Java Programming 1

PROFESSOR: CÂI FILIAULT

Topics for this week

- Objects
- Building GUI’s using JavaFX
- Images
- Scenes, Stages, Nodes

Employment Systems

NEW EMAIL

NEW CLIENT

Email

An email was sent to us this morning asking us to build yet another piece of software

Let's look at the email they sent us:

Hello,

We would like you to build a GUI that can be used in our employee database.

The GUI should contain the employees' picture, name, description.

-Thanks

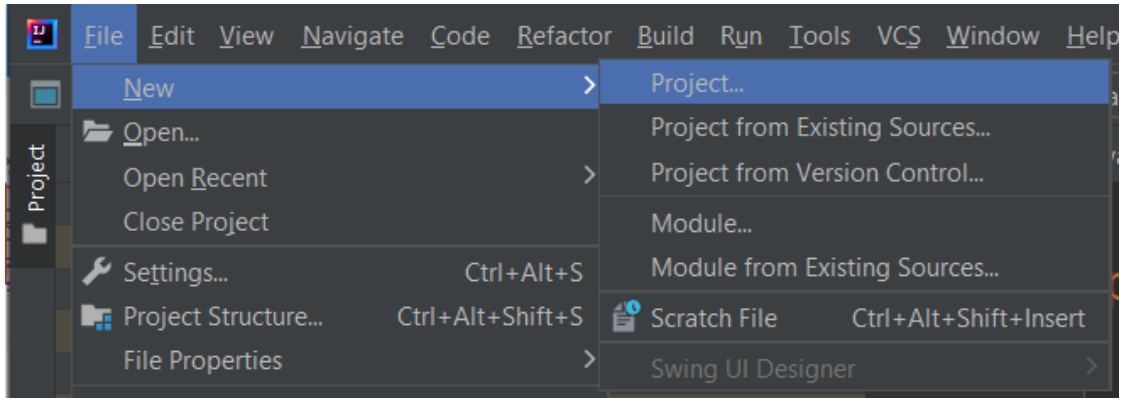
Key Information

- They are looking for us to build a GUI
- The GUI they are looking for us to build must contain the following elements:
  - Image
  - Text
  - They have also identified a specific look they would like for the application
  - We will need to try our best to have the application look exactly like they want it to.

Key Information

Step 1:

- When building a JavaFX project there are a few things steps that we need to follow
- Let's start by creating a new project
- Name the new project EmployeeSystem



New Project

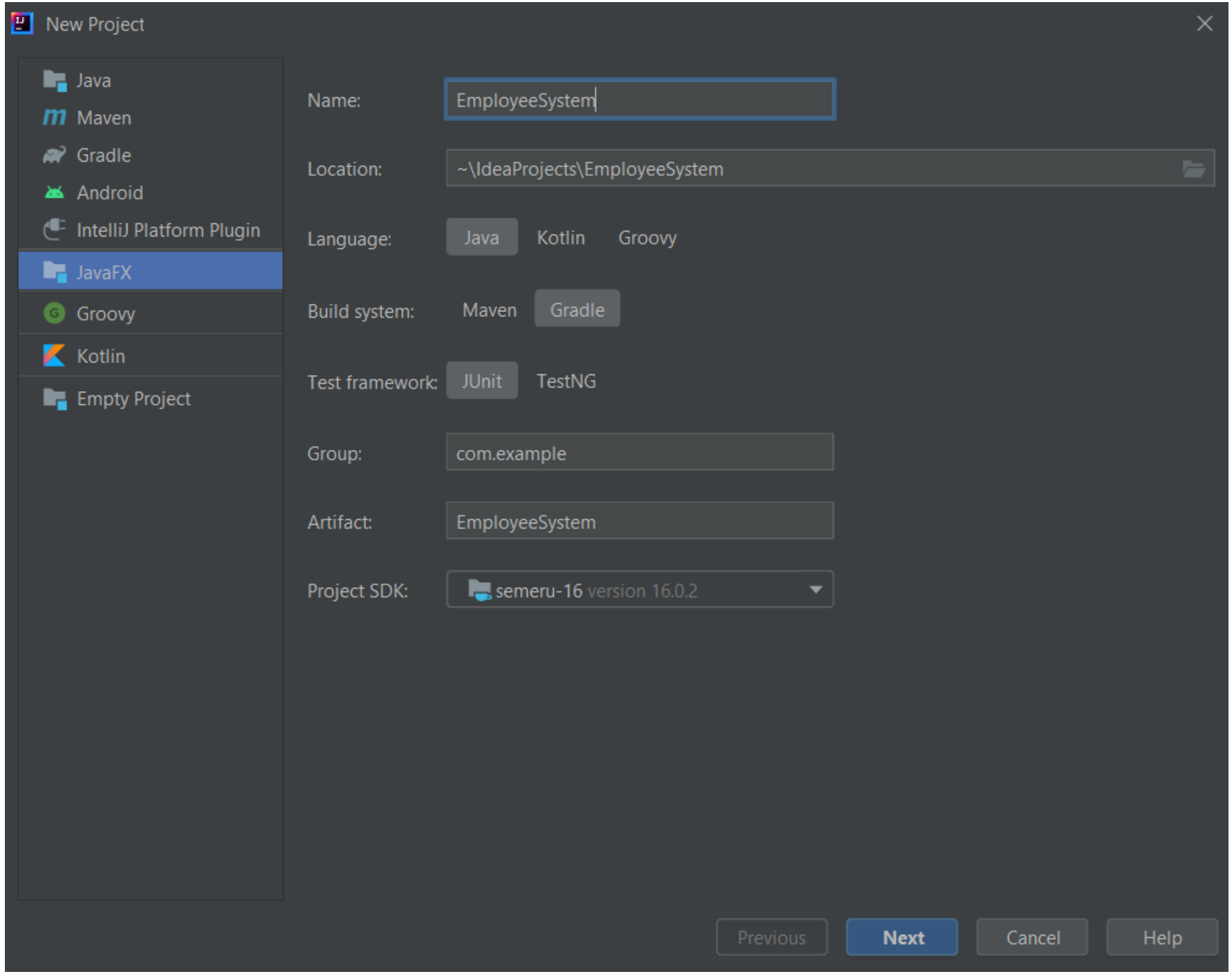
You will now see a dialog box appear *(It may be a little different than the one that I have shown)*

Starting on the left we will select a new JavaFX Project

We will now want to select the project SDK of 16

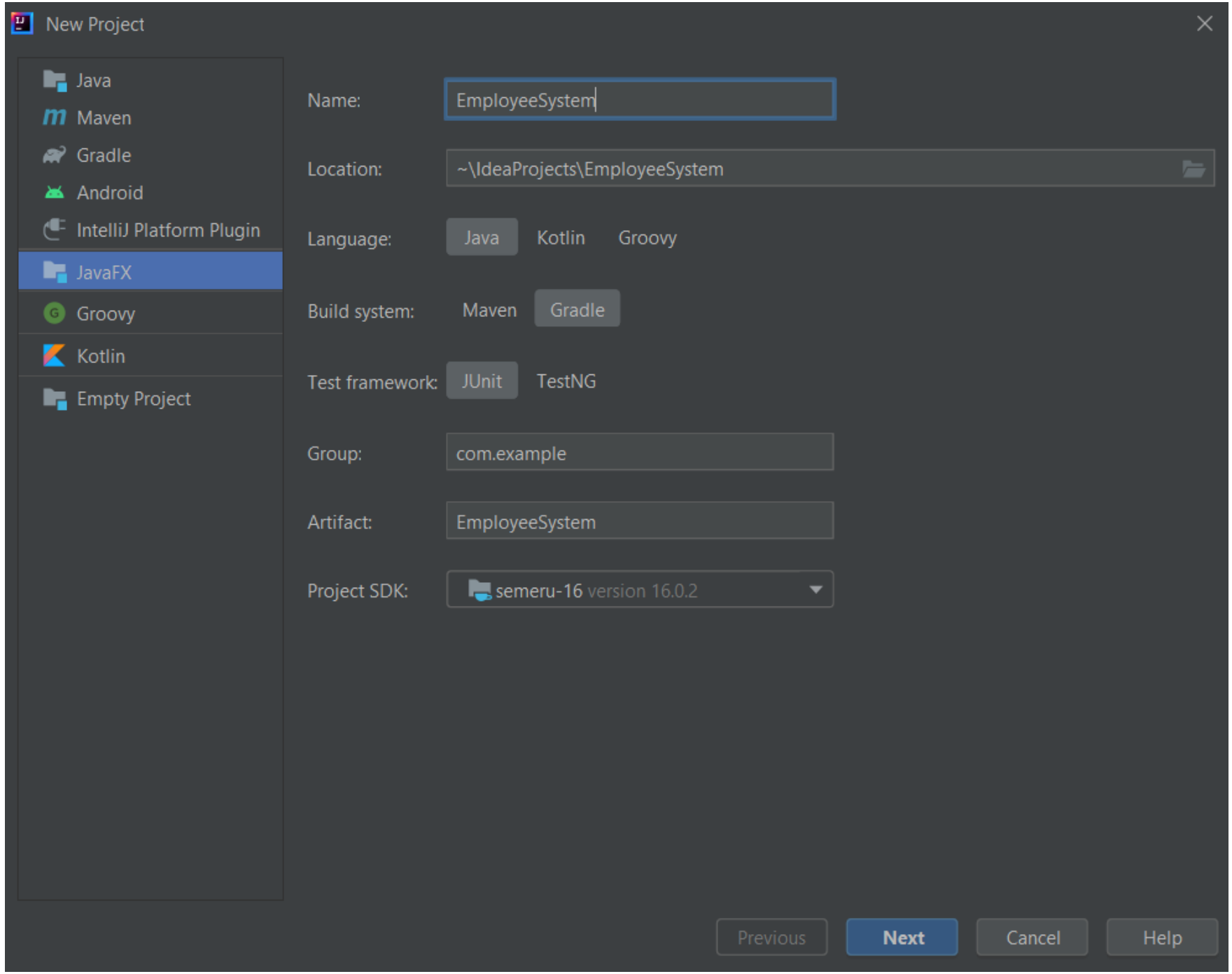
SDK stands for Software Development Kit, and it dictates which version of Java we are using.

At this point we can select Gradle as the build system



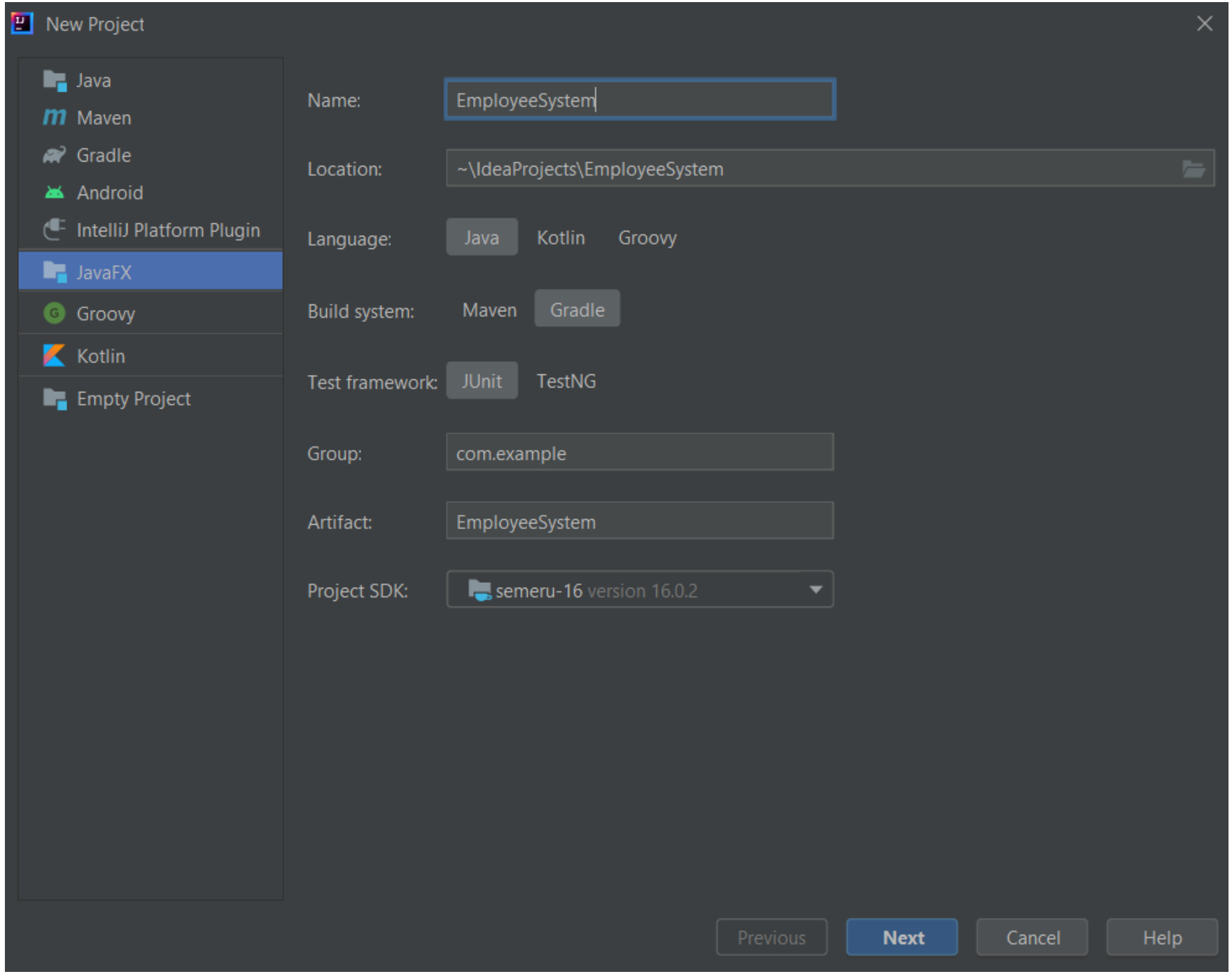
New Project

- Gradle is a dependency manager that permits us to work with other libraries
- After Java version 1.8 (8) JavaFX is no longer bundled with the JDK and needs to added manually.
- A dependency manager is used to automatically pull in different libraries that may be required for your project
- Many dependency managers exist but the two most popular in Java are Maven and Gradle.



New Project

- Next, we select next and finish



Creating a new project

- **Let's add the following comments to our code:**
- We need to add a start method
- Within that start method we need to create the appropriate panes that will layout the software correctly
- We then need to create a scene and add those panes to the scene
- Then we will add the scene to a stage
- Followed by setting the title and showing the stage

```
3  //have the program extend the application class
4  public class EmployeeDirectory {
5
6      //create the start method
7
8      //create the appropriate panes to outline the look of the program
9
10     //create a scene
11
12     //add the panes to the scene
13
14     //add the scene to the stage
15
16     //set the stage title
17
18     //show the stage
19
20     public static void main(String[] args) {
21         // TODO Auto-generated method stub
22
23     }
24 }
```

Creating a new class file

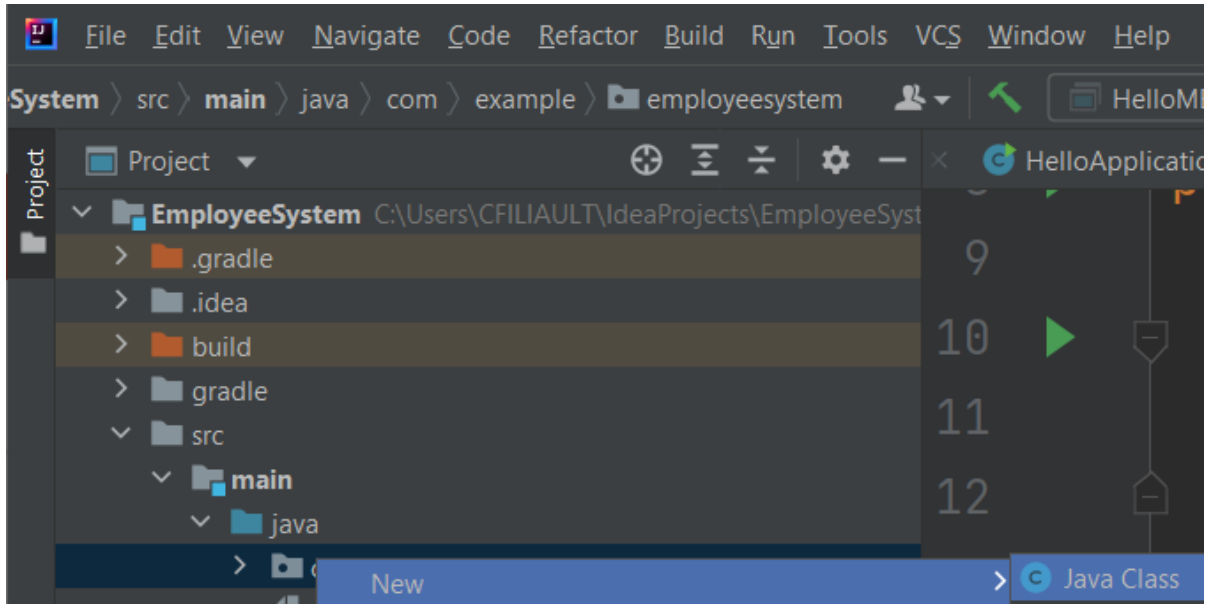
- Before we can begin, we need to review some basic concepts about GUI's
- Let's start with a simple GUI that displays your name to the screen
- Let's create a new class called HelloMe that we will review simple GUI concepts before we dig into this program

Right click on the **com.example.emploeesystem** folder and select:

New> Java Class

Creating a new class is like creating a new program

Name the class **HelloMe** and hit **enter**



Employment Systems Inc.

- Let's start by adding some basic comments to this document as well
- We will explain each of these concepts in the comments the following slides

```
public class HelloMe {

    public static void main(String args[])


    }


    //create the start method

    //create a simple pane

    //create some nodes

    //add the nodes to the pane

    //create a scene

    //add the pane to the scene

    //add the scene to the stage

    //set the stage title

    //show the stage

}
```

Employment Systems Inc.

- The first thing we need to do is have that application extend the application class
- The application class is an abstract class that has some contracts that will need to be filled
- Hover your mouse over application and add the imports
- Hover your mouse over HelloMe and add the unimplemented methods
- The unimplemented method would be the start method
- Take a second to properly organize your comments so that the start method is comment is on the outside and all other comments are on the inside
- The next thing we need to do is change the parameter inside of the start method

```
5  public class HelloMe extends Application {
6
7      public static void main(String args[]){
8
9
10     }
11
12     //create the start method
13     public void start(Stage arg0) throws Exception {
14
15         //create a simple pane
16
17         //create some nodes
18
19         //add the nodes to the pane
20
21         //create a scene
22
23         //add the pane to the scene
24
25         //add the scene to the stage
26
27         //set the stage title
28
29         //show the stage
30
31     }
32
33 }
```

Employment Systems Inc.

- Change the Args0 to primaryStage
- It's easier to work with primaryStage than args0

```
//create the start method
public void start(Stage primaryStage){
```

Employment Systems Inc.

- Next, we need to create a pane
- Let's take a second to discuss the different panes and their purpose
- The first pane we create is a StackPane
- StackPane's are used to store nodes
- Nodes can be other panes, buttons, images, checkboxes, combo boxes, password fields etc.
- There are lots of different nodes that can be created
- The purpose of a pane is to store all these nodes
- The nodes that get stored are stored differently depending on the type of pane that is used

```
14⊖    public void start(Stage primaryStage){
15
16         //create a simple pane
17         StackPane pane = new StackPane();
18         //create some nodes
19
20         //add the nodes to the pane
```

Employment Systems Inc.

- Ideally, we have one pane the stores all the contents of our screen.
- We give this node the name root

```
14⊖    public void start(Stage primaryStage){
15
16         //create a simple pane
17         StackPane pane = new StackPane();
18         //create some nodes
19
20         //add the nodes to the pane
```

Employment Systems Inc.

StackPane:

- A StackPane is a pane that groups all nodes together stacking them one on top of each other in the order that they appear.
- A StackPane would be useful if you wanted to have a label on top of an object

Employment Systems Inc.

FlowPane:

- A FlowPane places nodes row by row either horizontally or column by column vertically
- This can be customized by setting the panes orientation
- A flow plane is good to use when you want an objects one on top of each other or listed

Employment Systems Inc.

GridPane:

- A GridPane places the nodes in the cells of a two-dimension grid.
- You have probably heard of building websites out of tables
- This is a similar concept in GUI building
- The default is a 2x3 grid (table) that you can fill the cells with content
- Columns can be added on the fly (3x3)
- This is a good way of orienting information into a grid like layout (Calculator?)

Employment Systems Inc.

GridPane:

- A GridPane places the nodes in the cells of a two-dimension grid.
- You have probably heard of building websites out of tables
- This is a similar concept in GUI building
- The default is a 2x3 grid (table) that you can fill the cells with content
- Columns can be added on the fly (3x3)
- This is a good way of orienting information into a grid like layout (Calculator?)

Employment Systems Inc.

BorderPane:

- A BorderPane is used to layout information in different section of the screen
- The BorderPane allows for the use of 5 main sections of the screen
- The top, right, bottom, left and center.
- When adding nodes to this pane you need to specify what section you are adding them into
- This is probably the easiest and most used pane

Employment Systems Inc.

HBox

- A HBox places the nodes in a single row

VBox

- A VBox places the nodes in a single column

Employment Systems Inc.

- Next, we need to create a label
- A label is a node that can be used to display text to the screen
- After create the label you will need to hover your mouse over label and import the label
- Next, we will use the methods that are associated with the label object to alter the color
- This is expecting us to provide it with a color and we use the constants that are created within the color class.
- We can see that I have selected the Color.BLUE
- Select whichever color you like

```
14  @Override
15  public void start(Stage primaryStage) throws Exception {
16      StackPane pane = new StackPane();
17      Text name = new Text("Cai Filiault");
18      name.setFill(Color.BLUE);
```

Employment Systems Inc.

- Next, we want to add the node to the pane
- We use the getChildren().add() method to put nodes in the plane
- The getChildren() method returns an instance of the javafx.collections.ObservableList which is a class that works in a similar manner to the ArrayList where it can store a collection of elements
- We can see that we return the instance then add a node to it

```
29          //add the nodes to the pane
30          pane.getChildren().add(name);
31
```

Employment Systems Inc.

- The next step is to create a scene
- A scene is used to occupy a stage
- A scene stores within it 1 node, the node that is displayed will occupy the entire scene
- Recall that panes are scenes and store within them multiple panes, nodes, etc.
- Therefore, we will store within the scene a pane
- The pane will contain the remainder of the content
- We will also define the dimensions of the scene for when it gets displayed to the stage

```
33        //create a scene
34        Scene scene = new Scene(pane, 200, 200);
35
```

Employment Systems Inc.

- Next, we see we set the stage to have the scene we created
- The primary stage is an application window
- Each java application could have many stages that could be displayed
- In this case we just have the one stage
- We can see that we have the primary stage set to the scene of scene
- We then set the title to “My Name is”
- Followed by setting the stage to visible using the show() method.

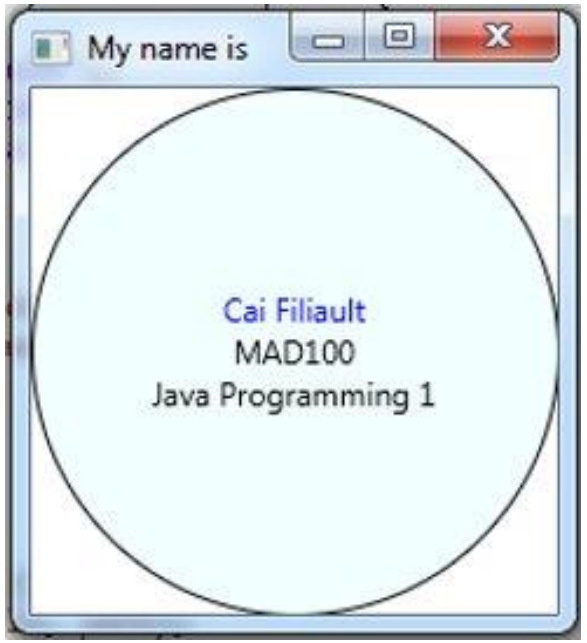
```
28          //create a scene
29          //add the pane to the scene
30          Scene scene = new Scene(pane, 200, 200);
31
32          //add the scene to the stage
33          primaryStage.setScene(scene);
34          //set the stage title
35          primaryStage.setTitle("My Name is");
36          //show the stage
37          primaryStage.show();
38
39      }
```

Employment Systems Inc.

Take a moment to view your application

Writing the program

- Let's make some more modifications to use some Java Scene shapes that exist.
- We will make some more modifications to have our software look like the one shown on the right



Writing the program

- The first step is to create a new shape
- JavaFX has a wide variety of shapes that it supports and several different functionalities that they can provide to your code
- They are Text, Line, Rectangle, Circle, Ellipse, Arc, Polygon and Ployline
- Initially this week we will be using labels for all our text needs. This is not necessarily correct, but it will serve the purpose of creating a simple GUI

```
Circle circle = new Circle();
circle.setRadius(100);
circle.setStroke(Color.BLACK);
circle.setFill(Color.AZURE);
```

Writing the program

- We set the radius, stoke and fill color for the circle
- We then next create a StackPane and set the children of the StackPane to the circle and original pane we created
- We then set the alignment of the StackPane to center and set the scenes pane to the new pane we created
- The final step is setting the VBox to have its alignment to center as well
- Save and deploy the project

```
Circle circle = new Circle();
circle.setRadius(100);
circle.setStroke(Color.BLACK);
circle.setFill(Color.AZURE);
```

```
StackPane pane2 = new StackPane();
pane2.getChildren().addAll(circle, pane);
pane2.setAlignment(Pos.CENTER);
//create a scene
//add the pane to the scene
Scene scene = new Scene(pane2, 200, 200);
```

```
pane.setAlignment(Pos.CENTER);
```

Writing the program

- Let's navigate back to the EmployeeDirectory program
- Whenever we create an application the first thing, we need to do is have that application extend the application class
- Hover your mouse over application and add the imports
- Hover your mouse over EmoployeeDirectory and add the unimplemented methods

```
6  //have the program extend the application class
7  public class EmployeeDirectory extends Application{
8
9
10     public static void main(String[] args) {
11         // TODO Auto-generated method stub
12
13     }
14
15     //create the start method
16     public void start(Stage arg0) throws Exception {
17
18
19         //create the appropriate panes to outline the look of the program
20
21         //create a scene
22
23         //add the panes to the scene
24
25         //add the scene to the stage
26
27         //set the stage title
28
29         //show the stage
30     }
```

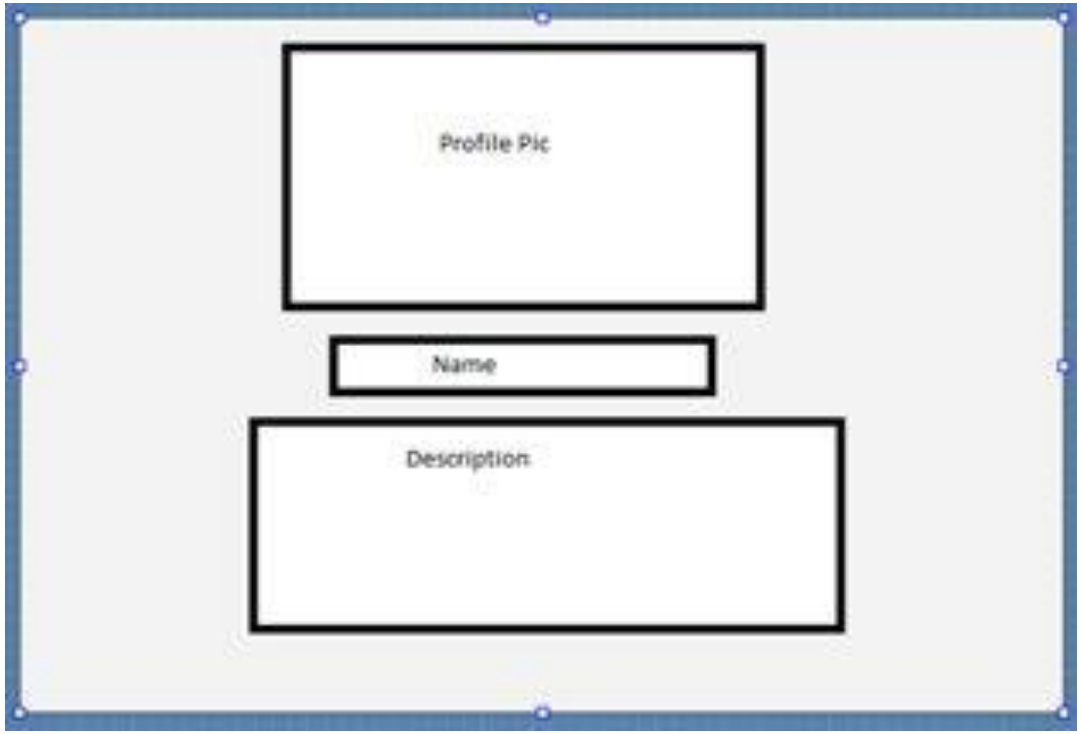
Writing the program

- The unimplemented method would be the start method
- Take a second to properly organize your comments so that the start method is comment is on the outside and all other comments are on the inside
- The next thing we need to do is change the parameter inside of the start method

```
6  //have the program extend the application class
7  public class EmployeeDirectory extends Application{
8
9
10     public static void main(String[] args) {
11         // TODO Auto-generated method stub
12
13     }
14
15     //create the start method
16     public void start(Stage arg0) throws Exception {
17
18
19         //create the appropriate panes to outline the look of the program
20
21         //create a scene
22
23         //add the panes to the scene
24
25         //add the scene to the stage
26
27         //set the stage title
28
29         //show the stage
30     }
```

Writing the program

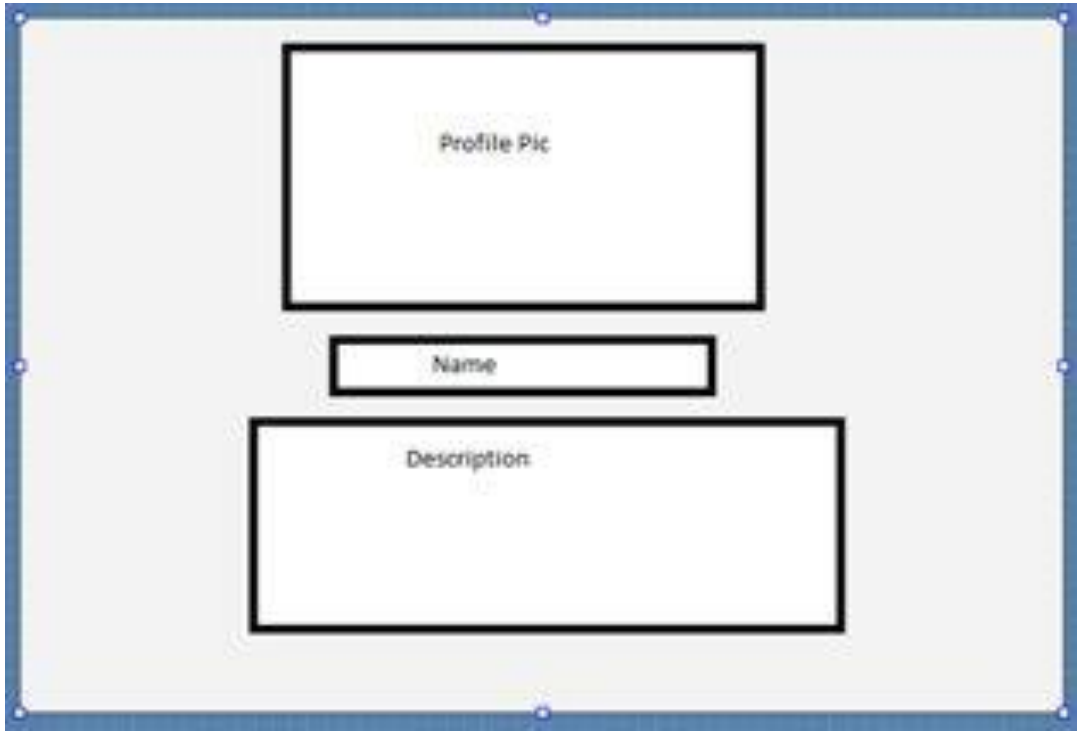
- Next, we need to create a pane
- I have preemptively drawn us a general layout that we would like for our application
- We want the top of our program to have a profile picture
- The center of the program should have the user's name
- The bottom of the program should have the description
- What pane do you think we should use?



Writing the program

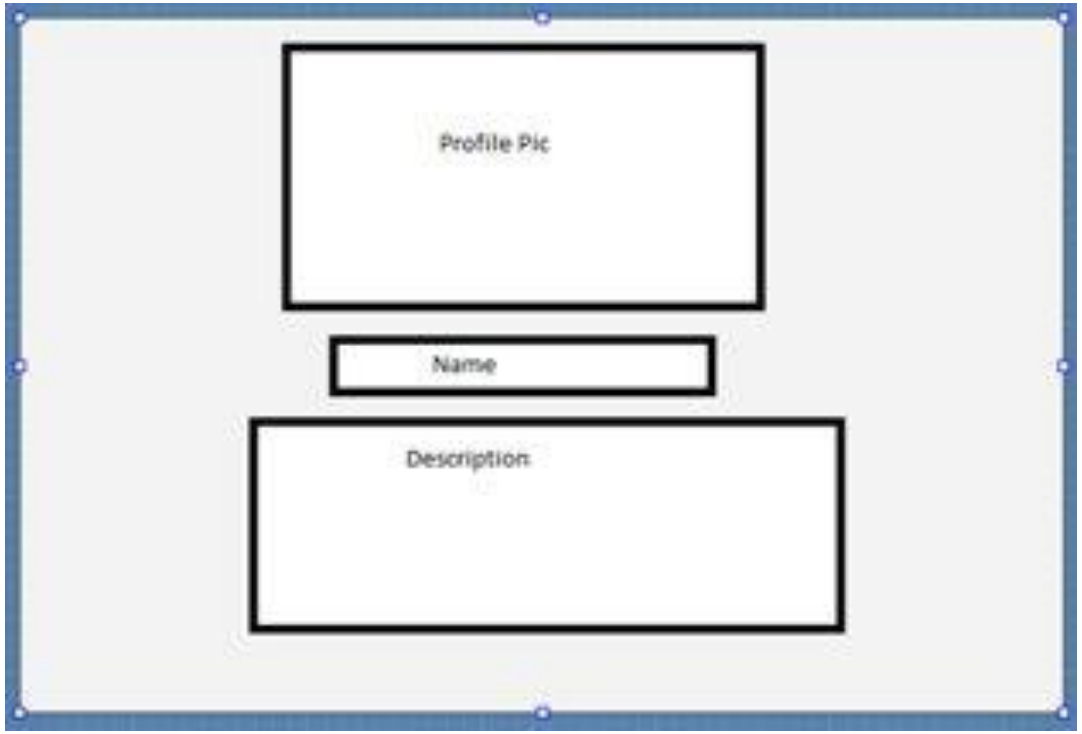
we should be using a BorderPane

- The BordePane allows us to put content in the top, center and bottom of the pane
- We can leave the left and right for right now because we do not require to put content there.
- Inside the top of the BorderPane ImageView (we will discuss this in a moment)
- In the center we are going to have a single text object to store the user's name



Writing the program

- Next, in the bottom of the BorderPane we will be storing a description



Writing the program

- We start by creating an Image takes as a parameter the location of the image we want to display to the screen.
- We then create an ImageView. The ImageView is like a picture frame where all it does is host an image.
- **I have my own image, take 5 minutes to locate an image of yourself (I assume everyone can grab one from the internet)**
- If you cannot grab an image of yourself just pic a celebrity from the internet.
- Name your image profilepic.jpg and put it inside of the **res** folder of your project

```java
    public void start(Stage primaryStage) throws Exception {
        /**
         * Create a BorderPane
         * We know that a BorderPane is divided up into different sections on the screen.
         * We will be using the top, center and bottom of the pane
         */
        BorderPane pane = new BorderPane();

        /**
         * When we want to display an image to the screen we do so in an
         * ImageView. Image views store inside of them Image objects.
         * In the following code we Create an ImageView and store within
         * it an image named profilepic.jpg
         * We then set the width and height of the image.
         * The default measurement used in JavaFX is pixel
         */
        Image pic = new Image("profilepic.jpg");
        ImageView profilePic = new ImageView(pic);
        profilePic.setFitHeight(350);
        profilePic.setFitWidth(400);
```

Writing the program

- Once you have located the image of yourself copy it from your computer
- Right-click on your res folder and paste the image in.
- Replace in your code “profilepic.jpg” with whatever the file name of your own picture is.
- I suggest shortening the image name before pasting it into your project.
- We then set the dimensions of the photo.
- (this may distort the look of your photo, but we will worry about this afterwards)

```java
    public void start(Stage primaryStage) throws Exception {
        /**
         * Create a BorderPane
         * We know that a BorderPane is divided up into different sections on the screen.
         * We will be using the top, center and bottom of the pane
         */
        BorderPane pane = new BorderPane();

        /**
         * When we want to display an image to the screen we do so in an
         * ImageView. Image views store inside of them Image objects.
         * In the following code we Create an ImageView and store within
         * it an image named profilepic.jpg
         * We then set the width and height of the image.
         * The default measurement used in JavaFX is pixel
         */
        Image pic = new Image("profilepic.jpg");
        ImageView profilePic = new ImageView(pic);
        profilePic.setFitHeight(350);
        profilePic.setFitWidth(400);
```

Writing the program

- Next, we create a Text object
- That text object contains within it our name.
- Please use your own name. (if you selected a photo of a celebrity, use the name of that celebrity)
- We then want to change the size and font family of the name that gets displayed to the screen
- We first use the setFont(Font.font())
- The setFont() method expects a font parameter to be passed
- We use the font method located in the font class to generate a font that can be used by the setFont()
  - We need to specify the font, weight, posture and size.
  - We will use times new roman
  - A weight and posture of normal
  - A font size of 30

```
41          /**
42           * Next we create the name of the employee we want to display
43           * to the screen. In this case I have used myself. Feel free
44           * to place your own name inside of this Text object
45           * We then set the Font to a new Font Object
46           * We pass the font object the parameters for font family
47           * weight, posture and size. Finally we set the fill color to
48           * blue
49           */
50          Text name = new Text("Cai Filiault");
51          name.setFont(Font.font("Times New Roman",
52                  FontWeight.BOLD, FontPosture.REGULAR, 40));
53          name.setFill(Color.BLUE);
54
```

Writing the program

- Next, we create the BorderPane
- We can then store in the BorderPane each of the other nodes we created
- Finally, we create the Scene and store within the scene the BorderPane we just created
- We set the dimensions of the scene to 400 x 600
- We then set the scene on the primaryStage
- We set the title to the primary stage to “Employee Directory”
- Next, we show the stage

```
        //Put all of the content inside of the Border
        pane.setTop(profilePic);
        pane.setCenter(name);
        pane.setBottom(description);

        Scene scene = new Scene(pane, 400, 600);
        //add the scene to the stage
        primaryStage.setScene(scene);
        //set the stage title
        primaryStage.setTitle("Employee Directory");
        //show the stage
        primaryStage.show();

    }

}
```

Test your software

Take a moment to test your software

Next Week

JavaFX Continued

Form Elements