

Java Programming 1

PROFESSOR: CÂI FILIAULT

Topics for this week

Objects

ArrayLists

Building GUI's using JavaFX

Form Elements

3 Pizza Inc

NEW EMAIL

NEW CLIENT

Email

Mike

Company owner

Hello,

We own a business called 3 pizza and would like to have a form that can be used to order pizza.

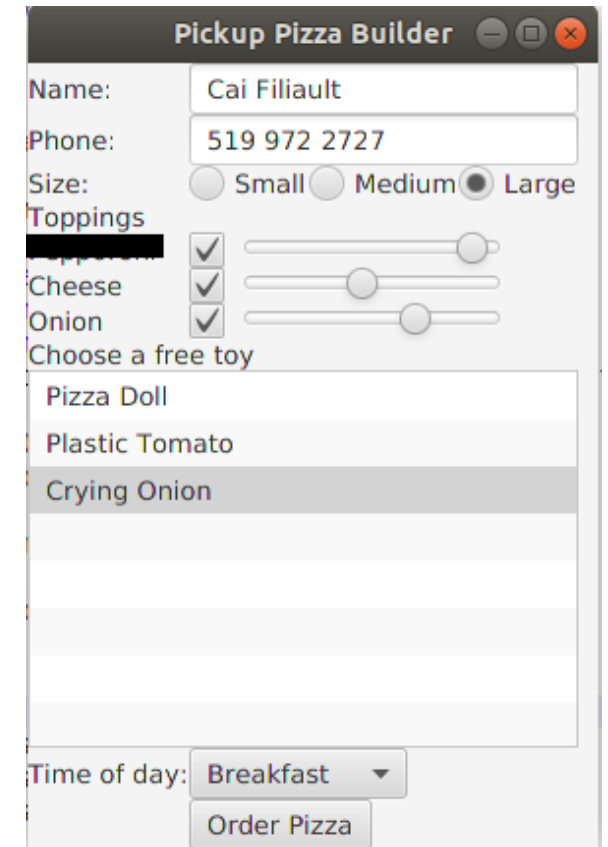
You will need to know the following:

- 3 sizes to a pizza (Small, Medium, Large)
- 3 possible ingredients (pepperoni, cheese, onion)
- 3 times of day you can order the pizza (breakfast, lunch, dinner)
- 3 possible free toys with every meal (Pizza Doll, Plastic Tomato, Crying Onion)
- The form should also receive the users name and phone number

-Thanks

Key Information

They are looking for us to build a GUI
We have made the following mockup image



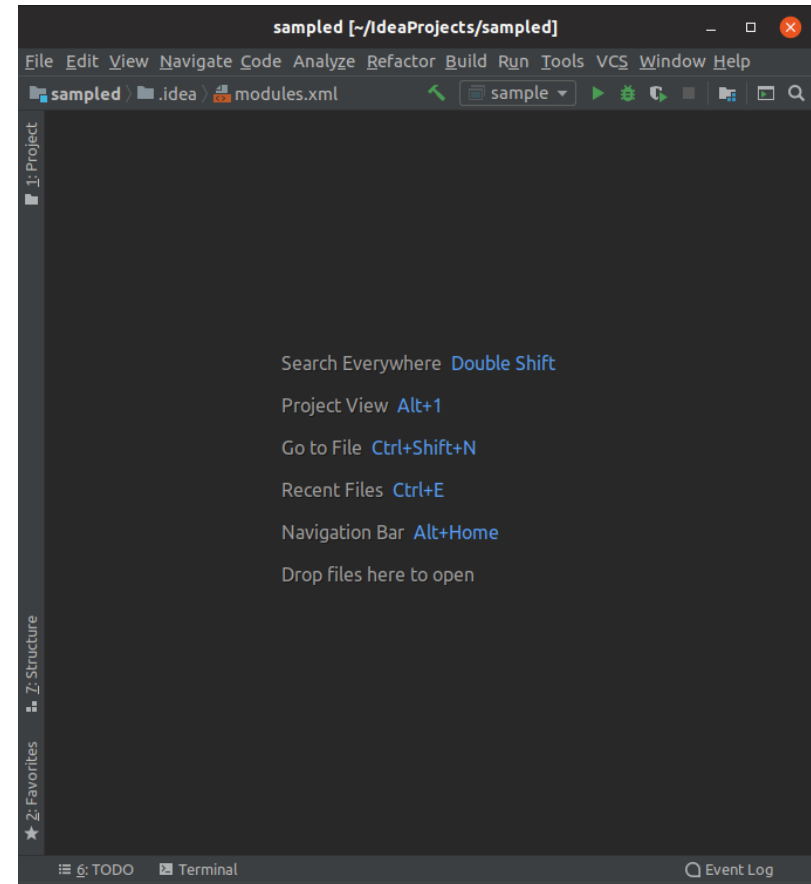
The mockup shows a window titled "Pickup Pizza Builder" with standard OS window controls. It contains the following elements:

- Name:** A text input field containing "Cai Filiault".
- Phone:** A text input field containing "519 972 2727".
- Size:** Three radio buttons labeled "Small", "Medium", and "Large". The "Large" button is selected.
- Toppings:** A section with a blacked-out header. It contains three items, each with a checked checkbox and a slider control:
 - Item 1: Checked checkbox, slider at approximately 80%.
 - Cheese: Checked checkbox, slider at approximately 50%.
 - Onion: Checked checkbox, slider at approximately 30%.
- Choose a free toy:** A list box containing four items: "Pizza Doll", "Plastic Tomato", "Crying Onion", and an empty space. "Crying Onion" is currently selected and highlighted.
- Time of day:** A dropdown menu currently showing "Breakfast".
- Order Pizza:** A button located at the bottom right of the window.

Creating a new project

Start by opening IntelliJ and selecting the following:

File> New> Project



New Project

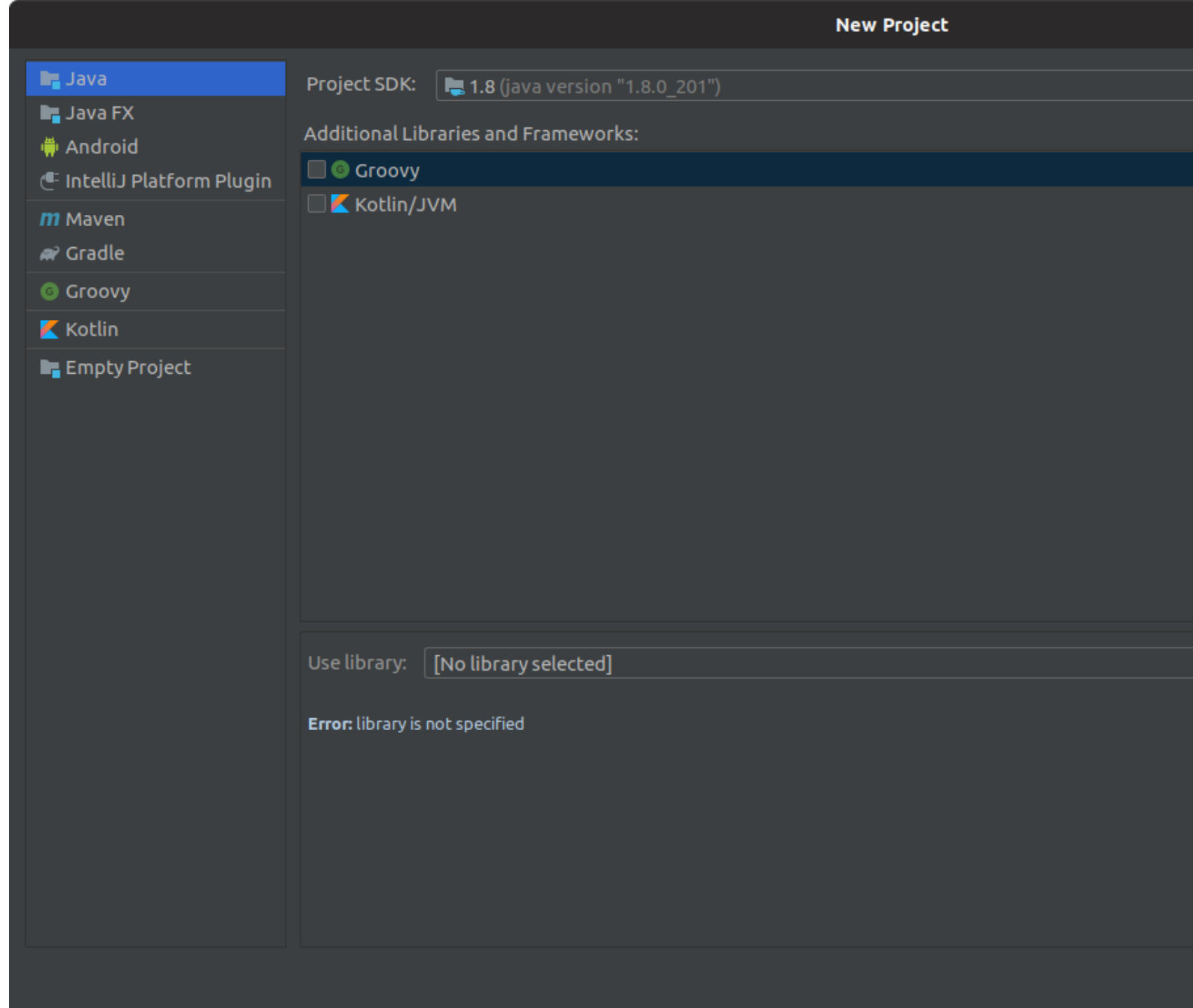
You will now see a dialog box appear *(It may be a little different than the one that I have shown)*

Starting on the left we will select a new Java Project

We will now want to select the project SDK of 1.8

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

At this point we can hit the next button twice



New Project

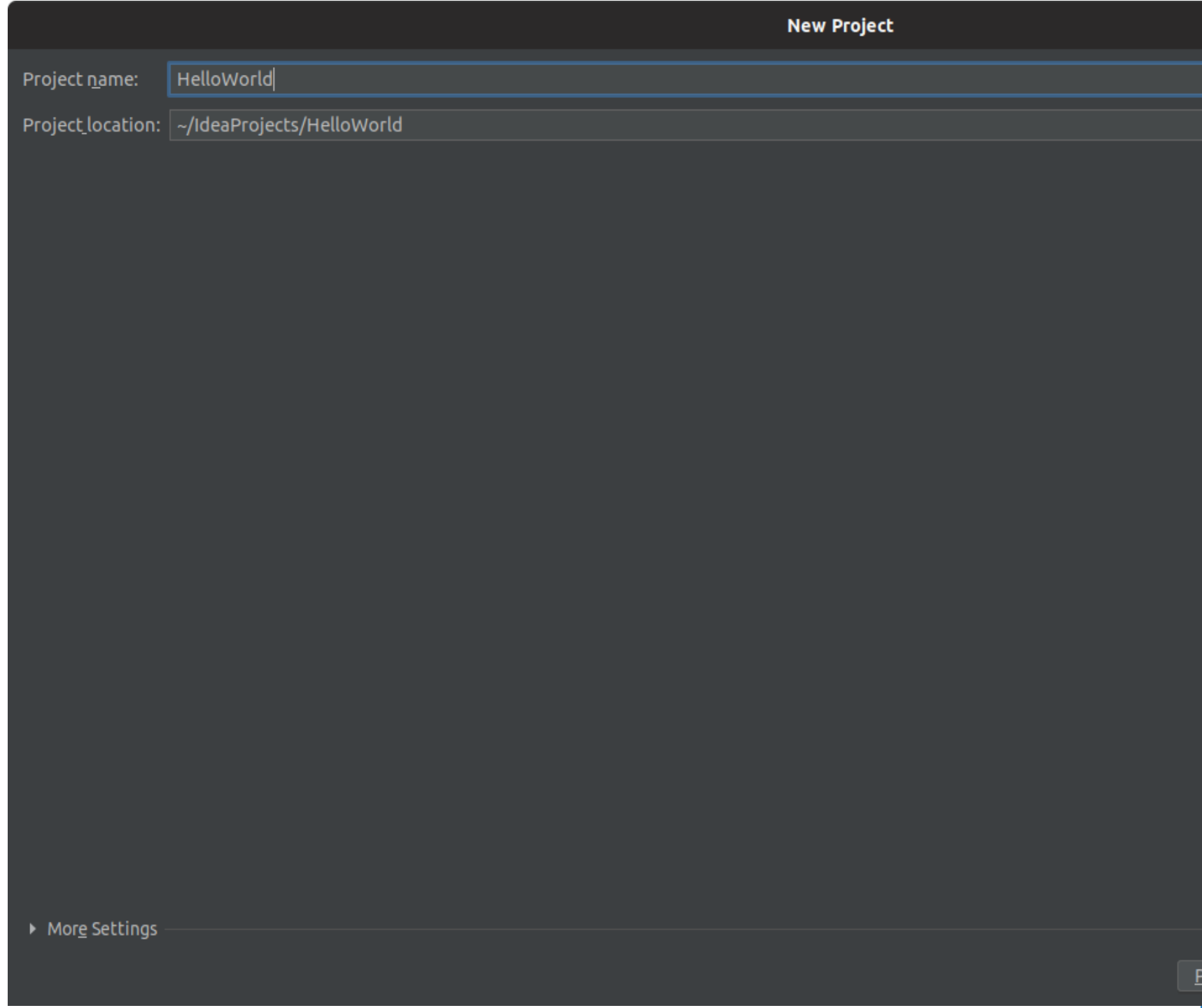
Give the project a name of:

ThreePizzaForm

At this time you can change the project location to any convenient place you would like

Tip:

Keeping your programming projects on a flash/jump drive is perfectly fine. However you will want to avoid from directly working on the drive.

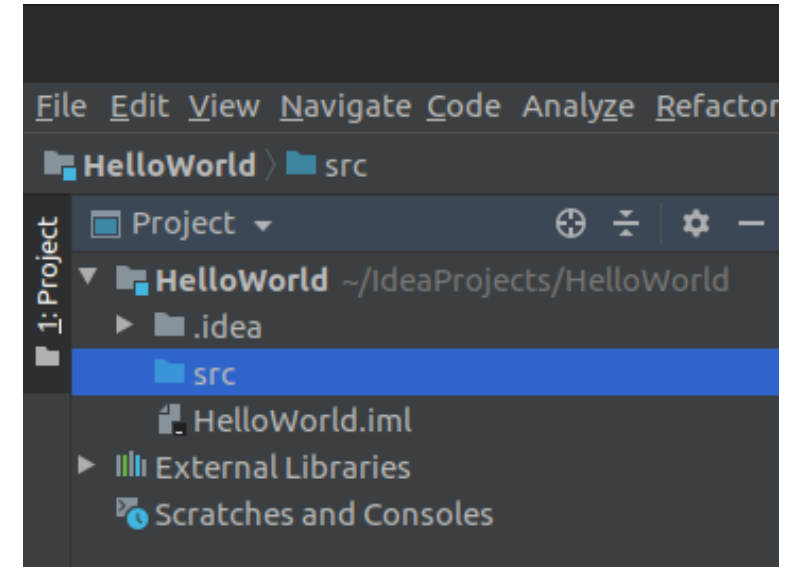


Creating a new project

You will now see in the top left-hand corner of your IDE a package explorer.

Within the package explorer you should see the ThreePizzaForm project.

Expand the ThreePizzaForm project to see a folder named **src**.



Creating a new class file

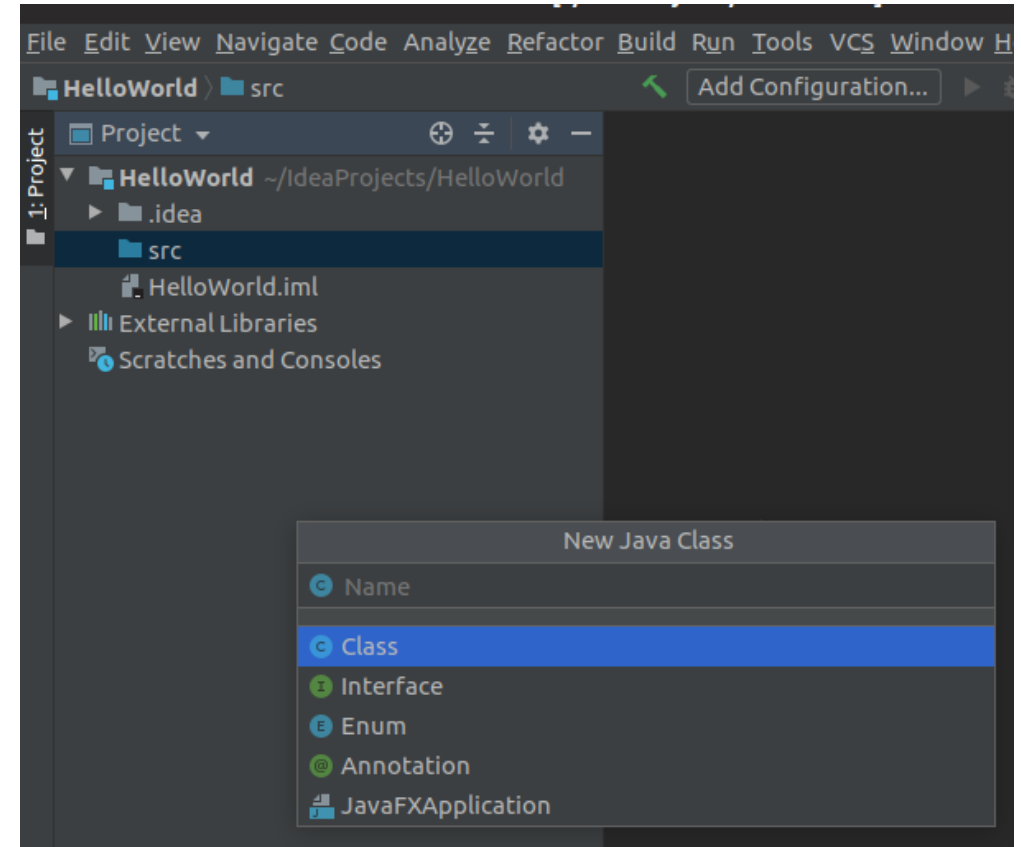
Next, we will need to create a new class file.

Right click on the **src** folder and select:

New> Java Class

Creating a new class is like creating a new program

Name the class **ThreePizzaForm** and hit **enter**



Writing the program

- Whenever we create an application 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 the class name 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

```
public class ThreePizzaForm extends Application {  
  
    public static void main(String[] args) {  
        Application.launch(args);  
    }  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        //Build a GridPane to display the form  
  
        //Build each row  
  
        //Create the Scene and add the GridPane  
        //Add the Scene to the Stage  
        //Set the title of the stage  
        //Show the stage  
    }  
}
```

Writing the program

- Let's start by creating a GridPane
- The GridPane works like a table, it has columns and rows.
- The columns and rows start at the index position of 0
- Everywhere in the grid can be addressed as a coordinate
- 0,0 would represent the column of the first row

```
@Override
public void start(Stage primaryStage) throws Exception {
    //Build a GridPane to display the form
    GridPane pane = new GridPane();

    //Build each row

    //Build the first row
    Label name = new Label("Name:");
    TextField nameTextField = new TextField();
    nameTextField.setPromptText("John Doe");

    pane.add(name, 0, 0);
    pane.add(nameTextField, 1, 0);

    //Build the second row
```

Writing the program

- We then build the first row which consists of a label
- A label is like a Text object with a few differences when it comes to CSS styling. We will go into more detail on this in a later course.
- For now we will know that when labeling form fields we should use the label class
- We can see that we add the name label to 0,0 and the text field to 1, 0
- This order is column, row
- We can see that the name and label are on the same row but different columns

```
@Override
public void start(Stage primaryStage) throws Exception {
    //Build a GridPane to display the form
    GridPane pane = new GridPane();

    //Build each row

    //Build the first row
    Label name = new Label("Name:");
    TextField nameTextField = new TextField();
    nameTextField.setPromptText("John Doe");

    pane.add(name, 0, 0);
    pane.add(nameTextField, 1, 0);

    //Build the second row
```

Writing the program

- The next step that I like to complete is building the scene and the stage.
- If we complete this step now, we will be able to view our progress as we make our way through the application.

```
Scene scene = new Scene(pane, 290, 400);
primaryStage.setTitle("Pickup Pizza Builder");
primaryStage.setScene(scene);
primaryStage.show();
```

Writing the program

- Next, we work on the third row
- The third row will consist of 3 radio buttons and a label
- We build the radio buttons for small, medium and large.
- We now need to know that when we click one of these values that another will deselect.
- To add this functionality each RadioButton must share a ToggleGroup
- We build a ToggleGroup named pizzaSize and assign the three radio buttons to it

```
//Build the third row
Label size = new Label("Size:");
ToggleGroup pizzaSize = new ToggleGroup();
RadioButton small = new RadioButton("Small");
small.setToggleGroup(pizzaSize);
RadioButton medium = new RadioButton("Medium");
medium.setToggleGroup(pizzaSize);
RadioButton large = new RadioButton("Large");
large.setToggleGroup(pizzaSize);

HBox pizzaHBox = new HBox();
pizzaHBox.getChildren().addAll(small, medium, large);

pane.add(size, 0, 2);
pane.add(pizzaHBox, 1, 2);

//Build the fourth row
```

Writing the program

- We then create an HBox to store the three buttons
- We want to ensure that the whole group of buttons occurs in the second column.
- We then add the label and the HBox to the GridPane

```
//Build the third row
Label size = new Label("Size:");
ToggleGroup pizzaSize = new ToggleGroup();
RadioButton small = new RadioButton("Small");
small.setToggleGroup(pizzaSize);
RadioButton medium = new RadioButton("Medium");
medium.setToggleGroup(pizzaSize);
RadioButton large = new RadioButton("Large");
large.setToggleGroup(pizzaSize);

HBox pizzaHBox = new HBox();
pizzaHBox.getChildren().addAll(small, medium, large);

pane.add(size, 0, 2);
pane.add(pizzaHBox, 1, 2);

//Build the fourth row
```


Writing the program

- Next, we begin to build the toppings.
- Recall that we have 3 toppings that the pizza may contain (pepperoni, onion, cheese)
- Considering the user may want any variations of these toppings they are best represented as a checkbox.
- We are also going to add a slider so that the user can specify how much of the ingredient they want on a scale from 1 – 10
- Just like the previous row we add the items into an HBox so that we do not exceed two columns
- We then add all the elements to the pane in the fifth row.

```
//Build the fourth row
Label toppings = new Label("Toppings");

pane.add(toppings, 0, 3);

//Build the fifth row
Label topping1 = new Label( );
CheckBox pepCheckBox = new CheckBox();
pane.add(topping1, 0, 4);
Slider pepSlider = new Slider();
pepSlider.setMax(10);
pepSlider.setMin(1);
HBox topping1HBox = new HBox();
topping1HBox.getChildren().addAll(pepCheckBox, pepSlider);
pane.add(topping1HBox, 1, 4);

//Build the sixth row
```

Writing the program

- Repeat the process for cheese

```
//Build the sixth row
Label topping2 = new Label("Cheese");
CheckBox cheeseCheckBox = new CheckBox();
pane.add(topping2, 0, 5);
Slider cheeseSlider = new Slider();
cheeseSlider.setMax(10);
cheeseSlider.setMin(1);
HBox topping2HBox = new HBox();
topping2HBox.getChildren().addAll(cheeseCheckBox, cheeseSlider);
pane.add(topping2HBox, 1, 5);
```

Writing the program

- Repeat again for onion

```
//Build the seventh row
Label topping3 = new Label("Onion");
CheckBox onionCheckBox = new CheckBox();
pane.add(topping3, 0, 6);
Slider onionSlider = new Slider();
onionSlider.setMax(10);
onionSlider.setMin(1);
HBox topping3HBox = new HBox();
topping3HBox.getChildren().addAll(onionCheckBox, onionSlider);
pane.add(topping3HBox, 1, 6);

//Build the eighth row
```

Writing the program

- Next, we need to represent a list of possible toys they can choose from
- Displaying a list is best done with a ListView
- The ListView takes in an ObservableArrayList of items. (An ArrayList that is viewable to the GUI)
- There is a method that exists that can convert regular arraylists into observable ones and it is in the FXCollections class

```
//Build the eighth row
Label toyLabel = new Label("Choose a free toy");
pane.add(toyLabel, 0, 7, 2, 1);

//Build the ninth row
ArrayList<String> toys = new ArrayList<>();
toys.add("Pizza Doll");
toys.add("Plastic Tomato");
toys.add("Crying Onion");

ListView listView = new ListView();
listView.setItems(FXCollections.observableArrayList(toys));
pane.add(listView, 0, 8, 2, 1);

//Build tenth row
```

Writing the program

- Finally, we need to represent the time of day which has 3 possible values (Breakfast, Lunch, Dinner)
- This time we will do this using a ComboBox
- ComboBox work in the same way that ListViews do whereby they take in an ObservableArrayList of items
- Taking what we now know we can build a String list of items and add them to the ComboBox.
- We then add the ComboBox to the screen with the label.

```
//Build tenth row
Label timeOfDay = new Label("Time of day:");

ComboBox<String> comboBox = new ComboBox();
ArrayList<String> times = new ArrayList<>();
times.add("Breakfast");
times.add("Lunch");
times.add("Dinner");
comboBox.setItems(FXCollections.observableArrayList(times));

pane.add(timeOfDay, 0, 9);
pane.add(comboBox, 1, 9);

//Build eleventh row
Button button = new Button("Order Pizza");
pane.add(button, 1, 10);
```

Writing the program

The last step is to add the order button to the form.

```
//Build tenth row
Label timeOfDay = new Label("Time of day:");

ComboBox<String> comboBox = new ComboBox();
ArrayList<String> times = new ArrayList<>();
times.add("Breakfast");
times.add("Lunch");
times.add("Dinner");
comboBox.setItems(FXCollections.observableArrayList(times));

pane.add(timeOfDay, 0, 9);
pane.add(comboBox, 1, 9);

//Build eleventh row
Button button = new Button("Order Pizza");
pane.add(button, 1, 10);
```

Test your software

Take a moment to test your software

