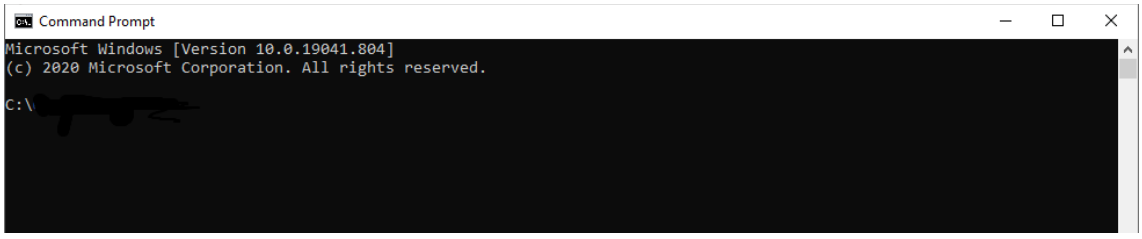
- select the Main class from the application you are JARring up.
- Select the destination directory and name (votingapplication.jar) that you want to give your JAR file
- Click Finish to generate the JAR file
- Click OK to any messages that appear



d. You may get warnings when your JAR file is created, do not worry about these as you will provide the arguments when you run the JAR file.



e. Open the command line to run the JAR file:



f. Type in the following command to run the JAR file (if you saved your fx files or JAR file to a different location then you will need to provide your own path in the command):

```
java --module-path "C:\JavaProgramming\fx\javafx-sdk-15.0.1\lib" --
add-modules javafx.controls,javafx.fxml -jar
C:\jars\votingapplication.jar
```

Here is a breakdown of what the components of the command represent:

`java` – Tells the compiler to run a java program

`--module-path "C:\JavaProgramming\fx\javafx-sdk-15.0.1\lib" --add-modules javafx.controls,javafx.fxml` – the path to the required JavaFX files needed to run the application

`-jar` – command to execute a jar file

`C:\jars\` - the path to the directory where the jar is stored

`votingapplication.jar` – Name of the jar file to be ran

g. You should see your application running outside the development environment!