

Creating a stand-alone Java application using JavaFX

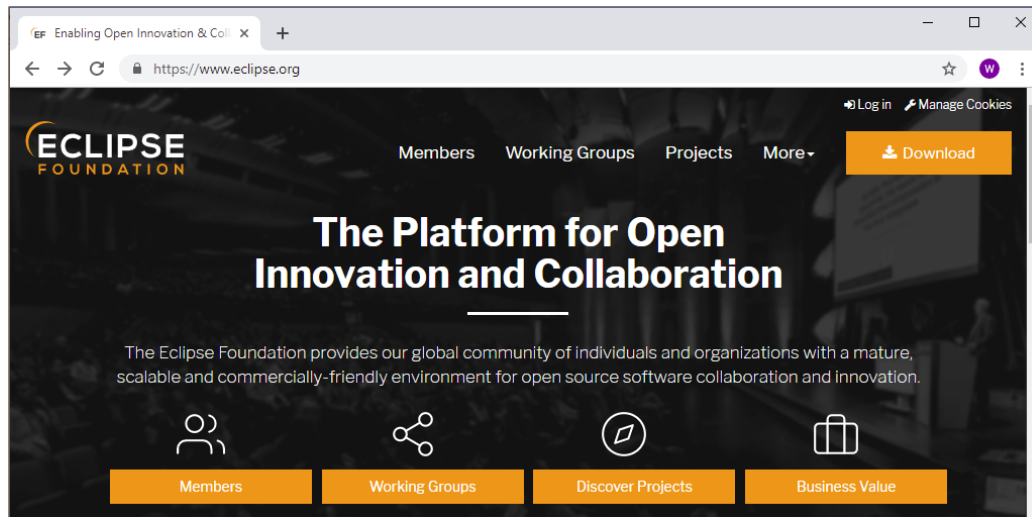


JP_5_3_Practice: Creating a JavaFX application

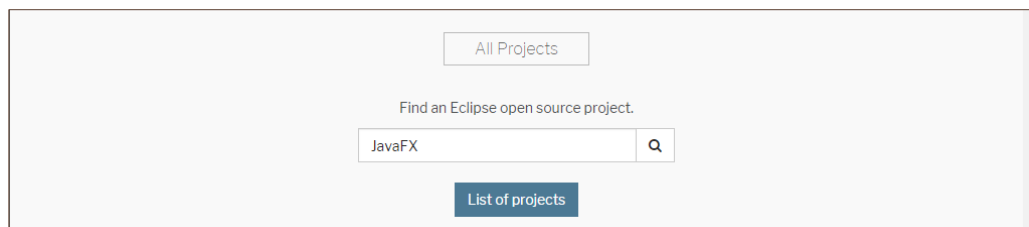
In this tutorial you will learn how to create a Java FX application in eclipse. The application will be a simple voting application that you can create a JAR file of so that it can be ran outside of the development environment.

Installing Java FX in Eclipse.

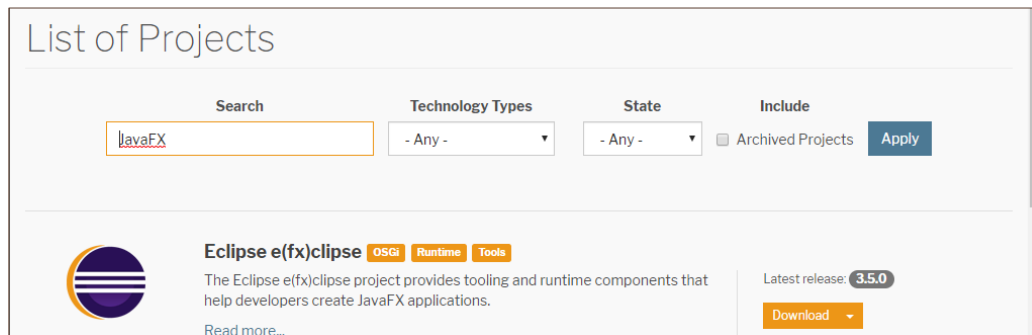
1. You need to get the latest version of JavaFX to install in Eclipse.
 - a. Go to www.eclipse.org in your browser to get the address to download the latest version of FX.
 - b. Click the Download button to access the download center.



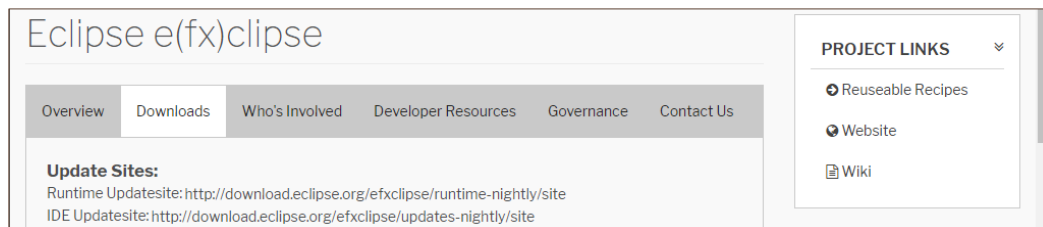
- c. Scroll down to the search bar and type JavaFX as your search criteria before clicking the search button.



d. The JavaFX project will be displayed, click on the Download button.



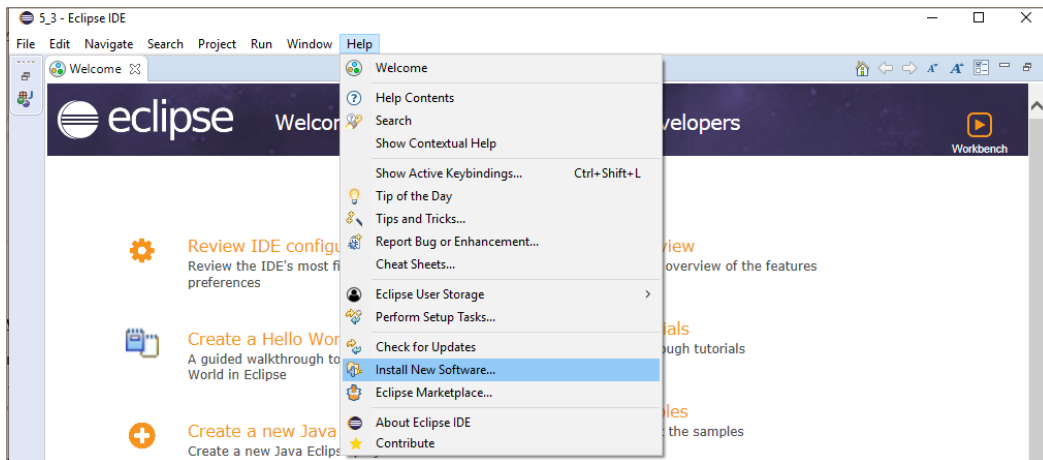
e. The IDE update site address will allow you to easily install JavaFX in Eclipse. Select and copy the address for use in the Eclipse IDE.



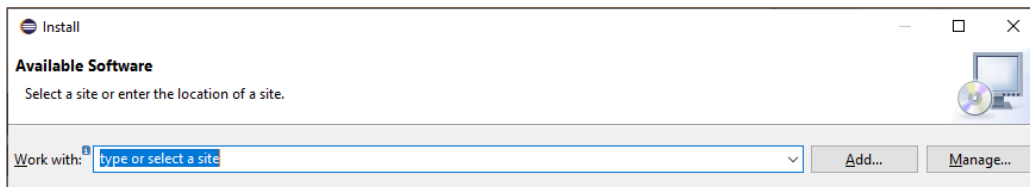
f. At the time of writing the address for JavaFX was:
http://download.eclipse.org/efxclipse/updates-nightly/site
You can copy this address if you can't access the eclipse site.

Adding JavaFX to Eclipse

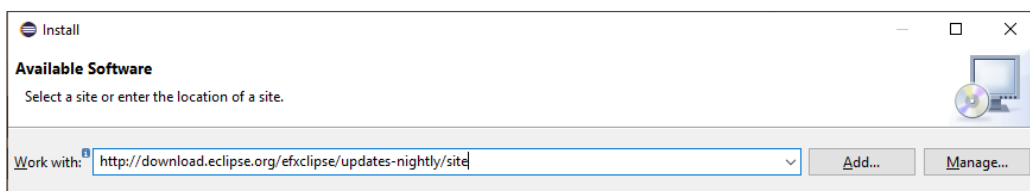
2. FX is not installed by default in Eclipse so must be added.
 - a. Start Eclipse and select help from the menu.
 - b. Select Install new Software from the drop-down menu.



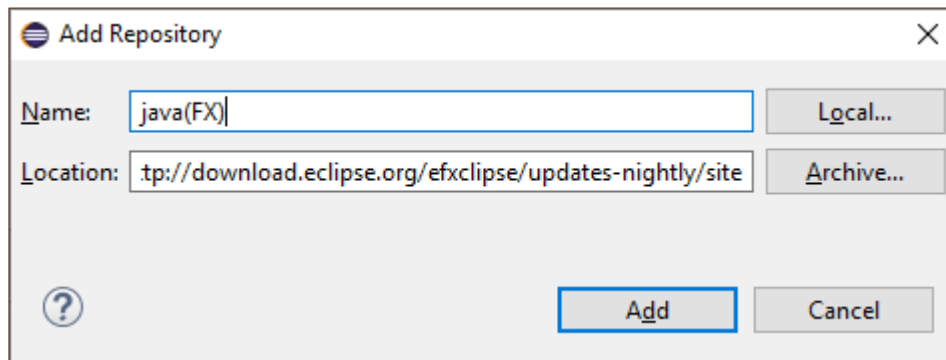
- c. In the work with box that displays a “Type or select site” message paste the IDE Update address that you copied from the Eclipse website.



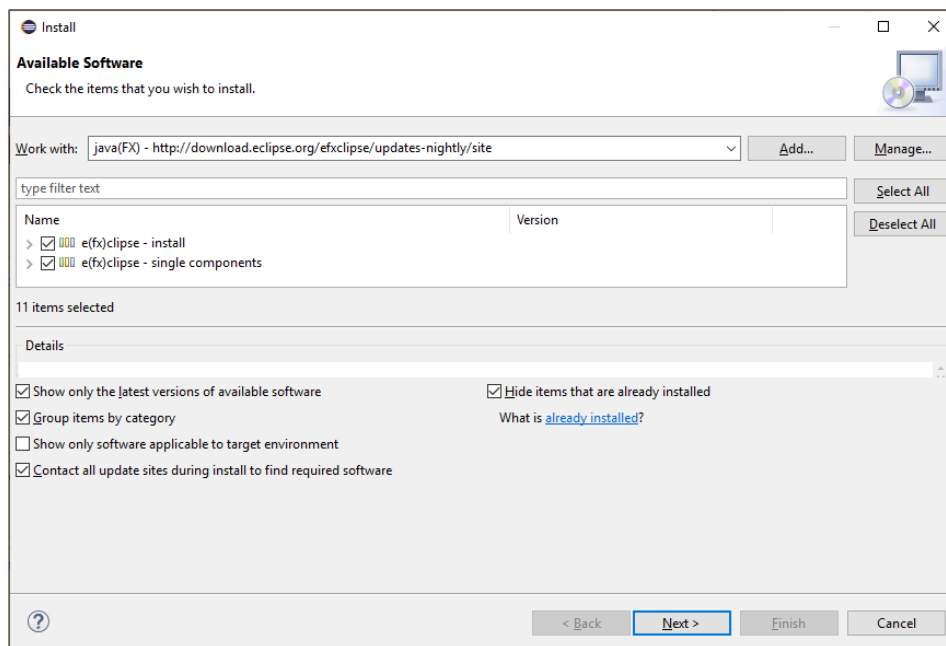
- d. Click the Add button



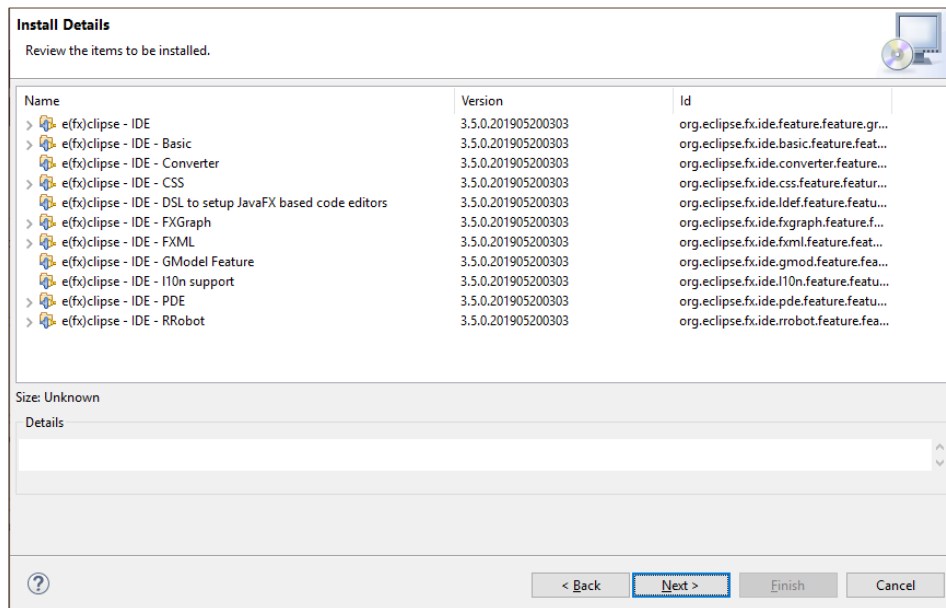
e. Name the repository as java(FX) and click Add.



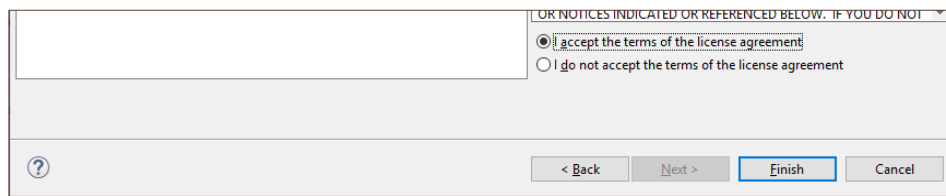
f. Tick both e(fx) boxes to select all of the components and click the Next > button.



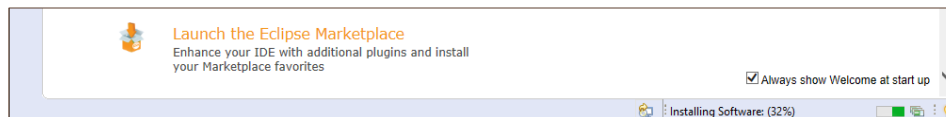
g. You will be shown a list of the items that will be installed. Click Next >.



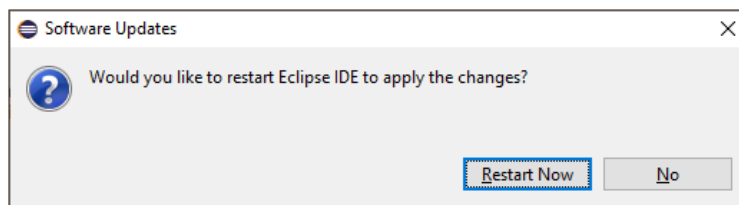
h. Accept the terms of the license agreement and click finish.



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

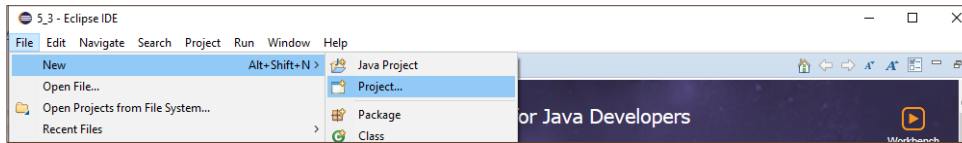


j. When the software is installed you will be asked to restart Eclipse, click Restart now to complete the install.

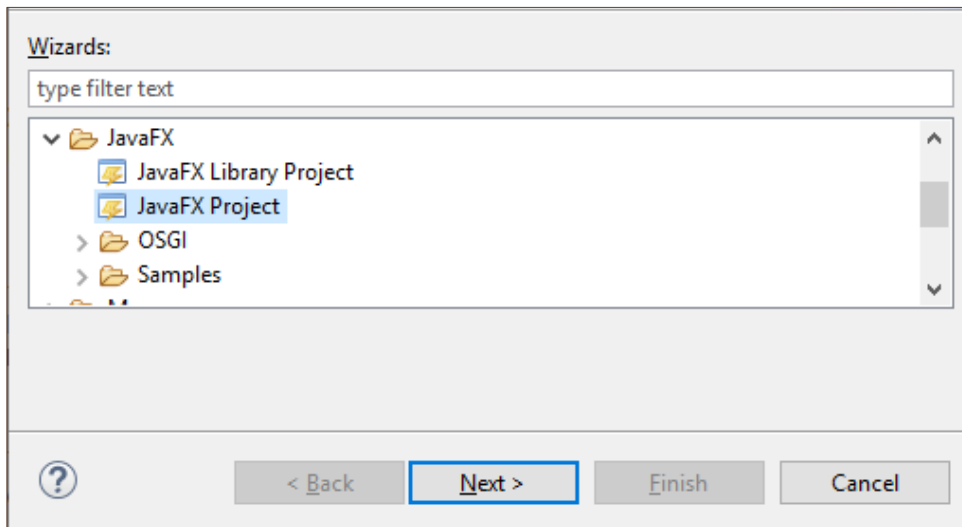


Creating a JavaFX application in Eclipse.

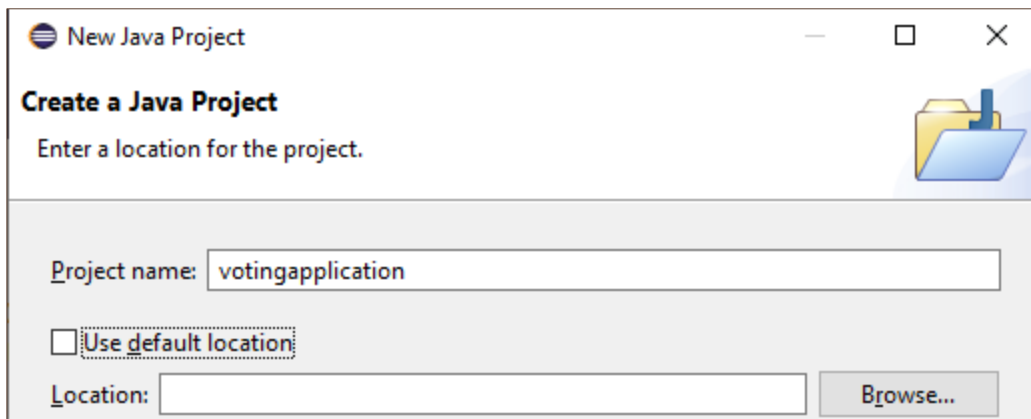
3. To create a JavaFX application in Eclipse you need to create a JavaFx project.
 - a. Select File, New and Project from the menu from the restarted Eclipse IDE.



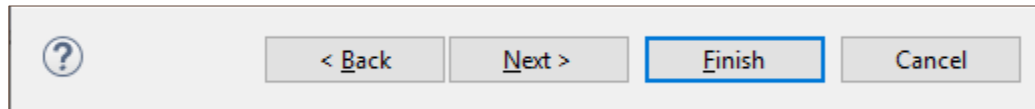
- b. Select JavaFX and then JavaFX Project from the drop-down list and click Next >.



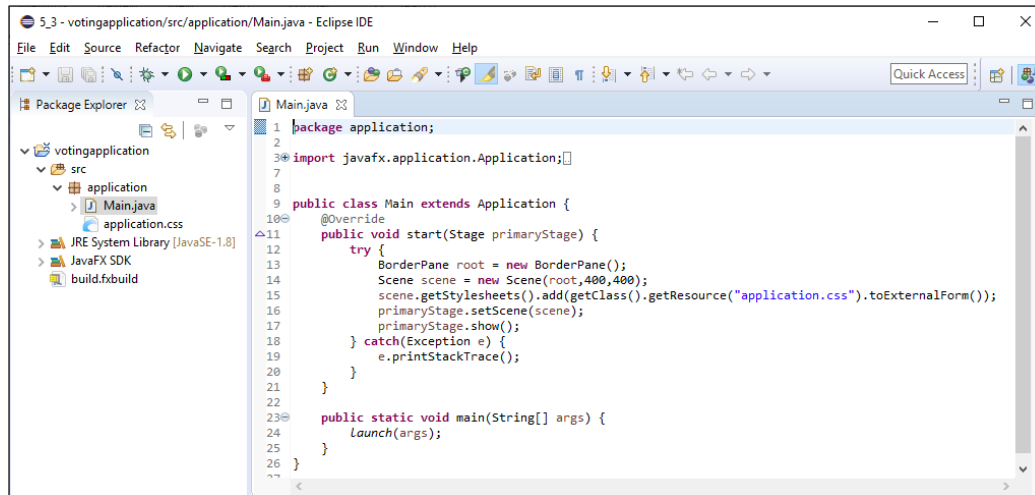
- c. Call the project votingapplication and select a workspace location or tick the box to use the default location.



- d. Leave the settings as their default and click Finish to create the JavaFX project.



- e. 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.



Coding a JavaFX application

4. 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:

```
    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.

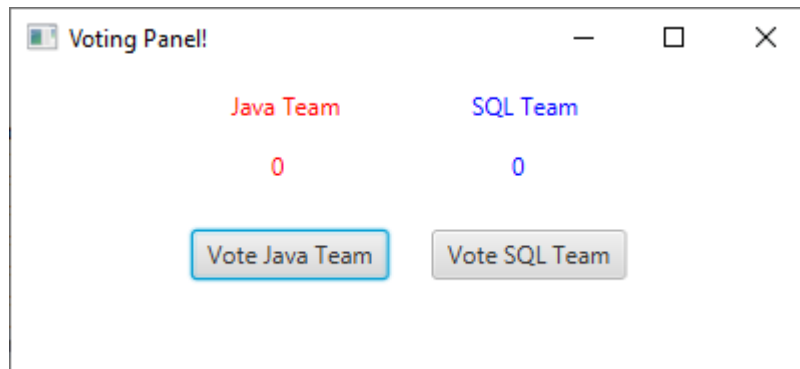
```
primaryStage.setTitle("Voting Panel!");
primaryStage.setMinWidth(scene.getWidth());
primaryStage.setMinHeight(scene.getHeight());
primaryStage.setScene(scene);
```

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

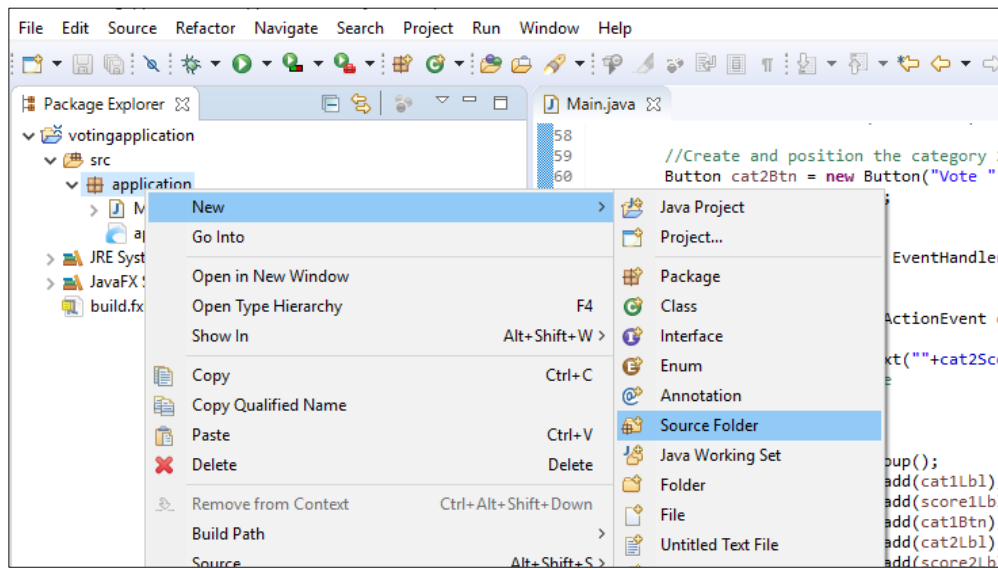
5. Use the following information to create a second set of labels and buttons:

- a. An integer instance field named `cat2Score` that is initialized to 0.
- b. A String instance field 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.

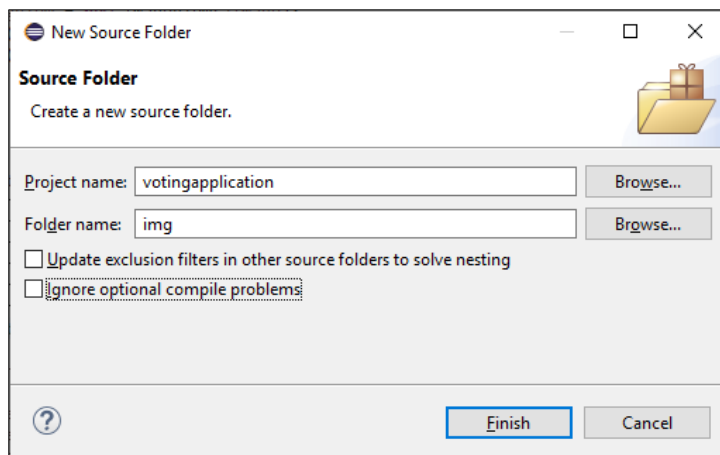
i. Your application should look like this:



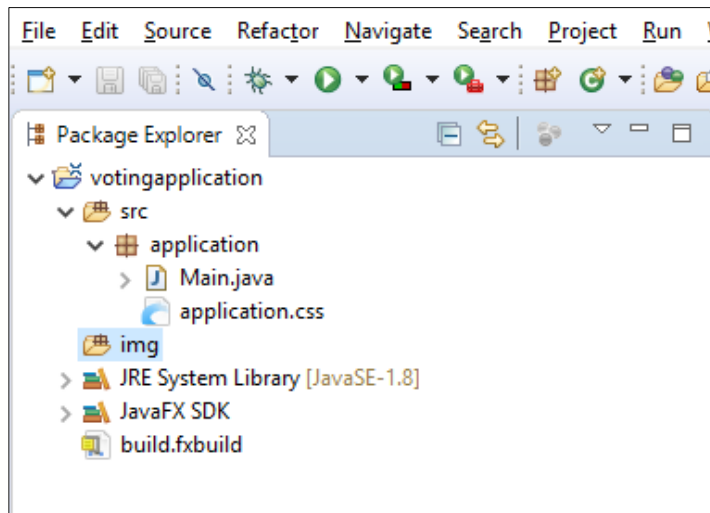
6. You will now add images to the application by including them in a resources folder with the application.
- a. In Eclipse, right click on the application package and select New, Source Folder.



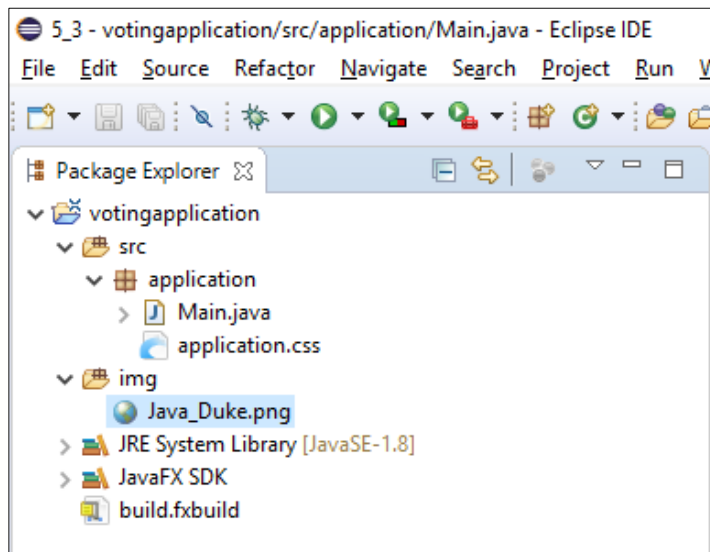
- b. Name the source folder img and then click Finish.



- 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.



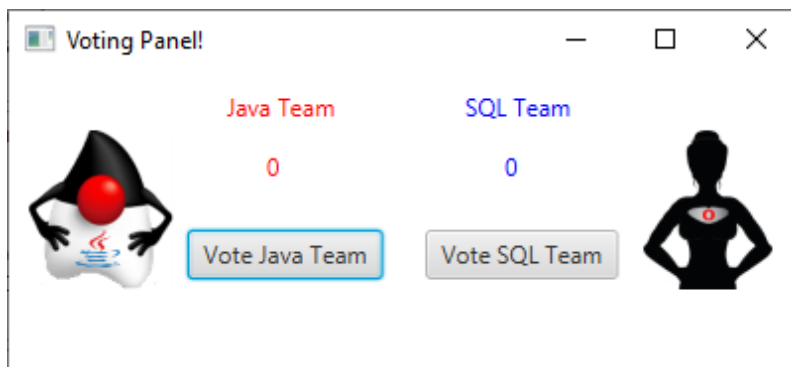
- e. Above the try catch block in your code add the following line of code that creates a reference (jImage) to the Java_Duke image in the img folder. You will need to import the fx image library.

```
//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);
```

7. Use the following information to add an image for the SQL Team vote.
- Drag the Datagirl.png image file into the img source folder.
 - Create a reference to the Datagirl image named sqlImage.
 - Create a sqlImgView using the sqlImage reference.
 - Place the sql image at the following co-ordinates: X- 320, Y-30.
 - Add the image view to the Group layout manager.
 - Your application should look like this:



8. You now have a working app that allows votes to be cast, however the user experience can be improved with a few tweaks.
- 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);
```

- ii. 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
```

- iii. 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;
```

- iv. 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.

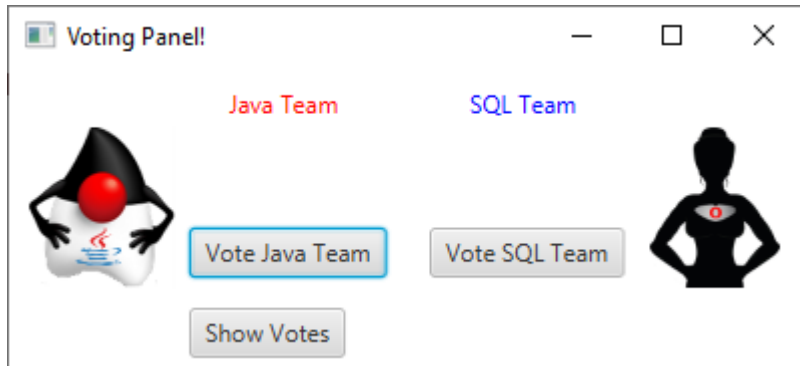
The errors that were attached to the `showVotes()` method should now have been resolved.

- v. 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,
 - i. 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.
 - ii. Call the showVotes() method from its event handler.
 - iii. Run and test your program
 - iv. 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.

- i. 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;
```

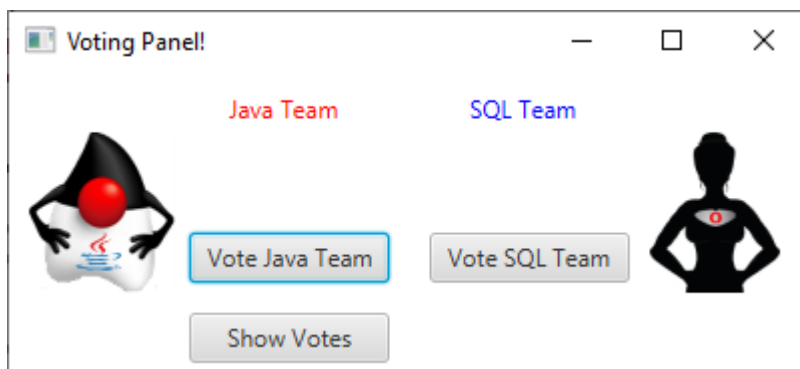
- ii. In the properties list for all of the buttons use the setPrefWidth() method to set the width of the button equal to the BTN_WIDTH value.

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

Do the same for the cat1Btn and the cat2Btn buttons.

- iii. Run your program

- iv. 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.

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

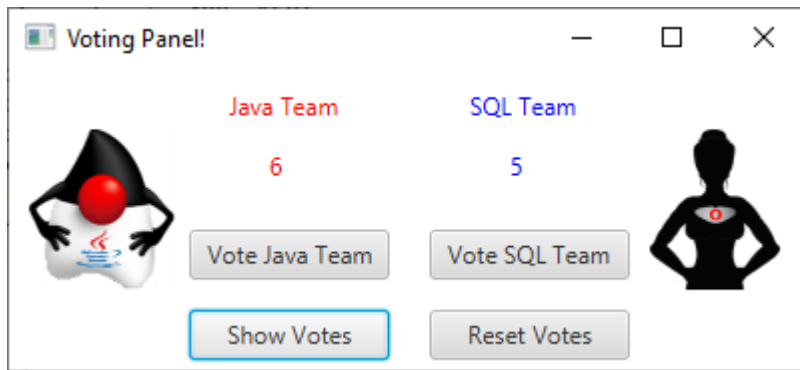
```
private void resetVotes() {  
    cat1Score=0;  
    cat2Score=0;  
    score1Lbl.setVisible(false);  
    score2Lbl.setVisible(false);  
} //end method resetVotes
```

- ii. Create a button named resetBtn that displays "Reset Votes" as its text, it's positioned at 210, 120 and uses the preferred button width.

- iii. Call the resetVotes() method from its event handler.

- iv. Run and test your program

- v. Your application should look like this:



- e. There are a few issues with the application:
 - i. After showing the vote values you can click the vote buttons to add to the totals.
 - ii. You can reset the votes at any time.
 - iii. 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
```

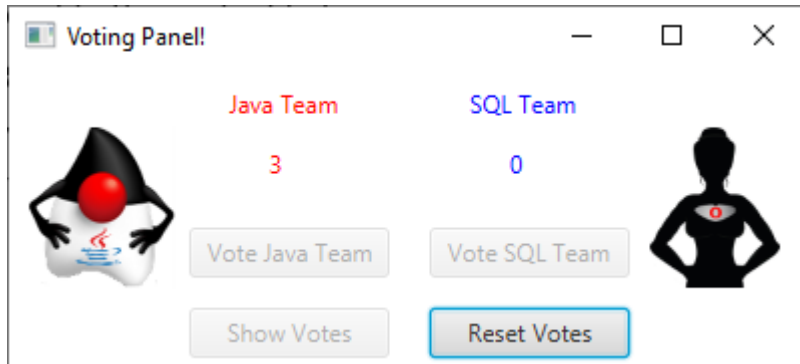
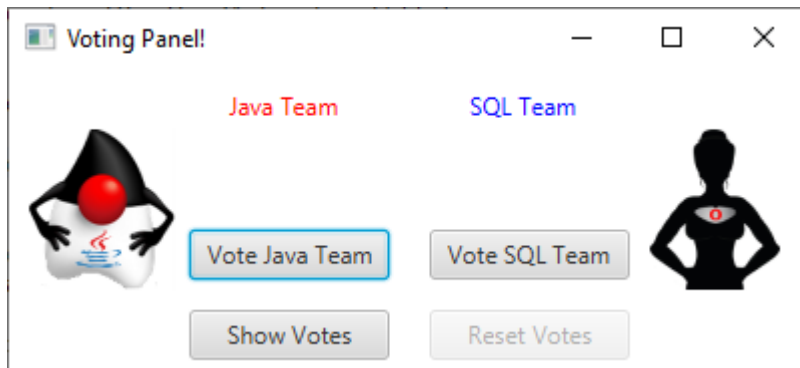
- g. You can fix these issues by simply controlling access to the buttons.
 - i. The reset button should be disabled until the `showVote` button is clicked. When the reset button is created set it to disabled:


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

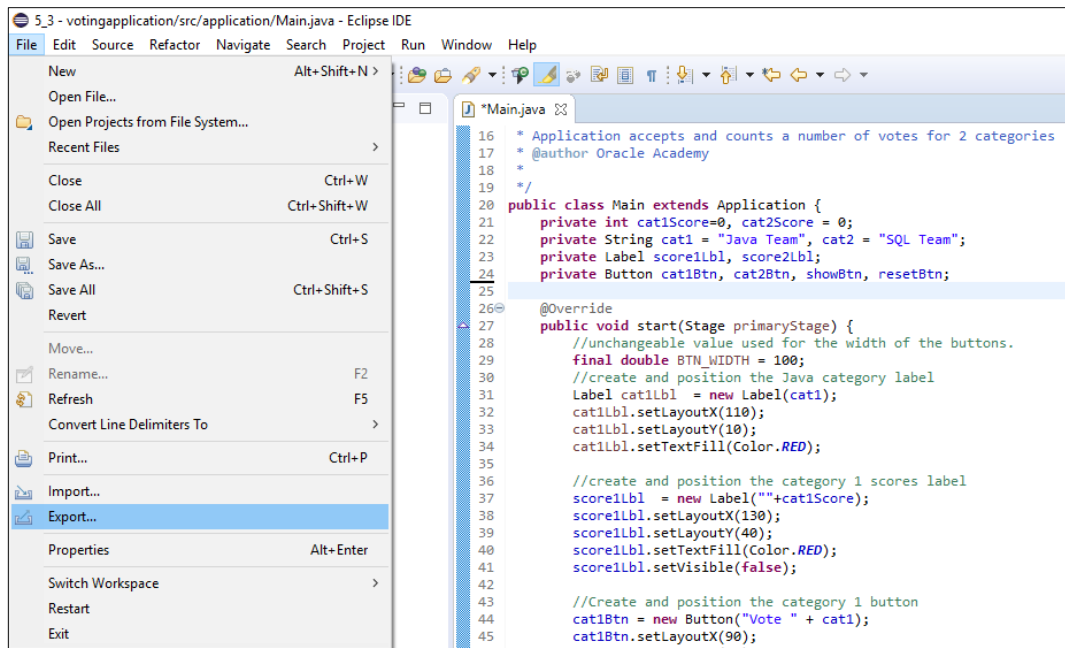
This will disable the two vote buttons and the show button and enable the reset votes button

- iii. 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.
- iv. Run and test your program
- v. Your application should look like this:

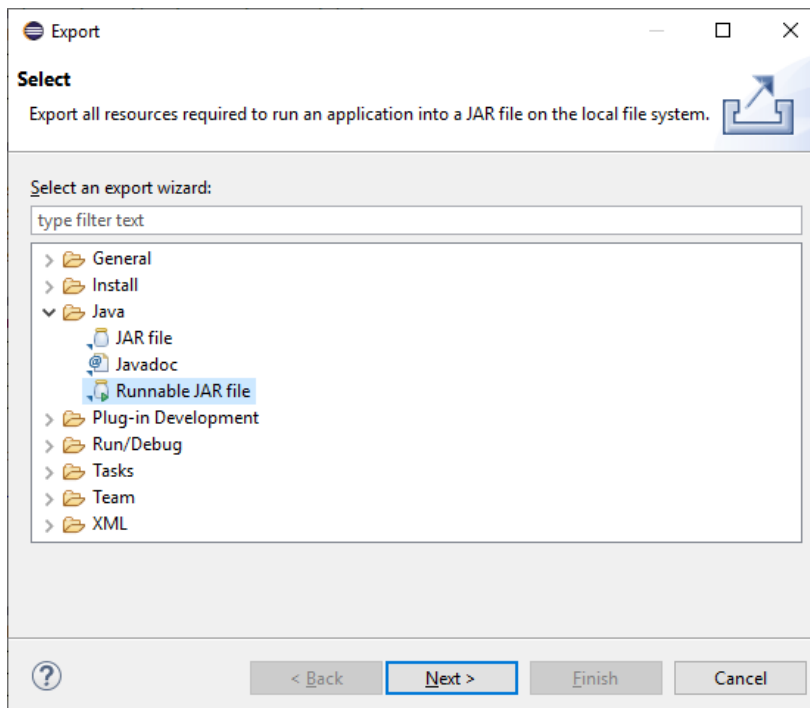


Creating a JAR file for your application.

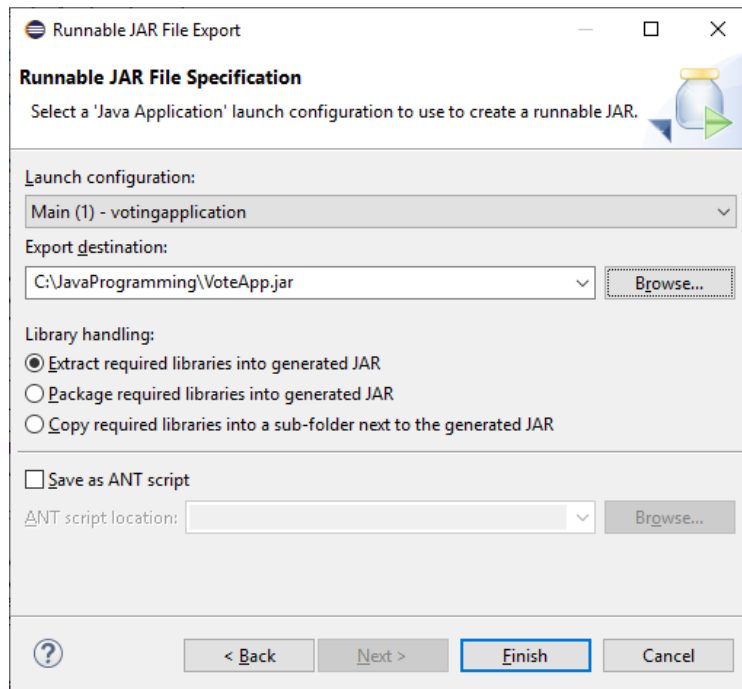
9. To be able to run the application outside of the Eclipse IDE you need to create a runnable JAR file.
 - a. In Eclipse select File, Export.



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



- c. In the Launch Configuration box select the Main class from the application you are JARring up. Select the destination directory and name that you want to give your JAR file. Click Finish to generate the JAR file, click OK to any messages that appear.



- d. Go to your chosen directory, double click the JAR file to launch it!

