COMP20050 - Software Engineering Project II

JavaFX Scene Builder
Hexagonal Boards

Ravi Reddy Manumachu
ravi.manumachu@ucd.ie

UCD School of Computer Science.

Scoil na Ríomheolaíochta
UCD.

Outline (Learning Objectives)

- Become familiar with the features in **Scene Builder**.
- Build a JavaFX application using **Scene Builder**.

User Guide and Tutorial

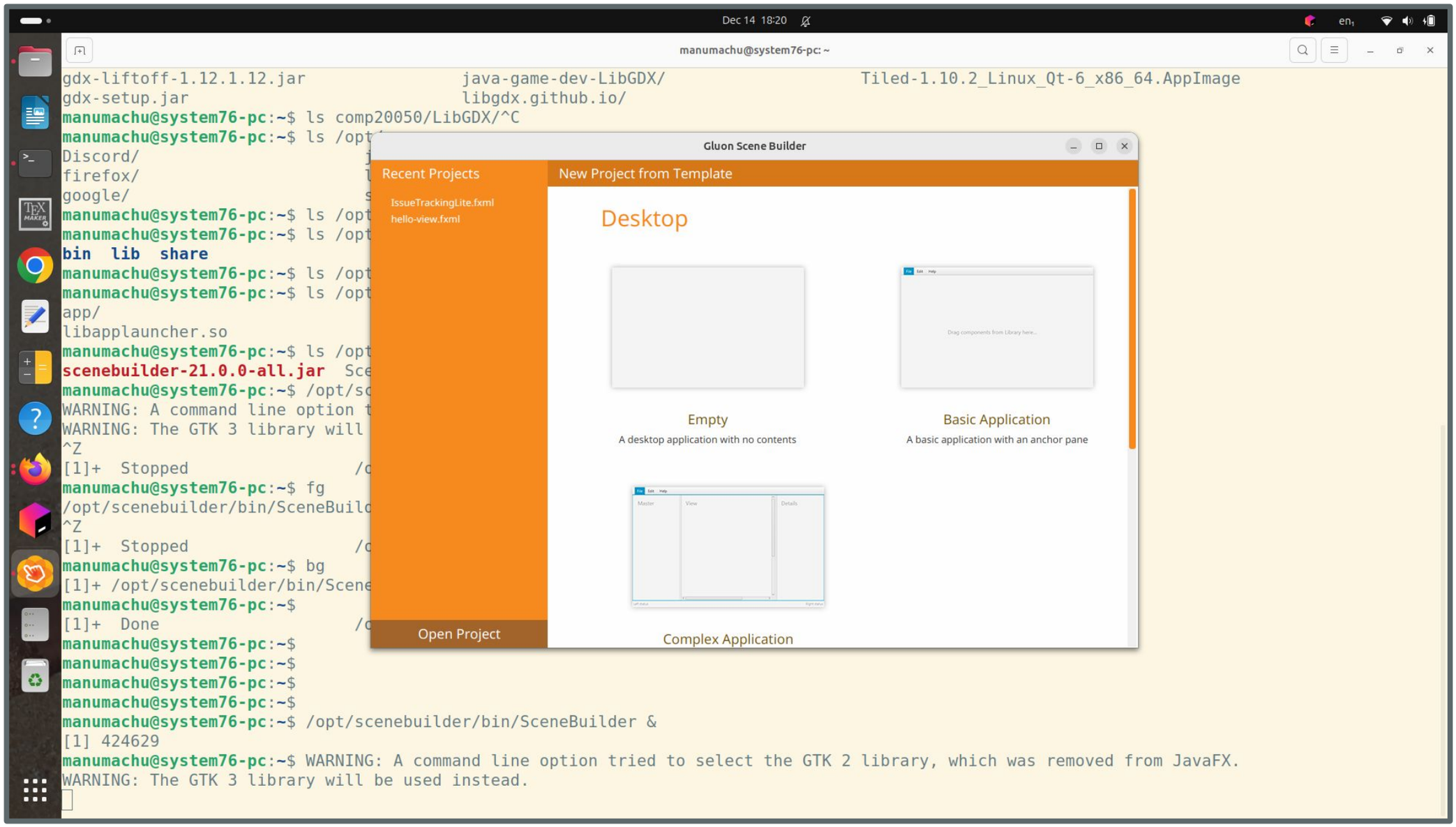
- User guide:
  - https://docs.oracle.com/javase/8/scene-builder-2/user-guide/index.html

- Tutorial:
  - https://docs.oracle.com/javase/8/scene-builder-2/get-started-tutorial/index.html

JavaFX IssueTracking Application Using Scene Builder

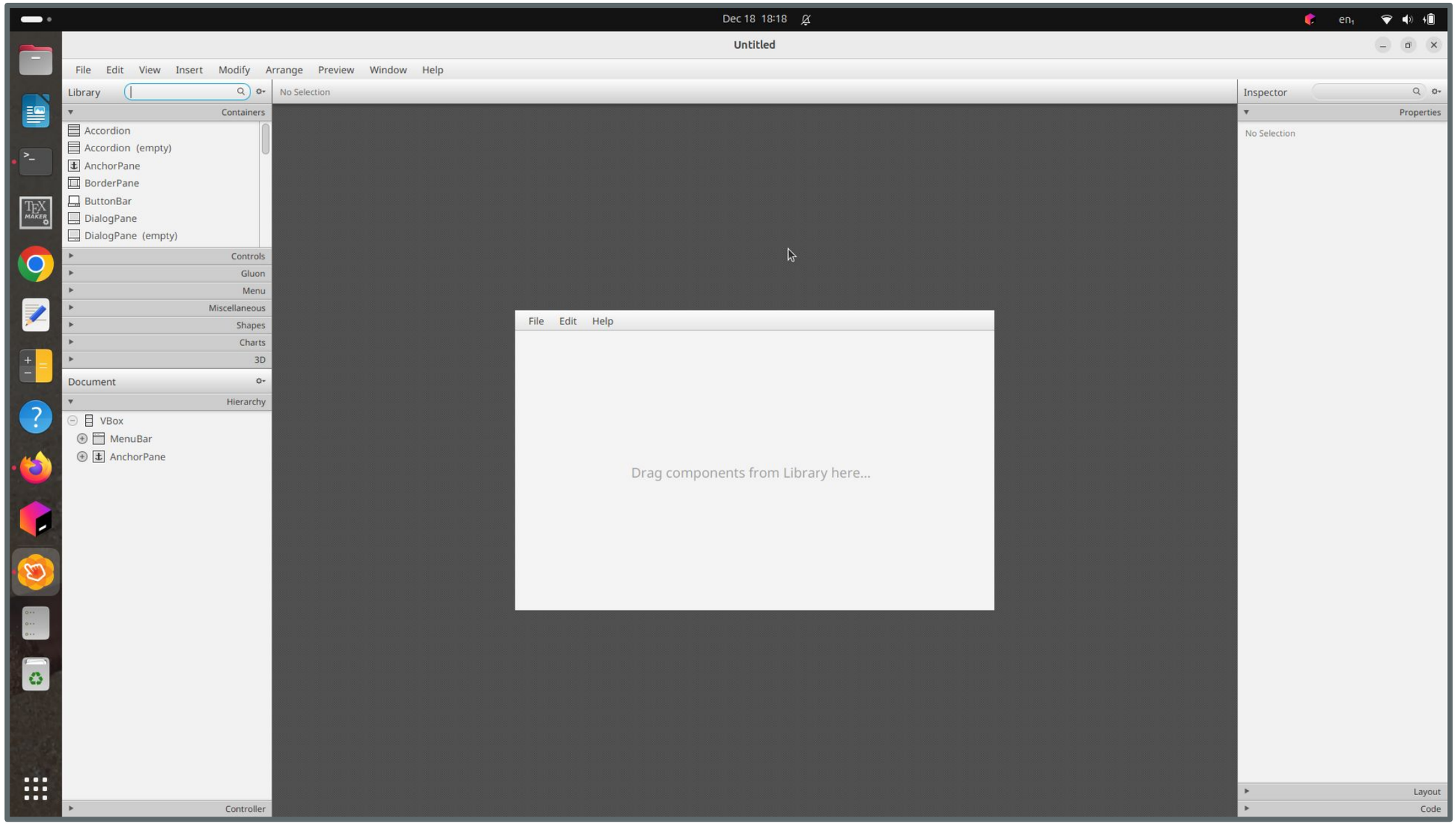
JavaFX Scene Builder Launch

- On my Ubuntu system I use the following command to launch Scene Builder.
  shell$ /opt/scenebuilder/bin/SceneBuilder &

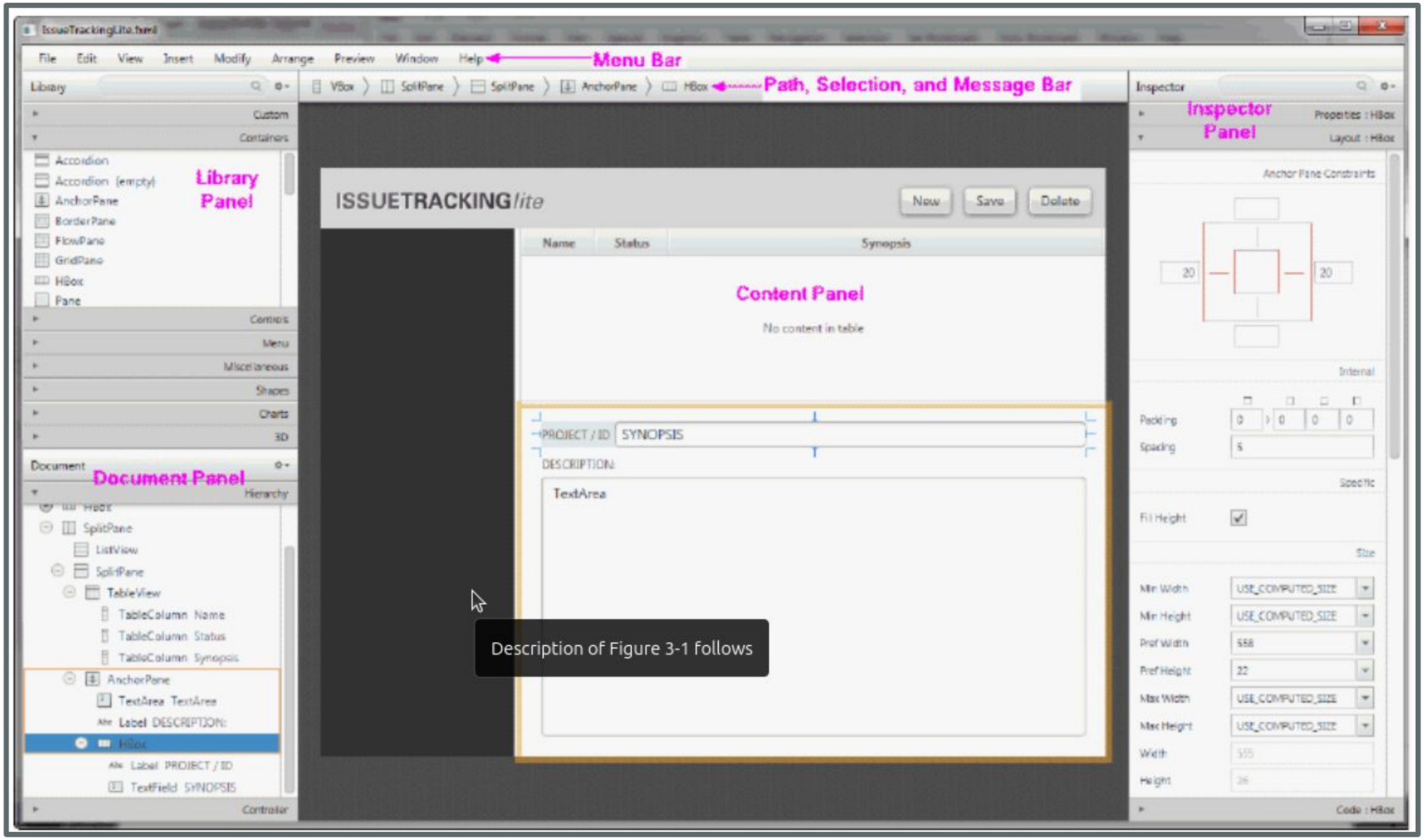


JavaFX Scene Builder Startup Window

- Click **Basic Application**.



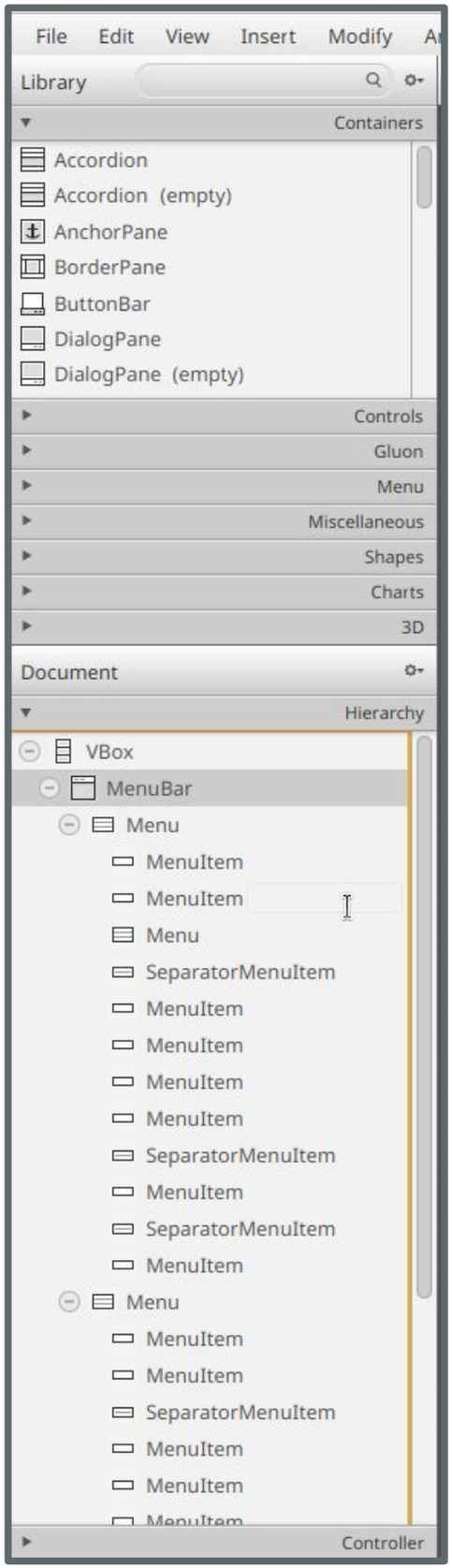
Anatomy of the Main Window



Library and Document Panels

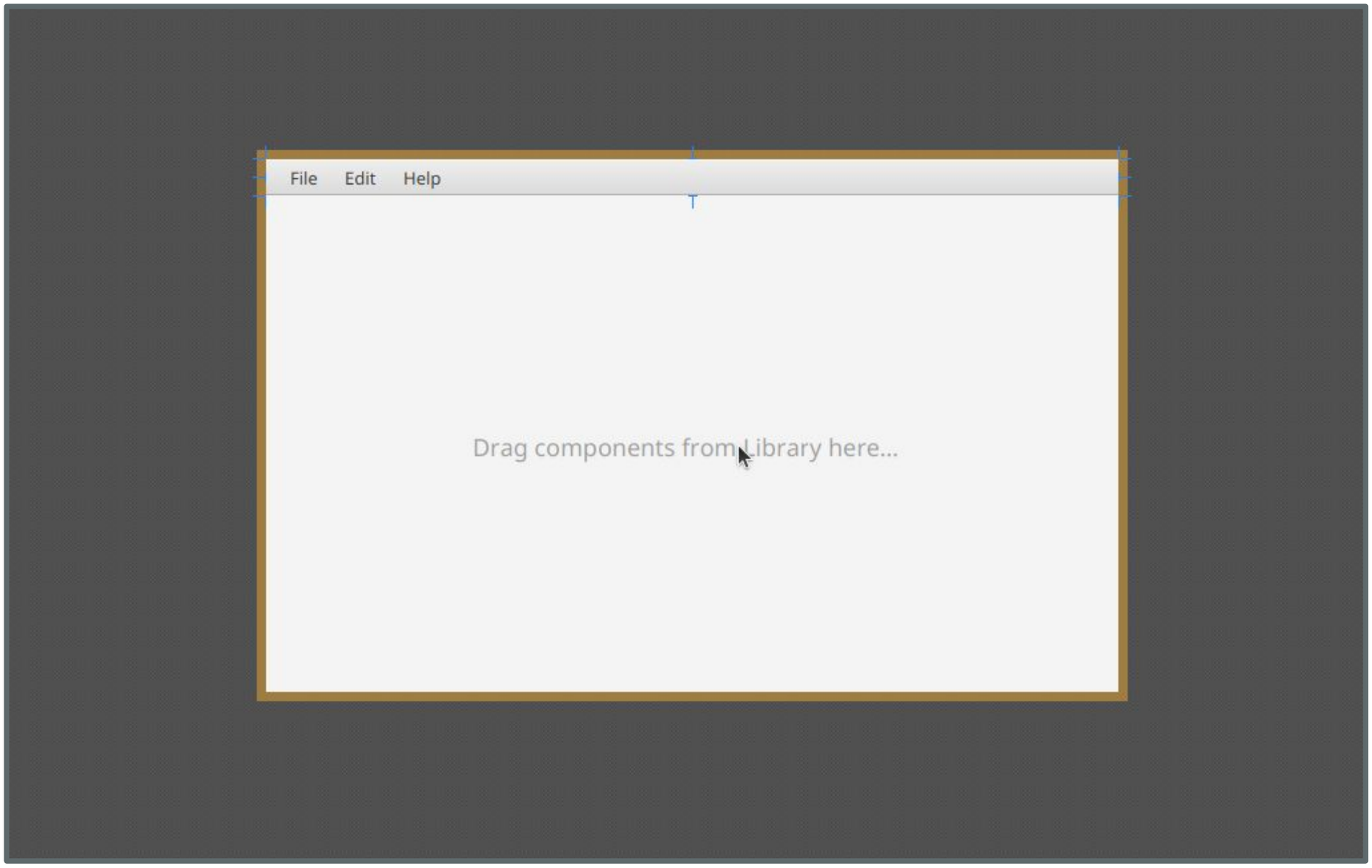
- The **Library Panel** lists the available JavaFX GUI elements or controls that you can use to build your FXML layout.

- The **Document Panel** contains the **Hierarchy section**, which displays a **tree view** representation of the **FXML layout** that you are building in the **Content panel**.



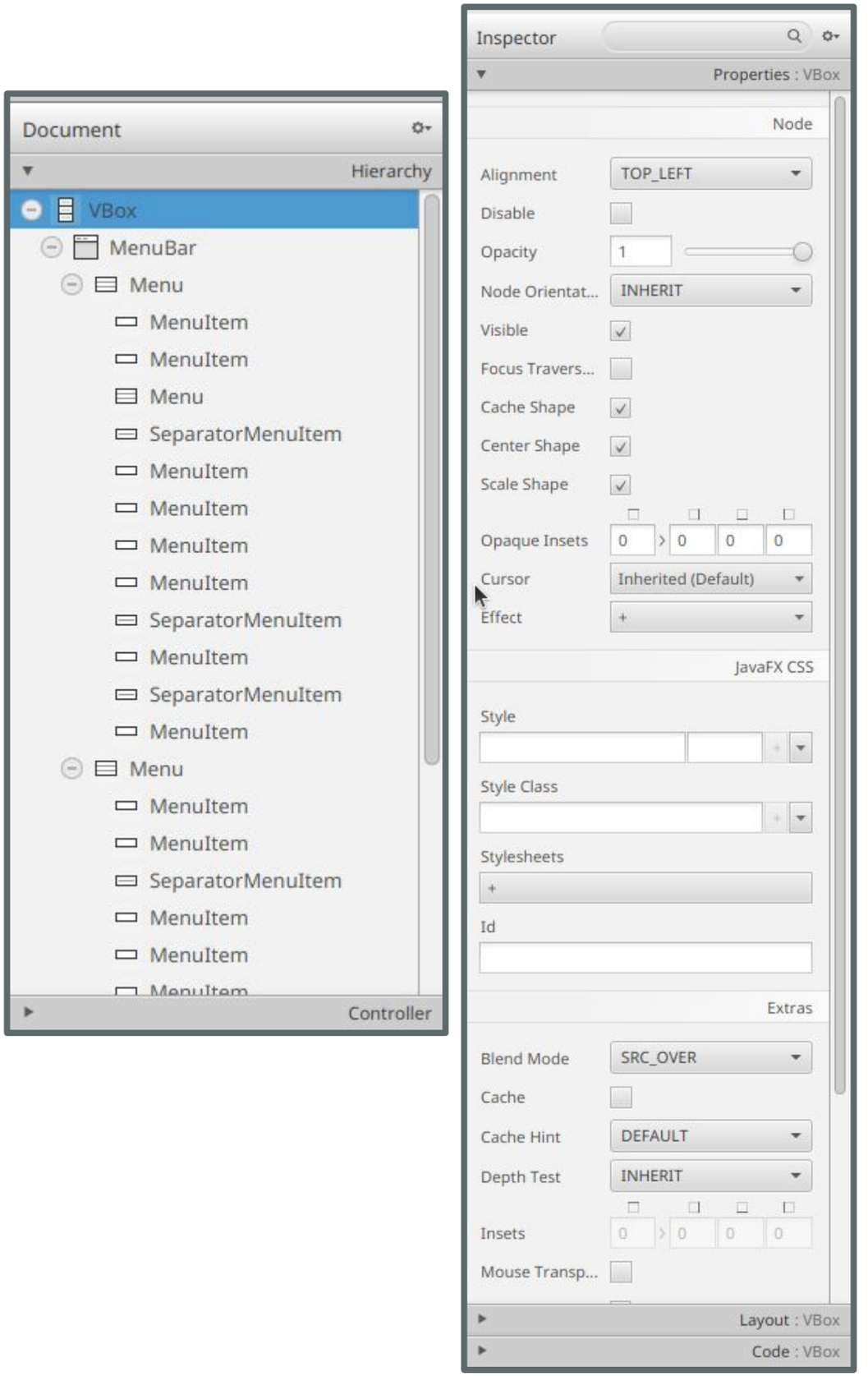
Content Panel

- The **Content Panel** is the scene container for the GUI elements that make up your FXML layout.



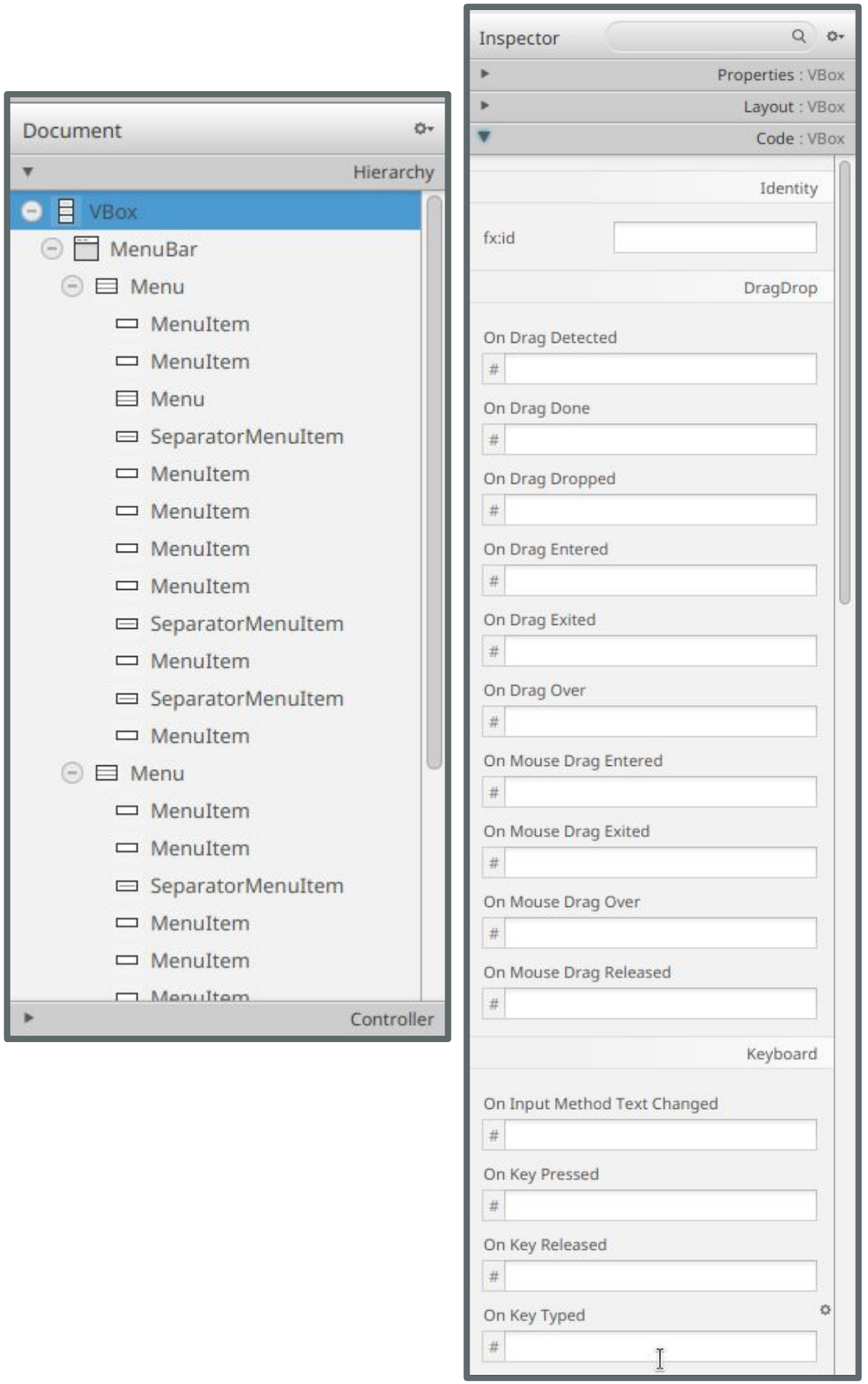
Inspector Panel

- The **Inspector Panel** contains the Properties, Layout, and Code sections.
- The **Properties and Layout sections** manage the properties of the currently selected GUI element.



Inspector Panel: Code Section

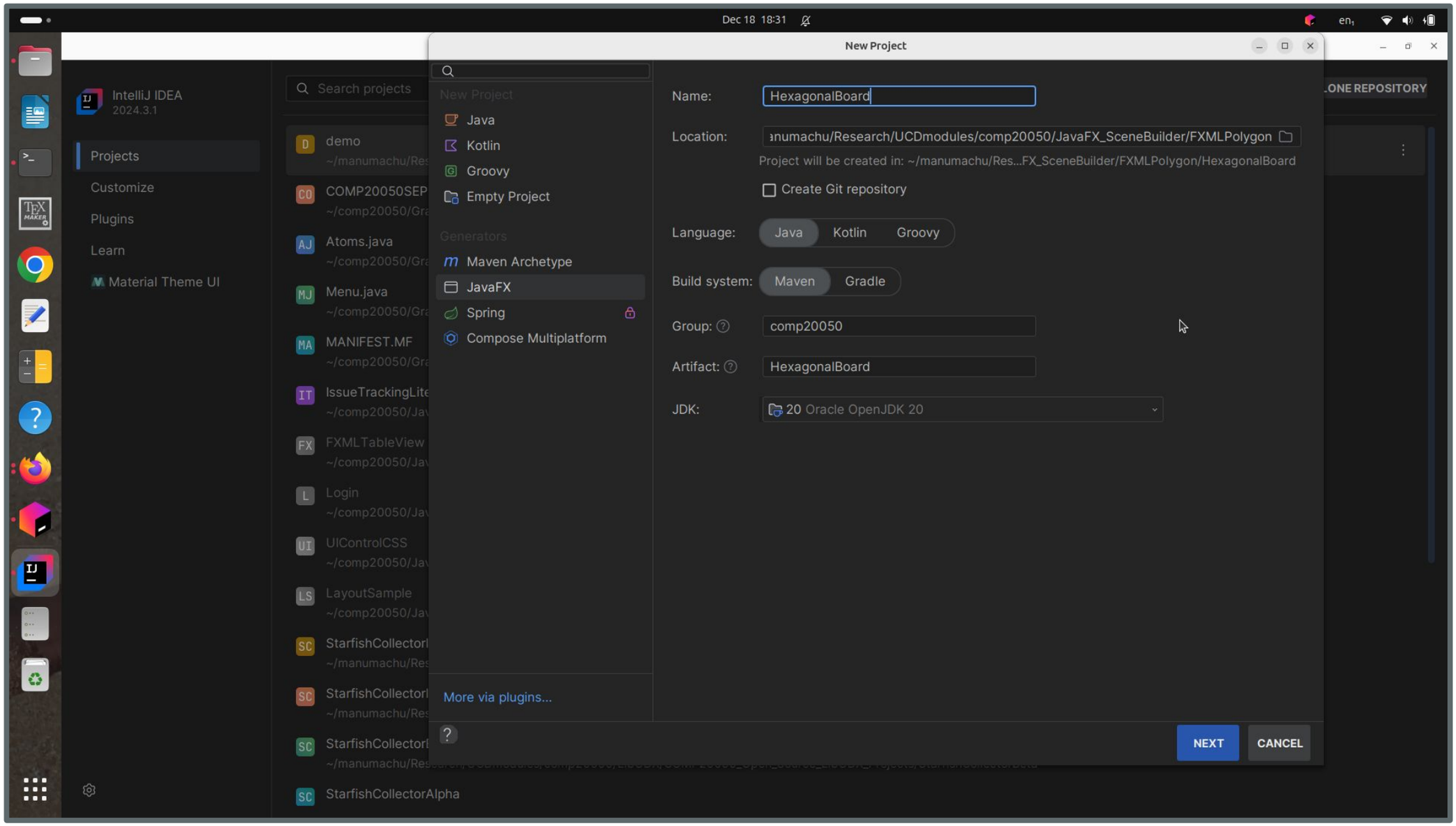
- The **Code section** allows to manage the event handling actions to use for the selected GUI element.



Hexagonal Boards Using Scene Builder

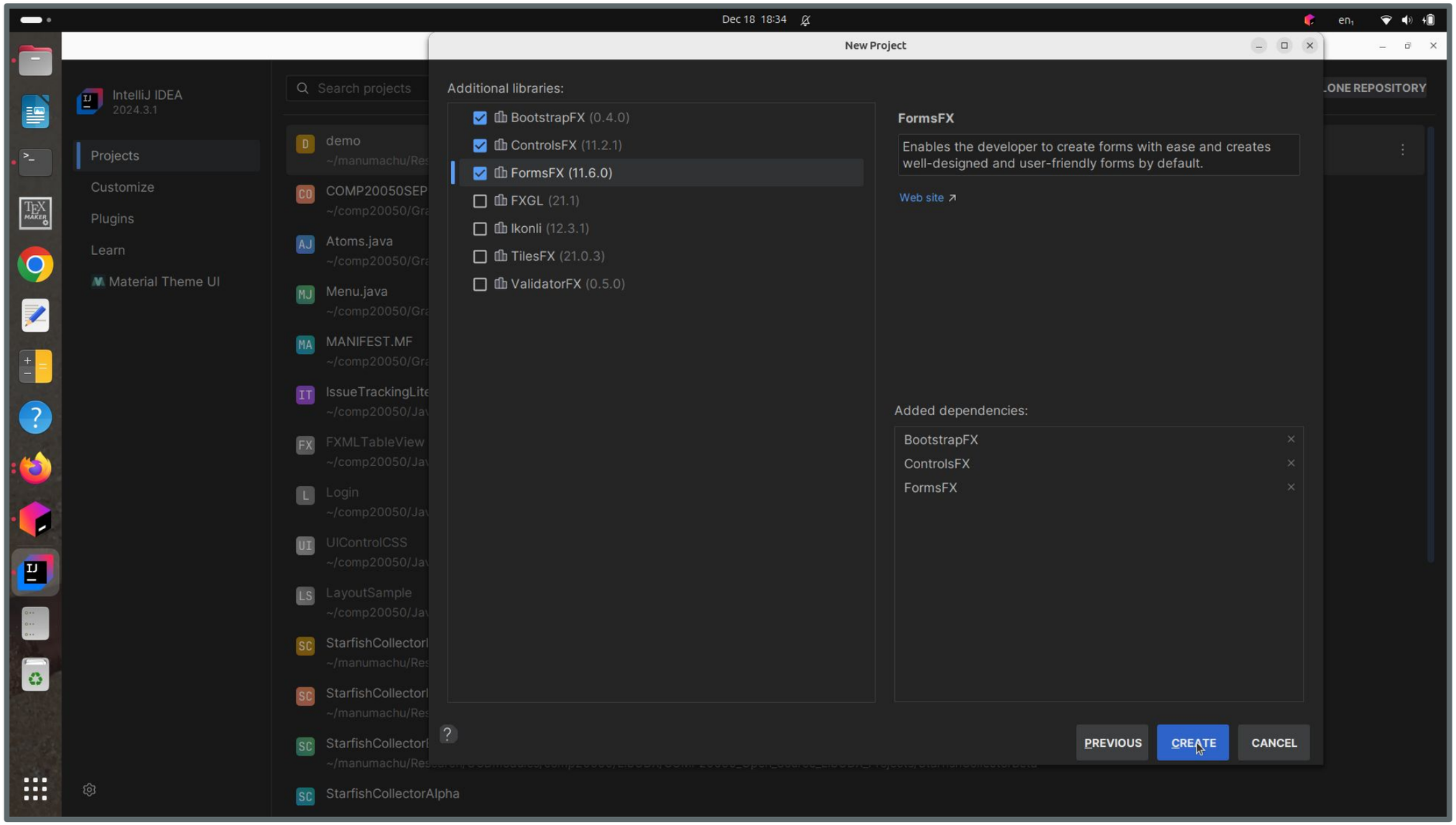
New JavaFX Project

- Create a new JavaFX project with the following attributes:
- **Name:** HexagonalBoard, **Group:** comp20050.



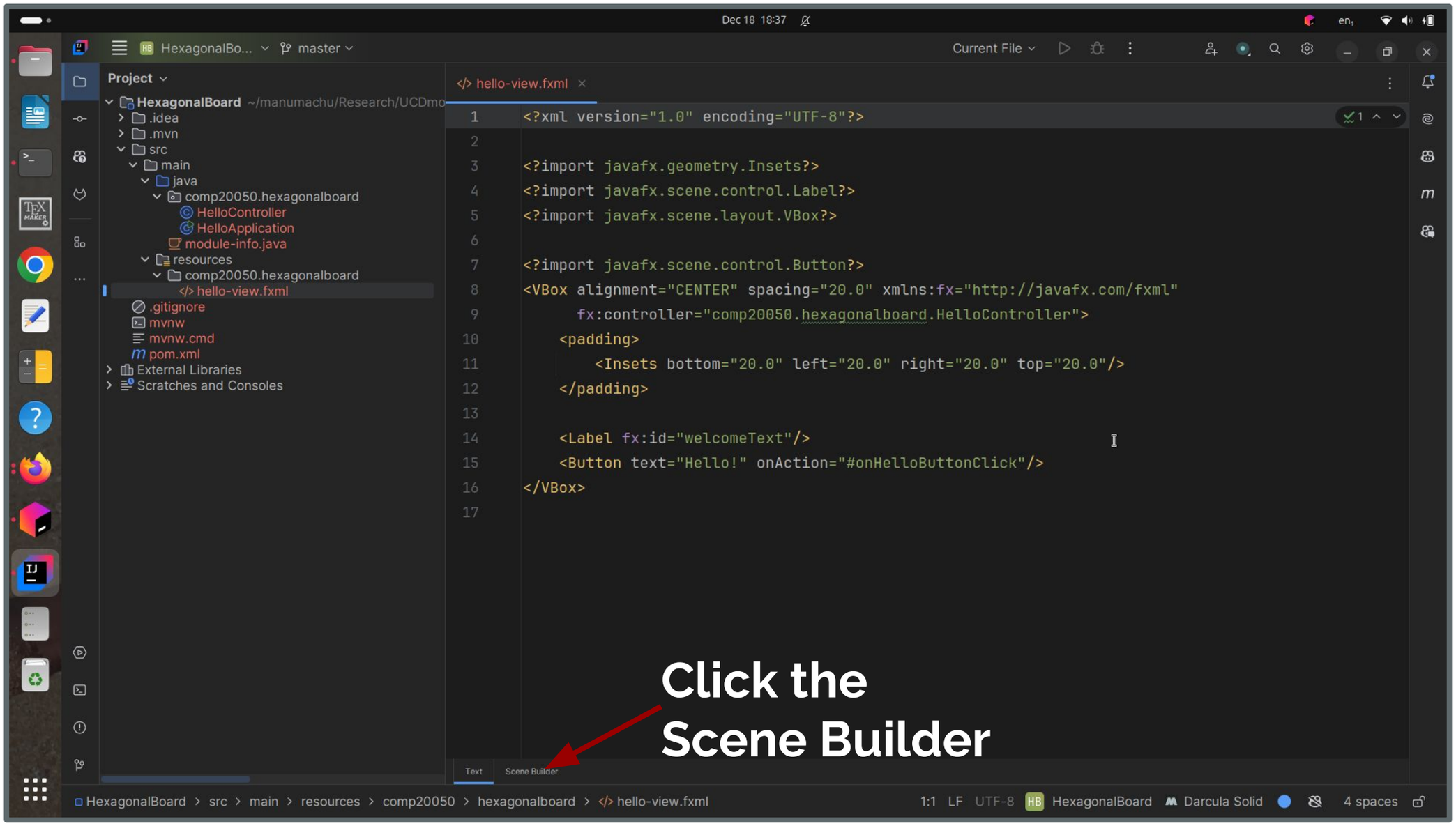
Additional Libraries

- Select any additional libraries.
- For this example, you will not be needing any of these.



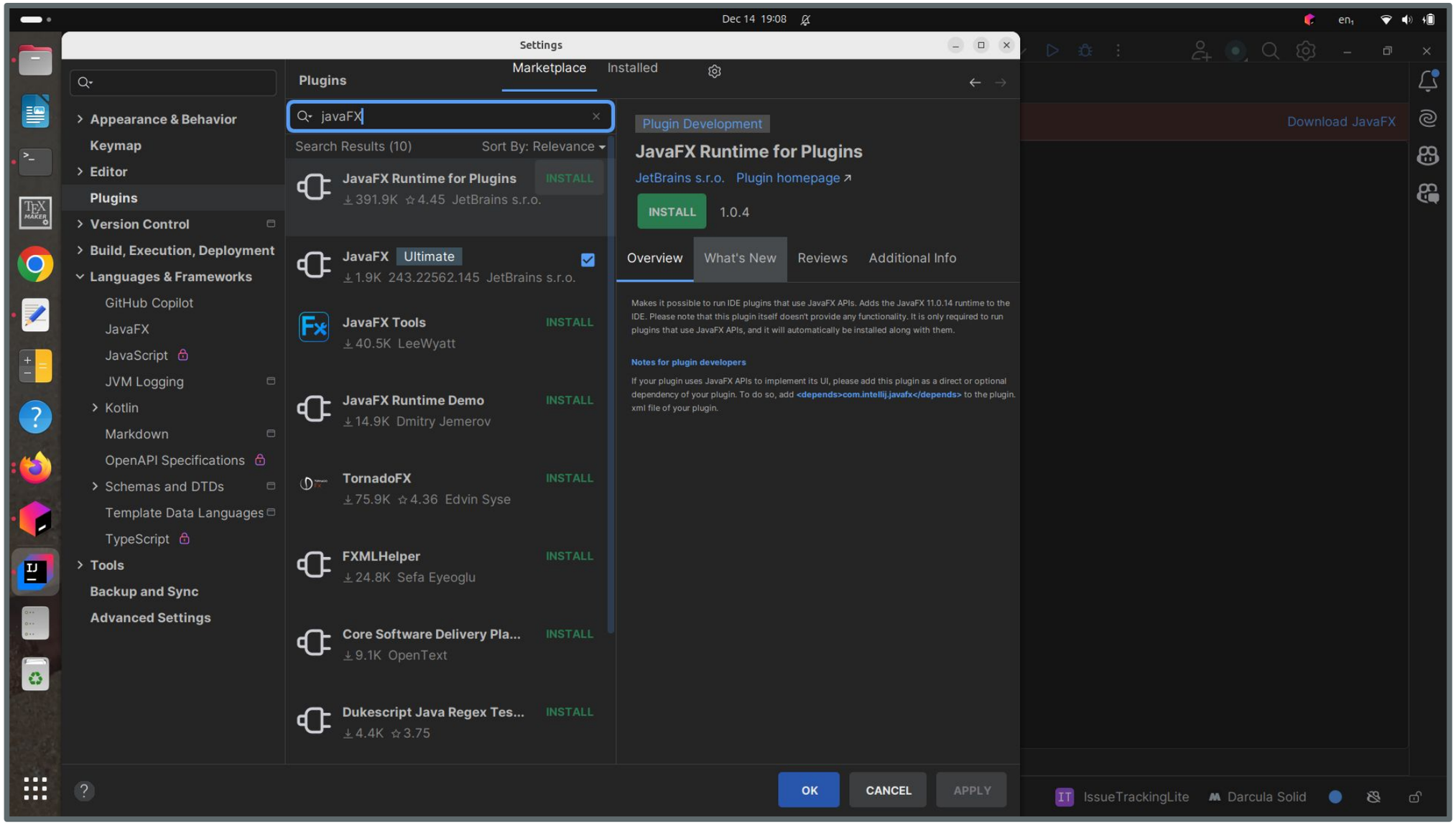
See FXML, Launch Scene Builder

- Click the FXML file under
  HexagonalBoard -> src -> main -> resources -> comp20050.*
- Click the **Scene Builder**.



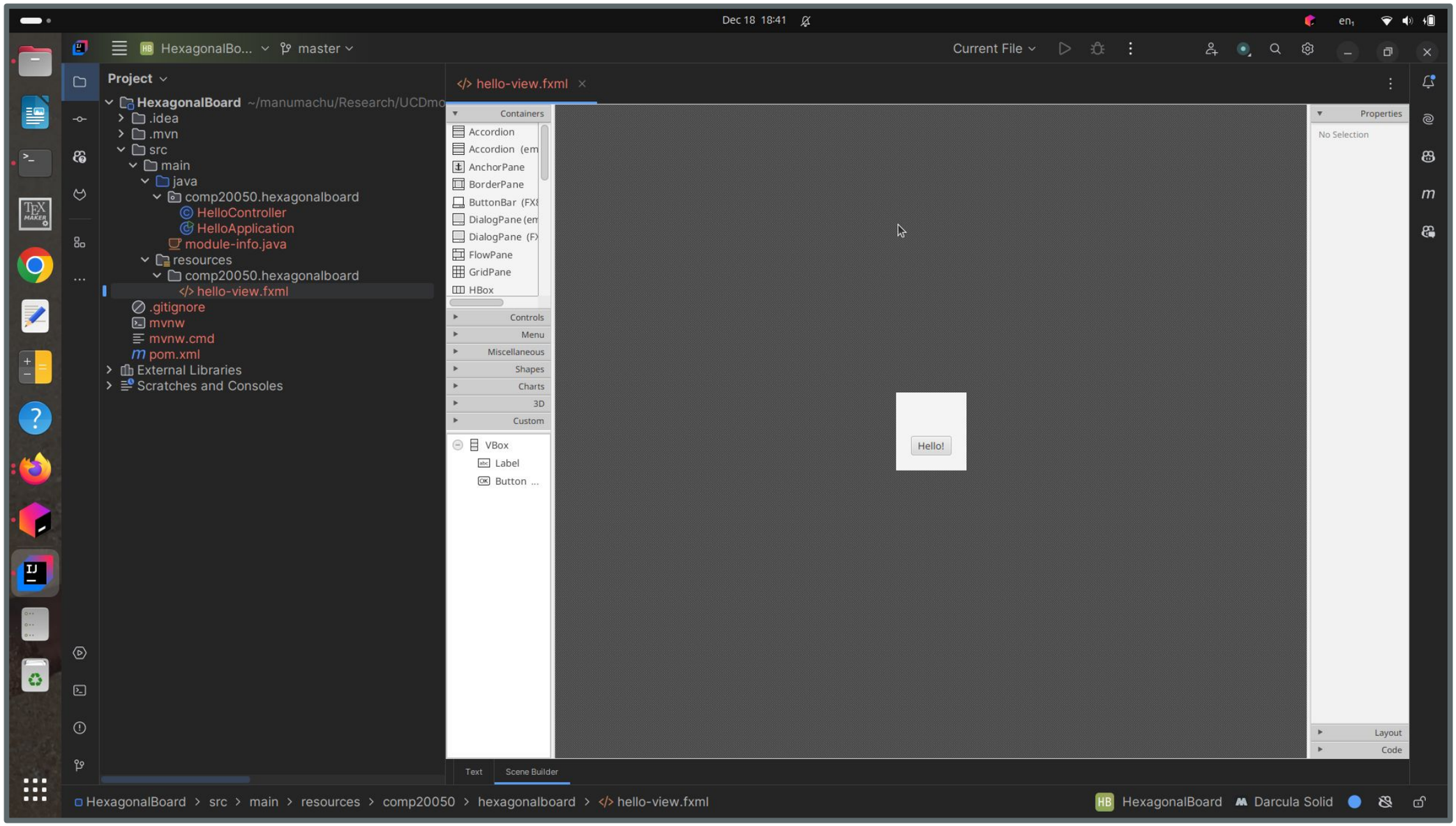
Launch Scene Builder: Install JavaFX Plugin

- Failed to open the file in the Scene Builder.
- Click **File** -> **Settings** -> **Plugins**.
- Install **JavaFX Runtime for Plugins** and **Restart IDE**.



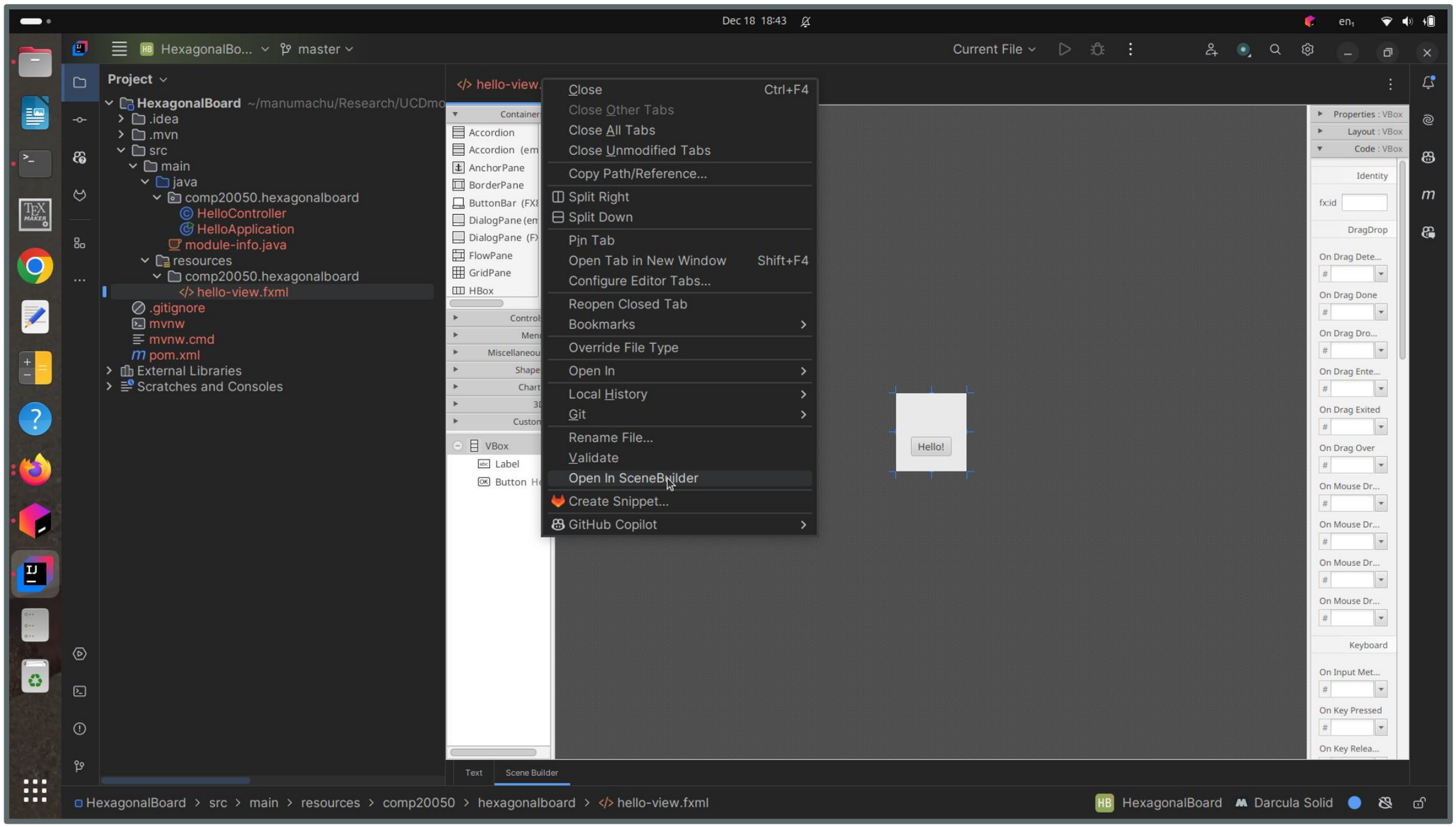
Launch Scene Builder

- The Scene Builder window is condensed. For example, the main menu is not visible.



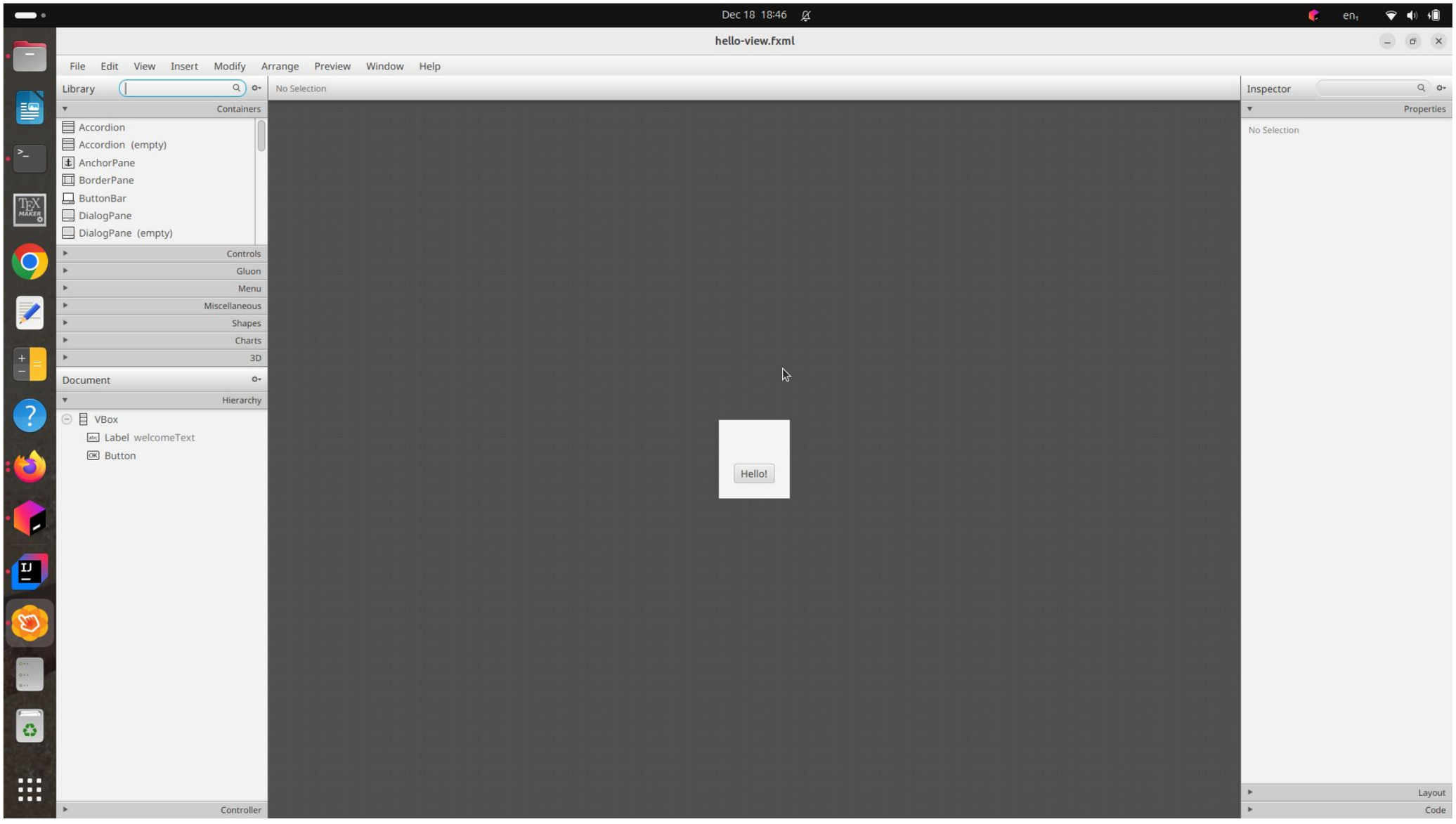
Full Scene Builder View

- To see the full Scene Builder, right-click **hello-view.fxml** tab using the mouse and click **Open In SceneBuilder**.



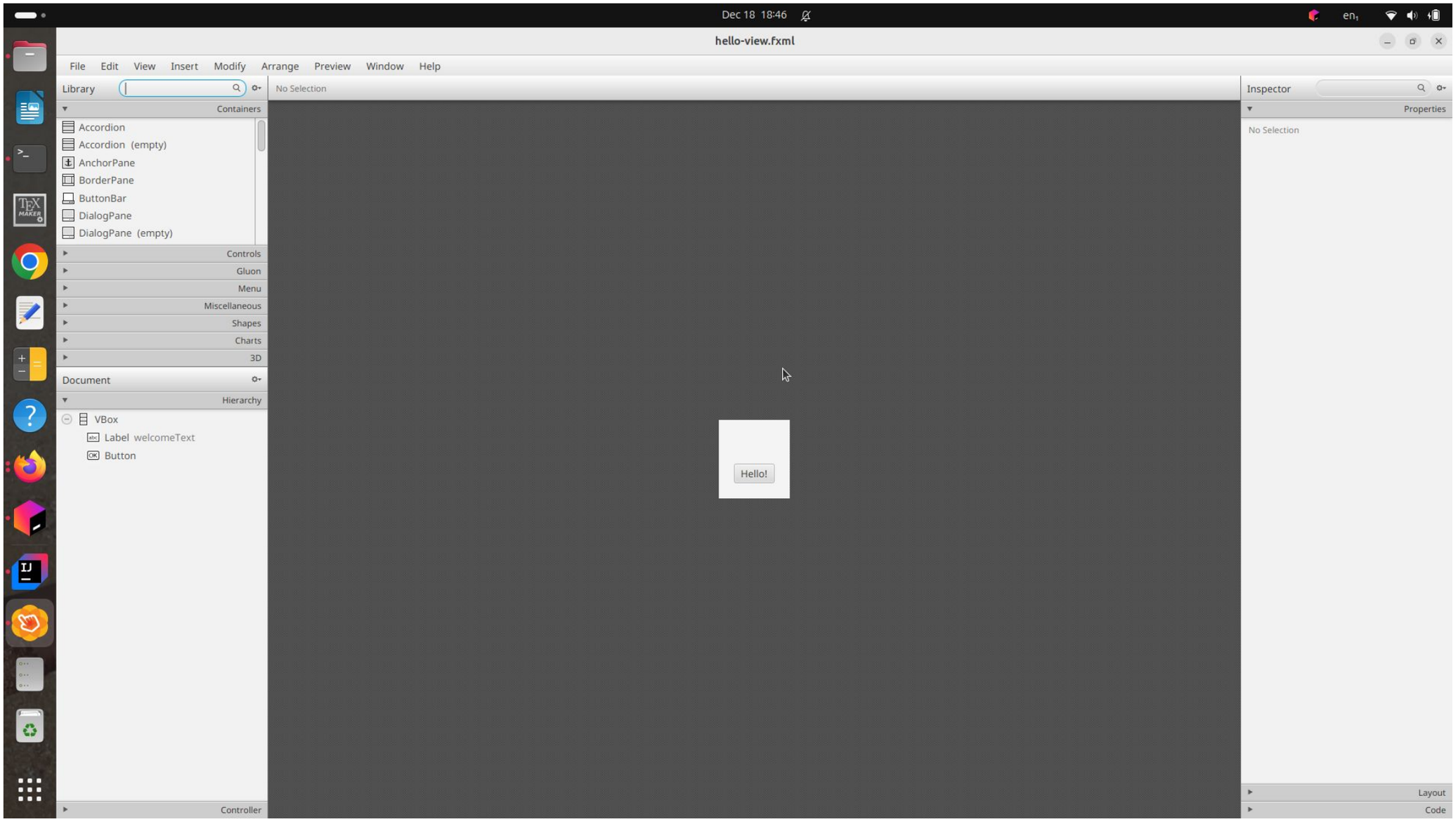
Full Scene Builder View

- I would recommend you design the GUI layout in this window.
- The changes will be reflected in the IntelliJ editor. Delete VBox and add an AnchorPane.



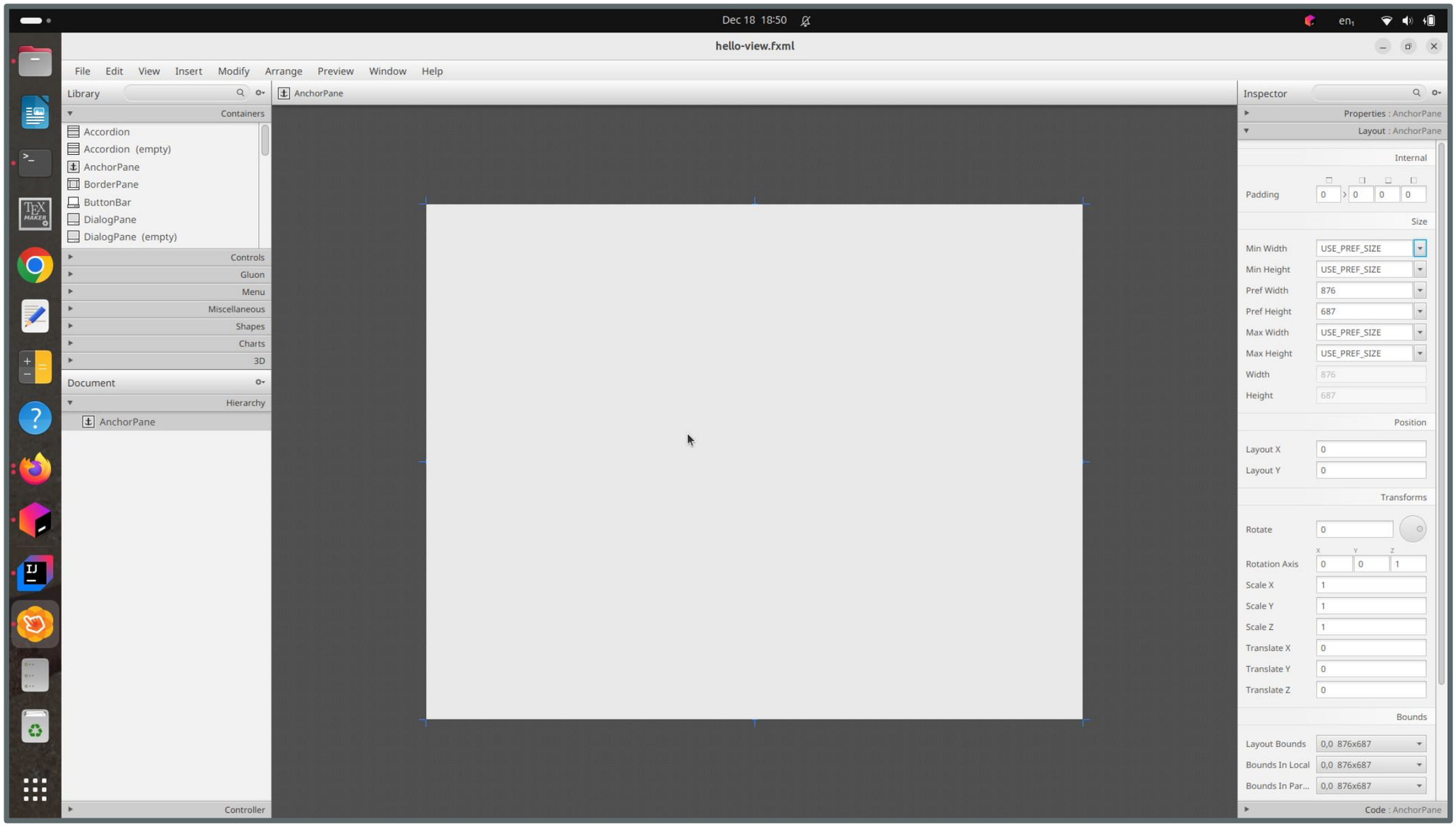
Full Scene Builder View

- I would recommend you design the GUI layout in this window.
- The changes will be reflected in the IntelliJ editor. Delete VBox.



Add AnchorPane

- Drag and drop the **AnchorPane** from the Library panel into the Content Panel.



Shape Class Hierarchy

Hierarchy For Package javafx.scene.shape

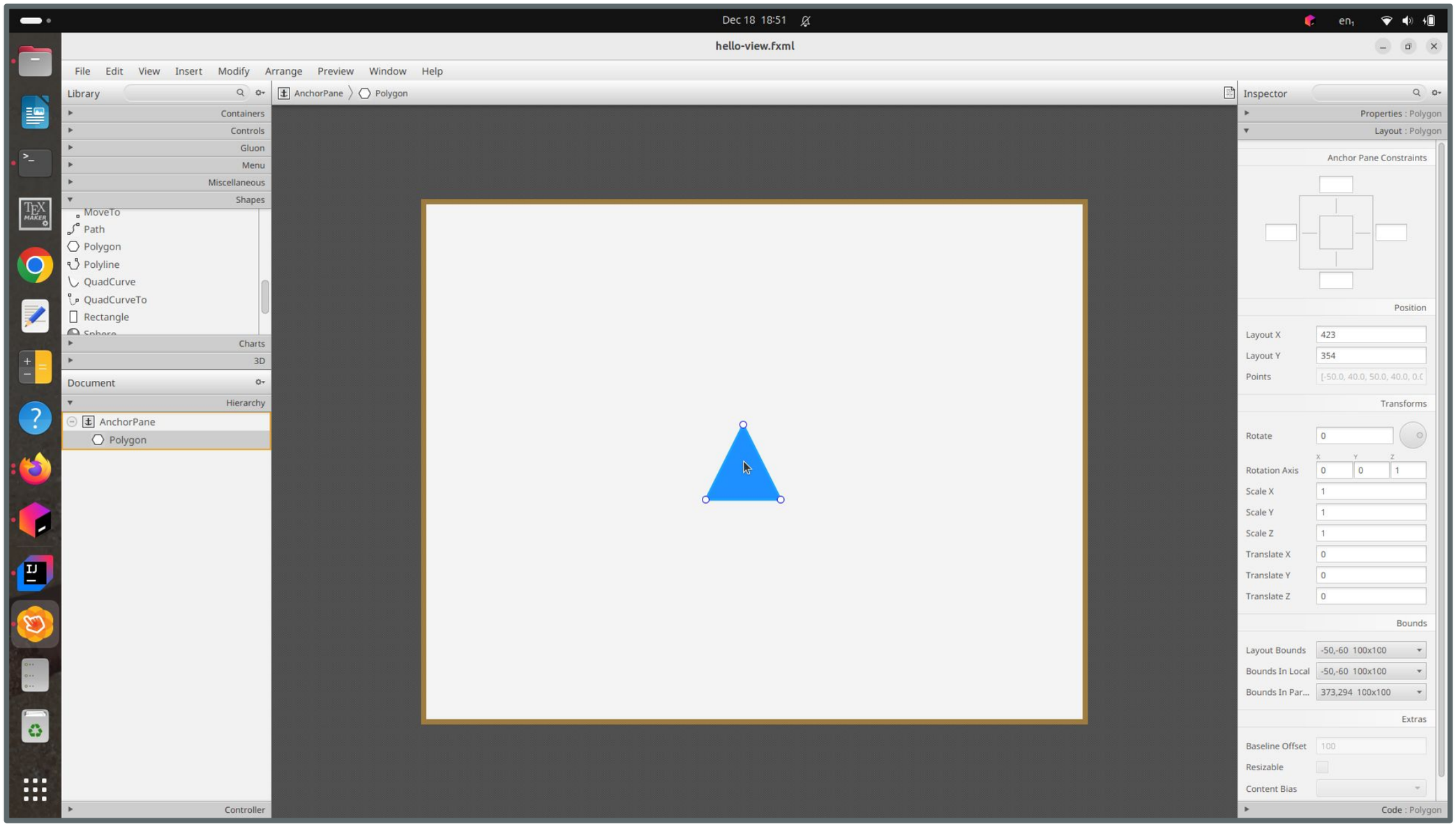
Package Hierarchies:
All Packages

Class Hierarchy

- java.lang.**Object**
  - javafx.scene.shape.**Mesh**
    - javafx.scene.shape.**TriangleMesh**
  - javafx.scene.**Node** (implements javafx.event.EventTarget, javafx.css.Styleable)
    - javafx.scene.shape.**Shape**
      - javafx.scene.shape.**Arc**
      - javafx.scene.shape.**Circle**
      - javafx.scene.shape.**CubicCurve**
      - javafx.scene.shape.**Ellipse**
      - javafx.scene.shape.**Line**
      - javafx.scene.shape.**Path**
      - javafx.scene.shape.**Polygon**
      - javafx.scene.shape.**Polyline**
      - javafx.scene.shape.**QuadCurve**
      - javafx.scene.shape.**Rectangle**
      - javafx.scene.shape.**SVGPath**
    - javafx.scene.shape.**Shape3D**
      - javafx.scene.shape.**Box**
      - javafx.scene.shape.**Cylinder**
      - javafx.scene.shape.**MeshView**
      - javafx.scene.shape.**Sphere**
  - javafx.scene.shape.**PathElement**
    - javafx.scene.shape.**ArcTo**
    - javafx.scene.shape.**ClosePath**
    - javafx.scene.shape.**CubicCurveTo**
    - javafx.scene.shape.**HLineTo**
    - javafx.scene.shape.**LineTo**
    - javafx.scene.shape.**MoveTo**
    - javafx.scene.shape.**QuadCurveTo**
    - javafx.scene.shape.**VLineTo**
  - javafx.scene.shape.**VertexFormat**

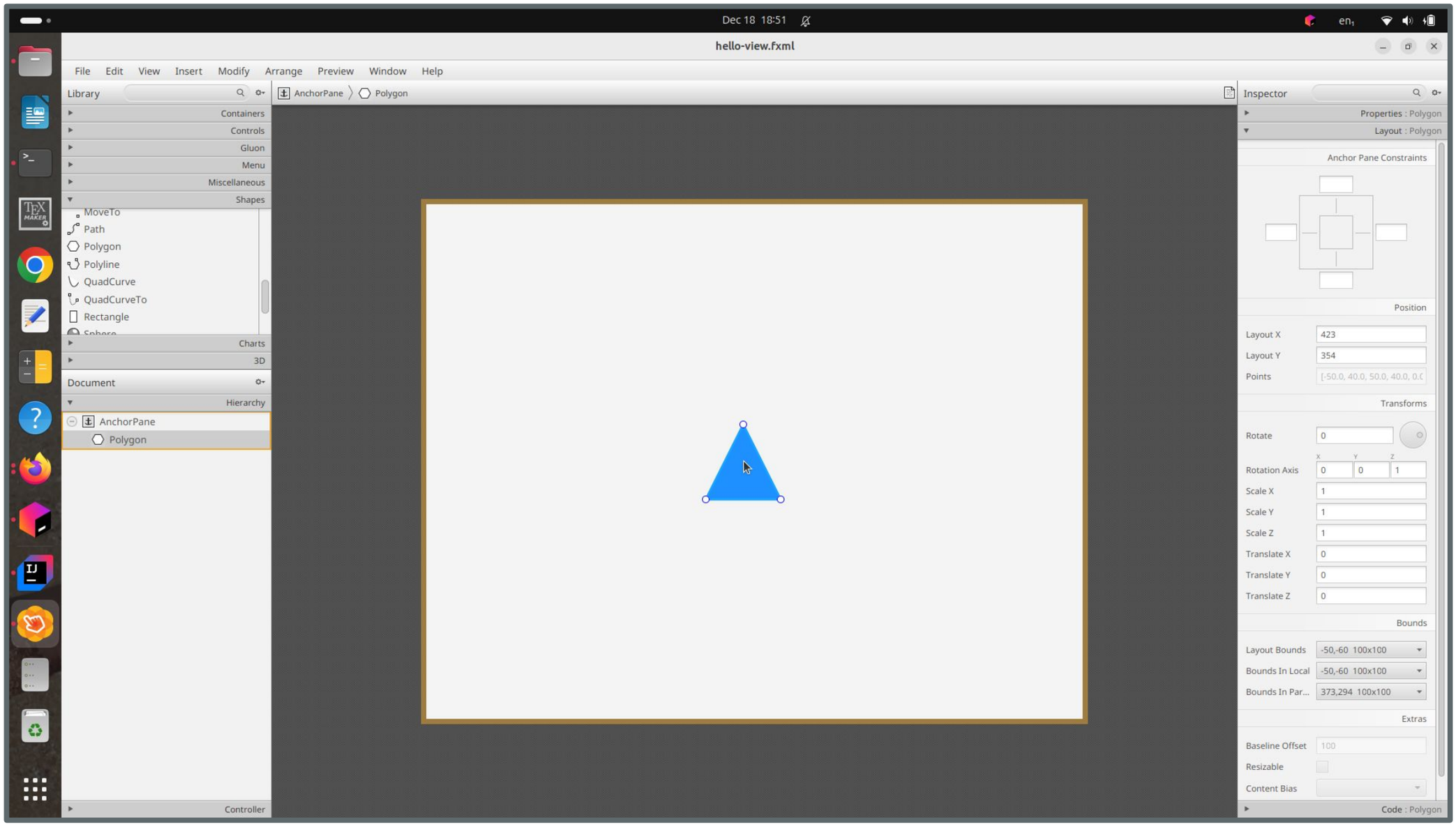
Add a Polygon to the AnchorPane

- Drag and drop a **Polygon** from the Shapes in Library panel into the AnchorPane.



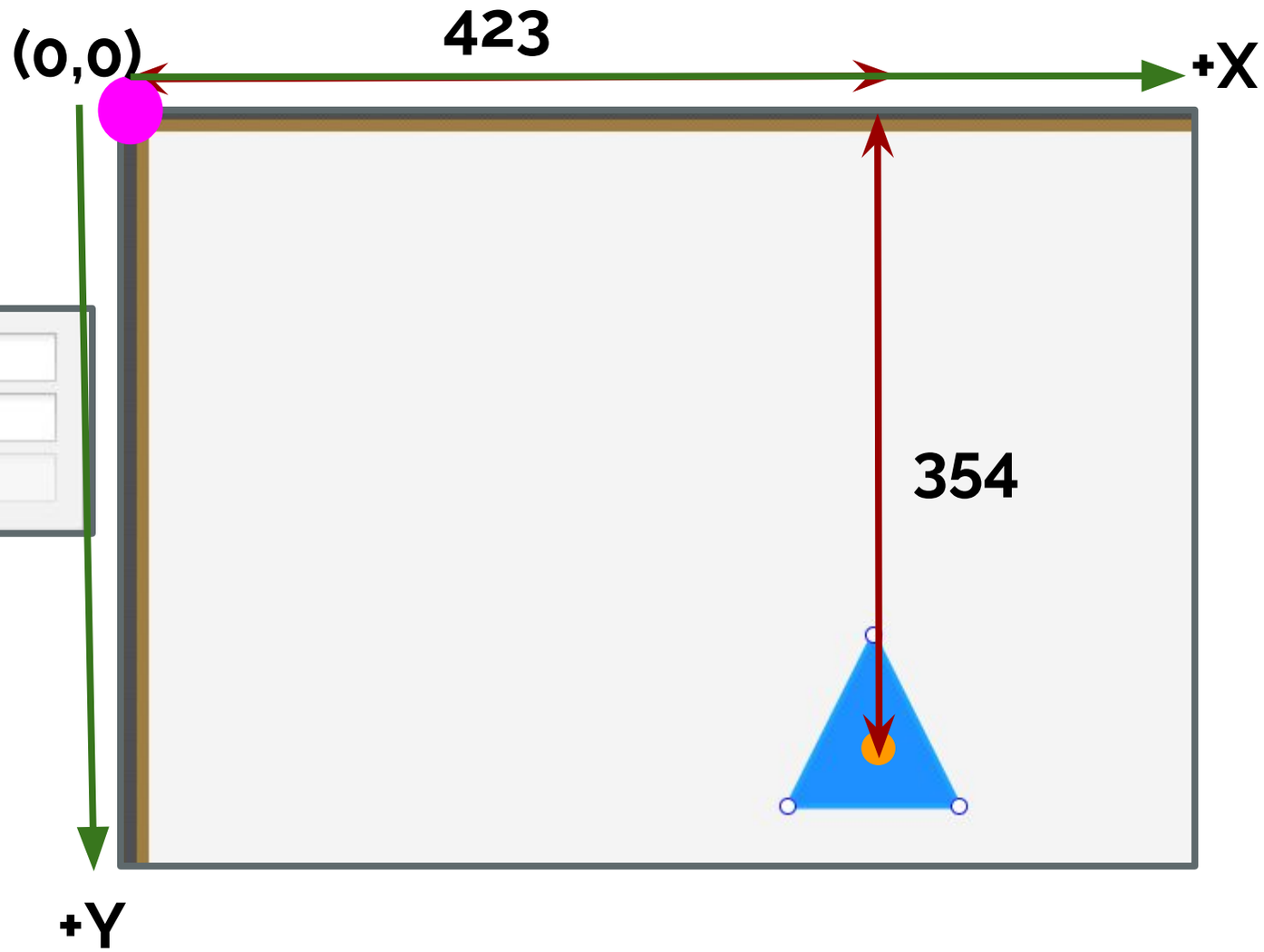
Polygon Properties

- By default, a triangle is added (not a Hexagon).
- Let us examine the Triangle Layout.



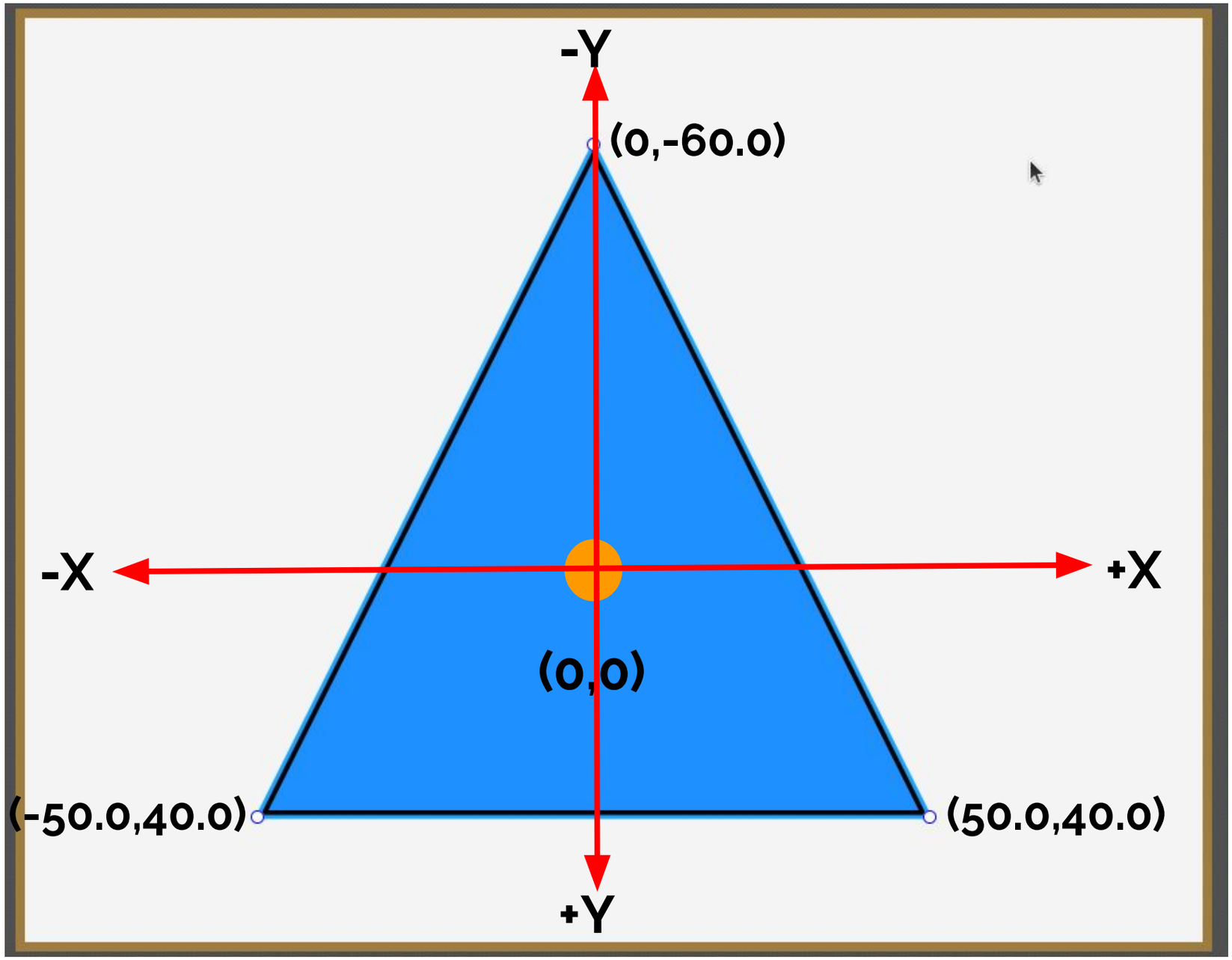
Polygon Properties

- In the Inspector Panel and Layout section, you will see three properties, **Layout X**, **Layout Y**, **Points**.
- **Unfortunately, Points is not editable.**
- **Layout X** and **Layout Y** are distances from the top right tip of the Anchor Pane ( ).



Polygon Points

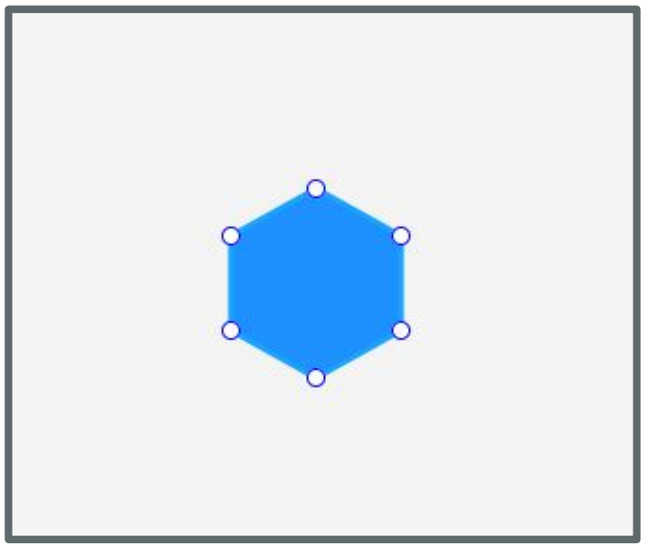
- The Points are the following:
  [-50.0, 40.0, 50.0, 40.0, 0.0, -60.0]
- A point is represented by a pair of coordinates (since this is a 2D board).
- I have scaled the triangle, which has no effect on the points.



Edit Polygon Points

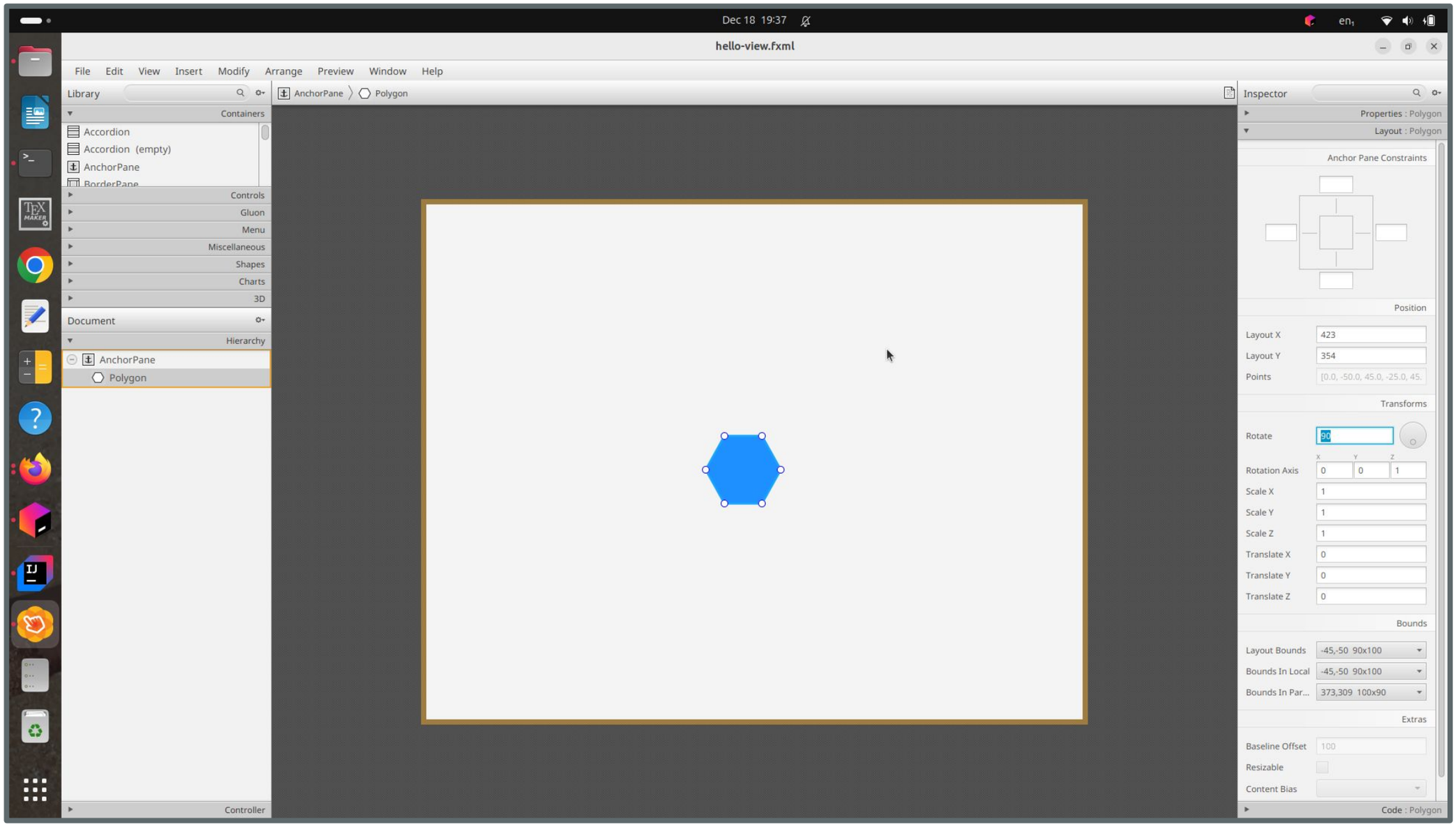
- To edit the polygon, you have to edit the FXML file.
- Under the **<points>** tag, I have provided six pairs of coordinates (six points of a hexagon).
- I will show you a method how to generate these points programmatically later.

```
<AnchorPane maxHeight="-Infinity" maxWidth="-Inf
   <children>
      <Polygon fill="DODGERBLUE" layoutX="423.0"
        <points>
            <Double fx:value="0.0" />
            <Double fx:value="-50.0" />
            <Double fx:value="45" />
            <Double fx:value="-25.0" />
            <Double fx:value="45" />
            <Double fx:value="25.0" />
            <Double fx:value="0.0" />
            <Double fx:value="50.0" />
            <Double fx:value="-45" />
            <Double fx:value="25.0" />
            <Double fx:value="-45" />
            <Double fx:value="-25.0" />
        </points>
      </Polygon>
   </children>
</AnchorPane>
```



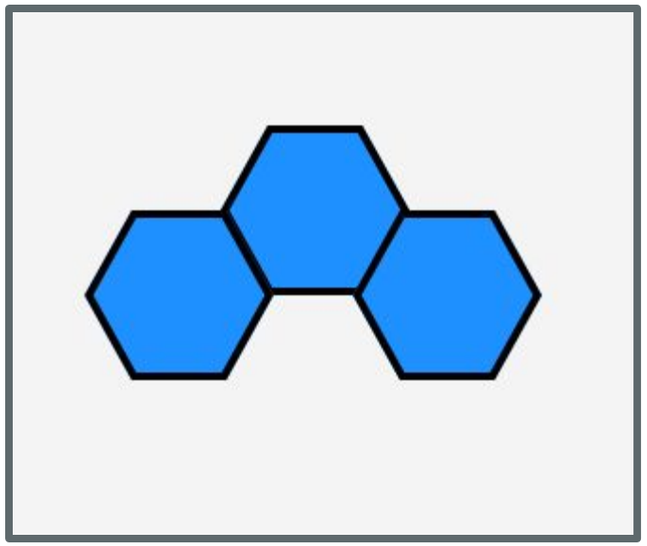
Pointy Top or Flat Top Hexagons

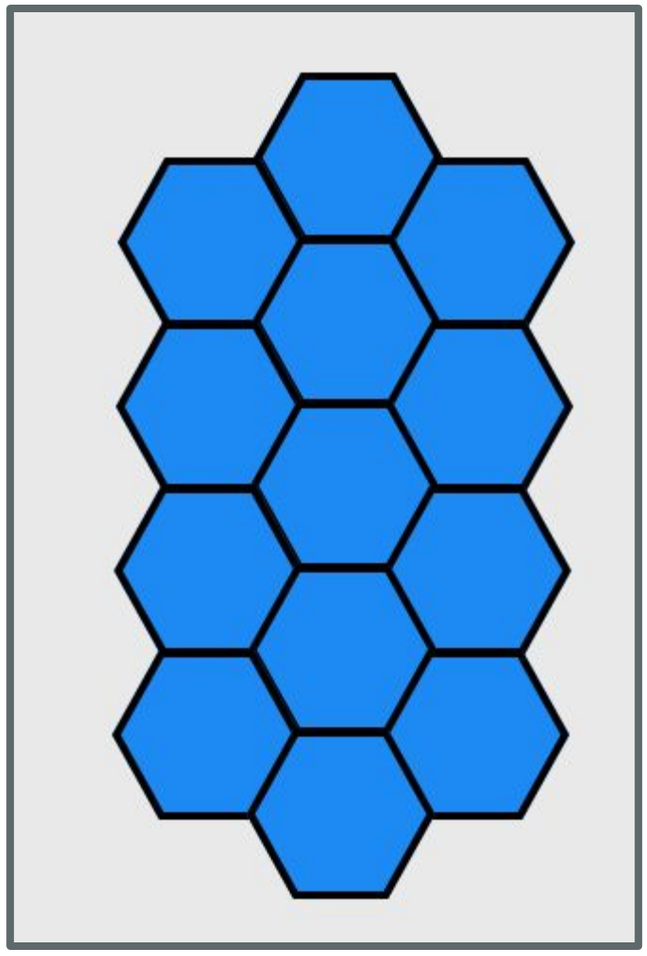
- You can change the orientation of the hexagon (if your board game hexagons are not pointy top but flat top).
- In **Layout** section, under **Rotate** property, provide **90**.

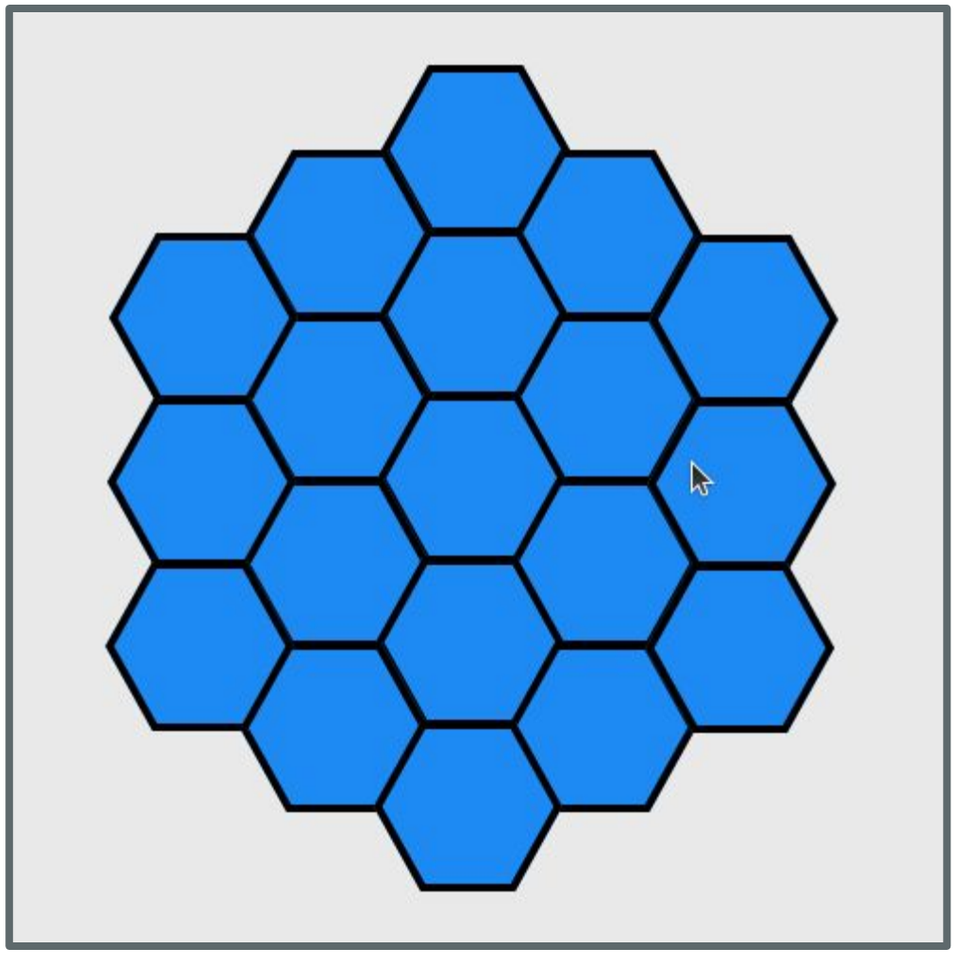


Create a Hexagonal Board

- Now duplicate the hexagon to create the required board.
- I have created a base-3 hexagonal board.

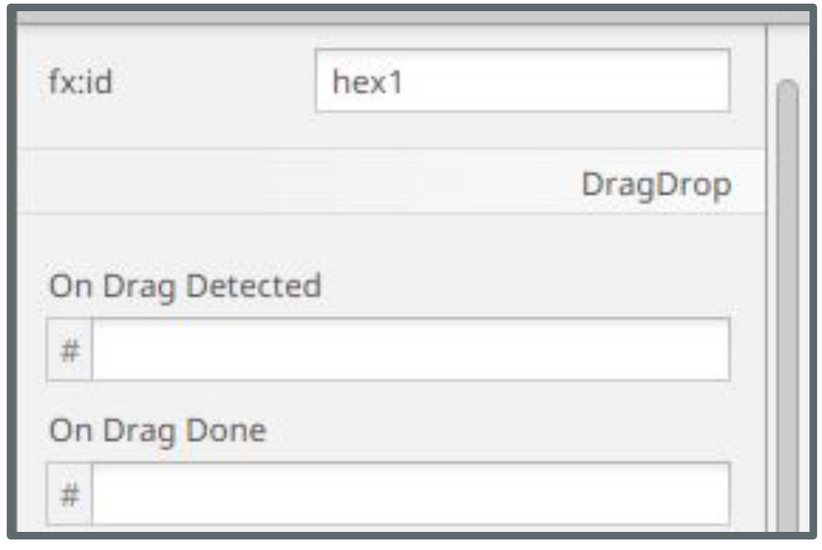


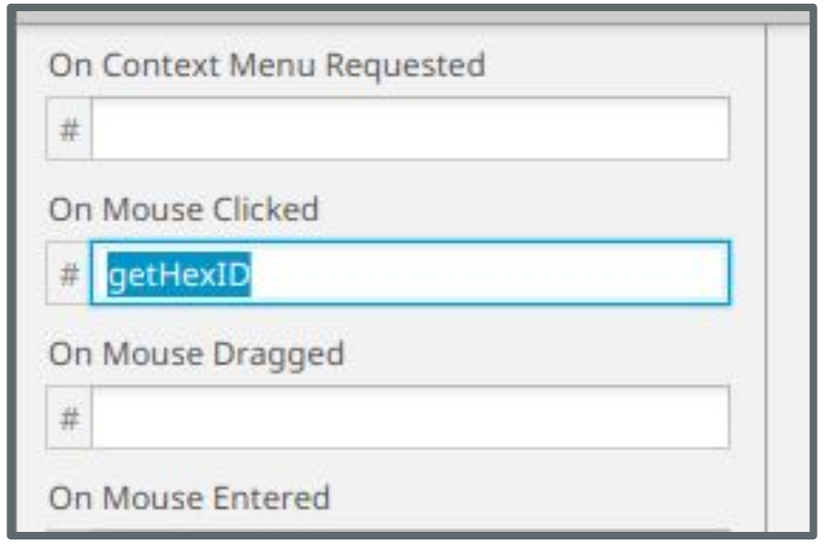




fx:id for the Hexagons (1/4)

- A GUI element is identified in the application code by an **fx:id**.
- We need to provide unique IDs for all the hexagons.
- In the Inspector panel, under Code section, you can give the fx:id.
- We will also provide a method (**getHexID**) to be invoked when a mouse click happens over a hexagon.
- Click the top-most hexagon and provide the following:





fx:id for the Hexagons (2/4)

- Close the Scene Builder.
- In the IntelliJ IDE, you will see that the top most hexagon gets an **fx:id** and **onMouseClicked** attributes.

```
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="720
   <children>
      <Polygon fx:id="hex1" fill="DODGERBLUE" layoutX="436.0" layoutY="153.0" onMouseClicked="#getHexID" rotate="
        <points>
            <Double fx:value="0.0" />
            <Double fx:value="-50.0" />
            <Double fx:value="45" />
            <Double fx:value="-25.0" />
            <Double fx:value="45" />
            <Double fx:value="25.0" />
            <Double fx:value="0.0" />
            <Double fx:value="50.0" />
            <Double fx:value="-45" />
            <Double fx:value="25.0" />
            <Double fx:value="-45" />
            <Double fx:value="-25.0" />
        </points>
    </Polygon>
```

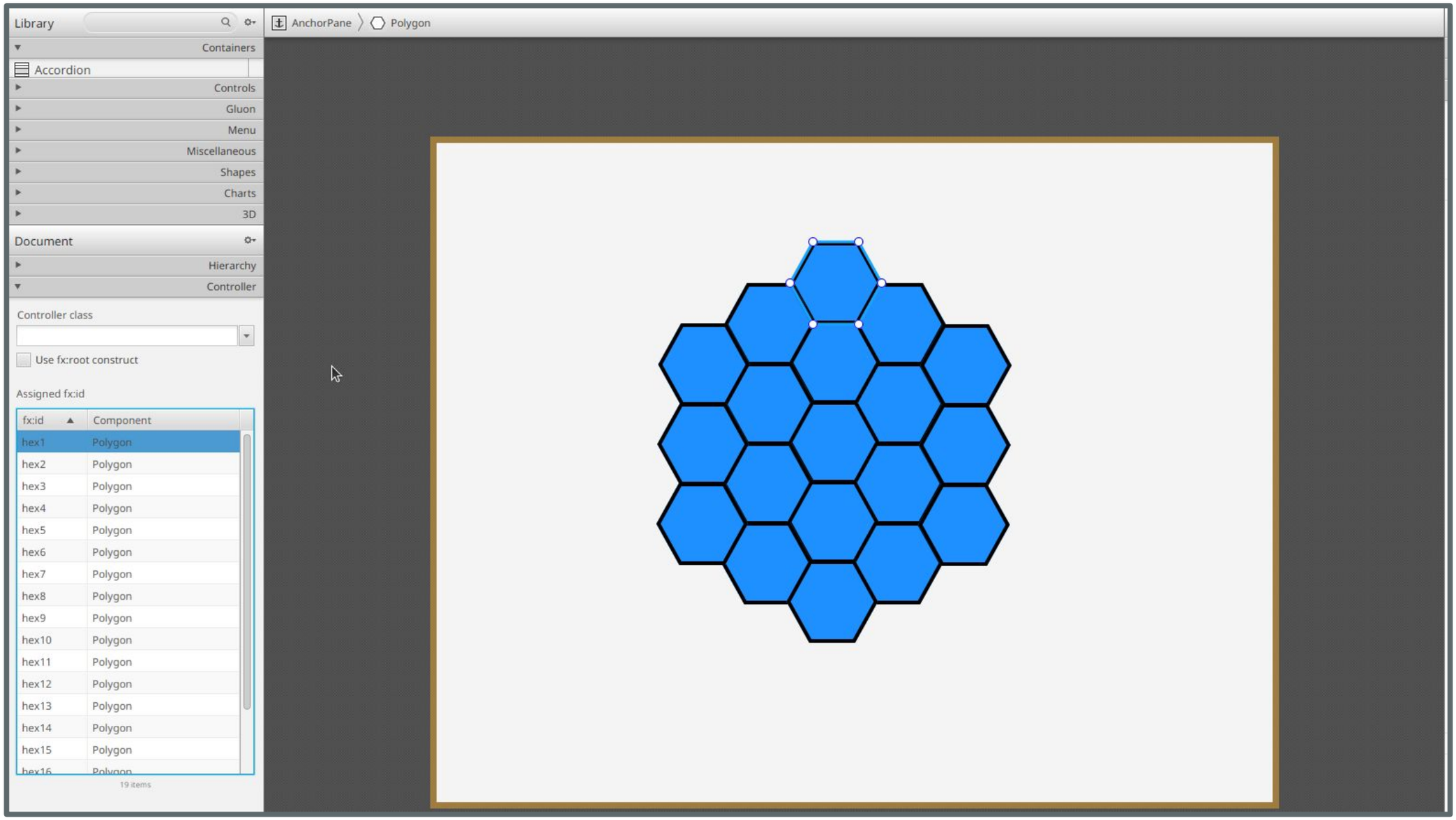
fx:id for the Hexagons (3/4)

- Edit the FXML and provide unique fx:id for all the hexagons.
- Similarly, provide the name-value pair (**onMouseClicked="#getHexID"**) for all the hexagons.

```
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="720
   <children>
      <Polygon fx:id="hex1" fill="DODGERBLUE" layoutX="436.0" layoutY="153.0" onMouseClicked="#getHexID" rotate="
        <points>
            <Double fx:value="0.0" />
            <Double fx:value="-50.0" />
            <Double fx:value="45" />
            <Double fx:value="-25.0" />
            <Double fx:value="45" />
            <Double fx:value="25.0" />
            <Double fx:value="0.0" />
            <Double fx:value="50.0" />
            <Double fx:value="-45" />
            <Double fx:value="25.0" />
            <Double fx:value="-45" />
            <Double fx:value="-25.0" />
        </points>
      </Polygon>
```

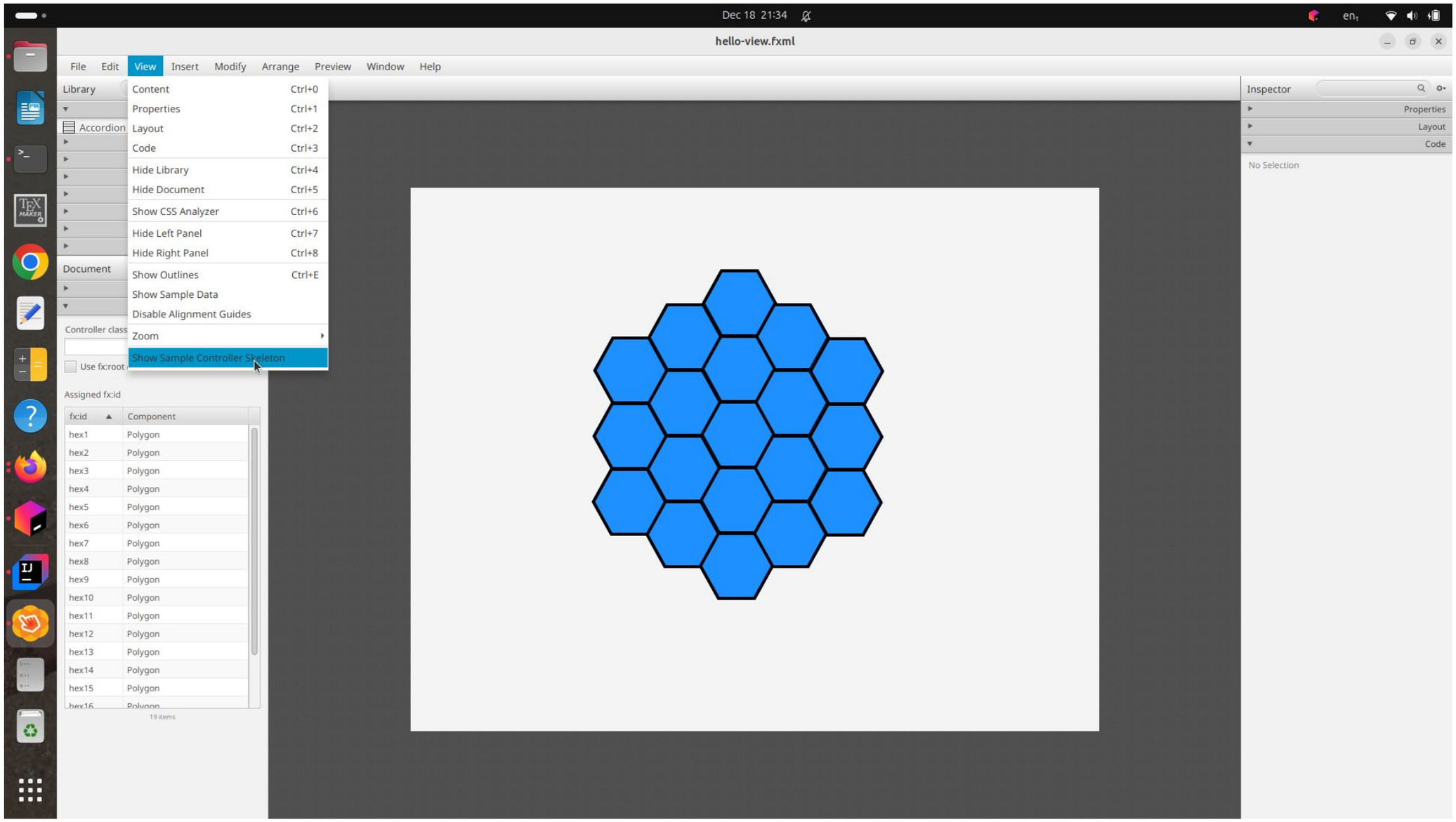
fx:id for the Hexagons (4/4)

- Check the fx:id for all the hexagons in the Scene Builder.
- Check the **Controller** section in the **Document** panel.



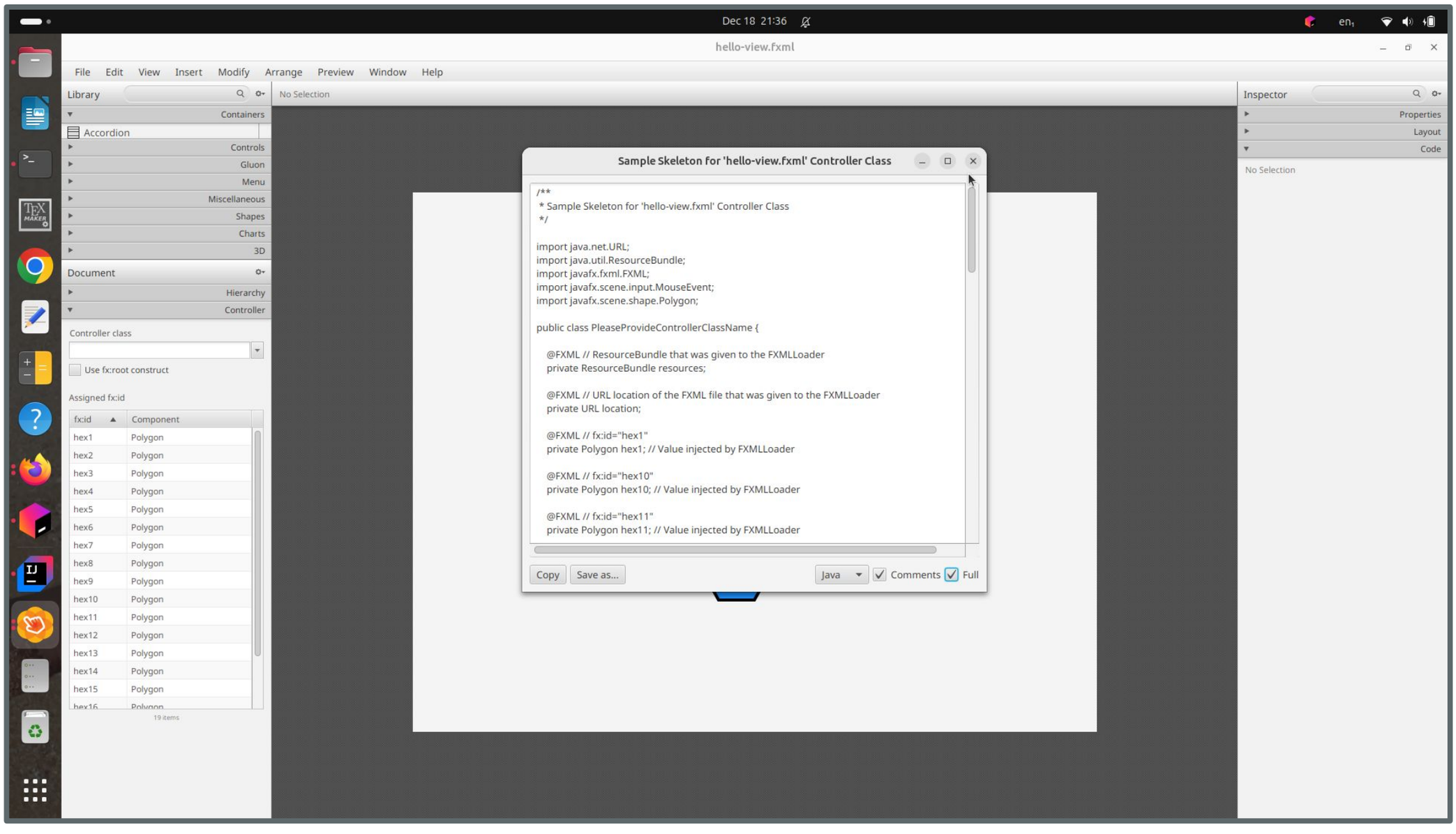
Copy Controller Source

- Use the sample Controller source as the starting point.
- In Scene Builder, click **View** and Select **Show Sample Controller Skeleton**.



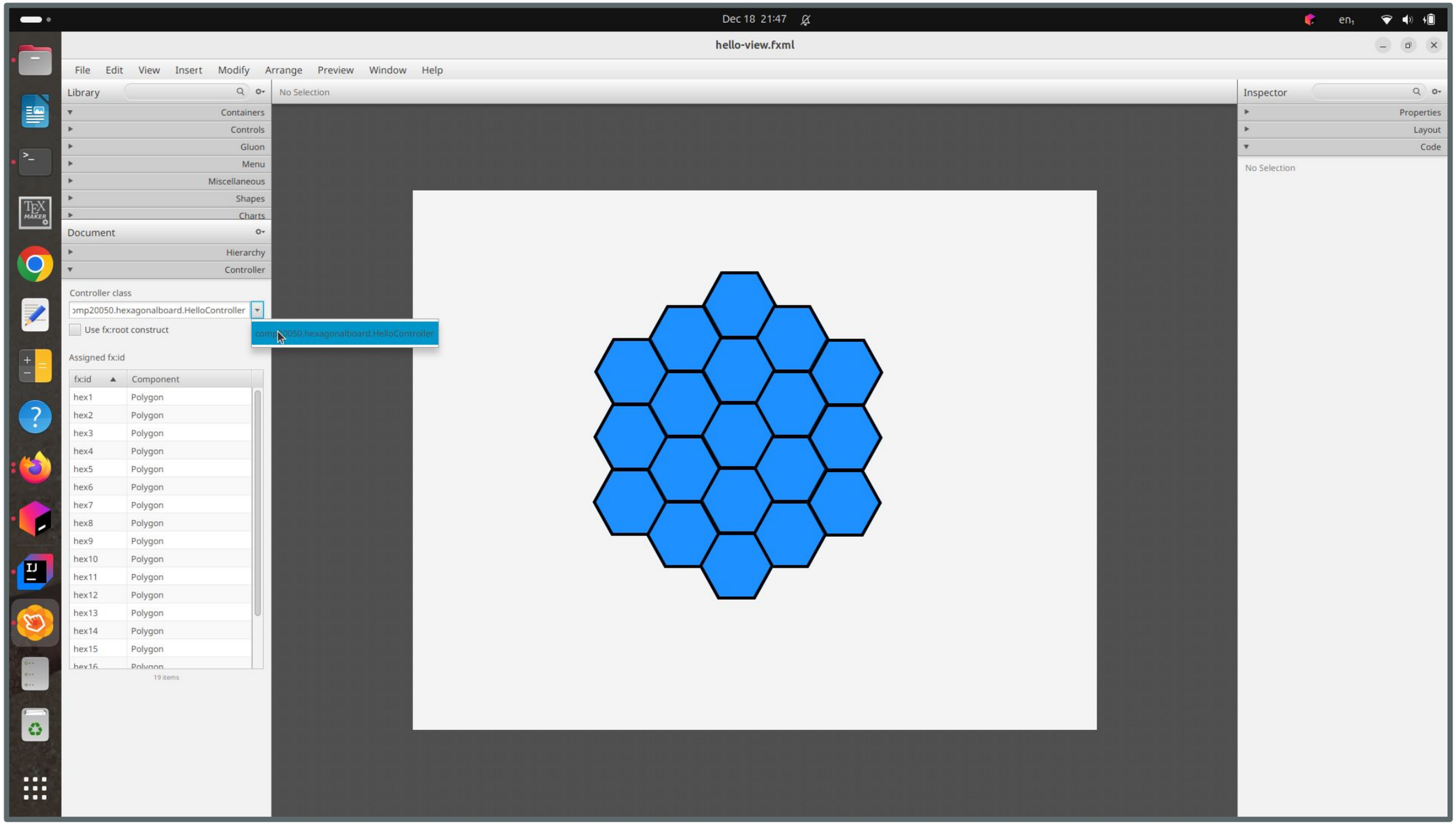
Modify Controller in Application

- Tick all the checkboxes.
- Copy and Paste from the skeleton into the HelloController.java source.



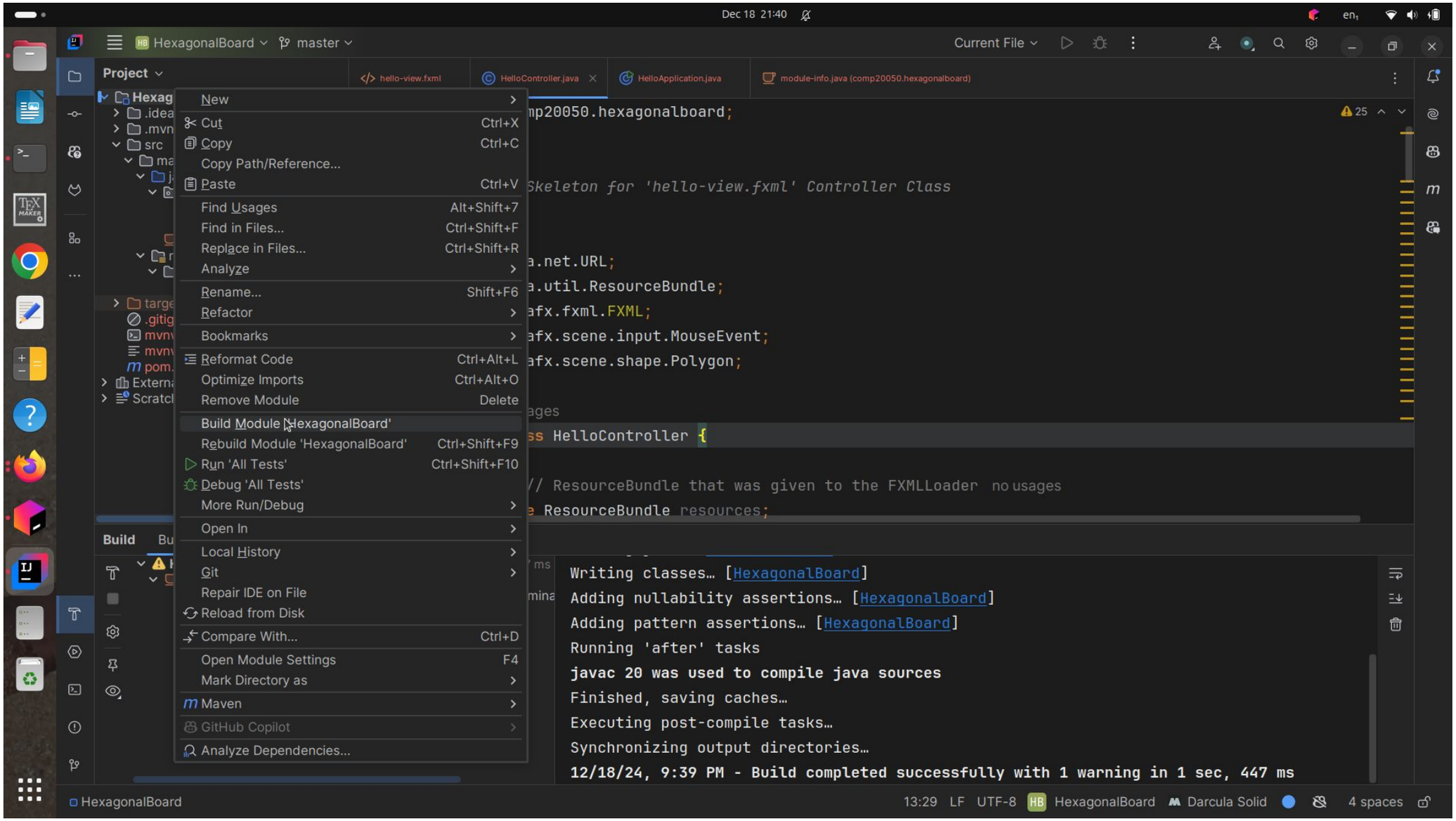
Provide Controller Class in SceneBuilder

- Tick all the checkboxes.
- Copy and Paste into the HelloController.java source.



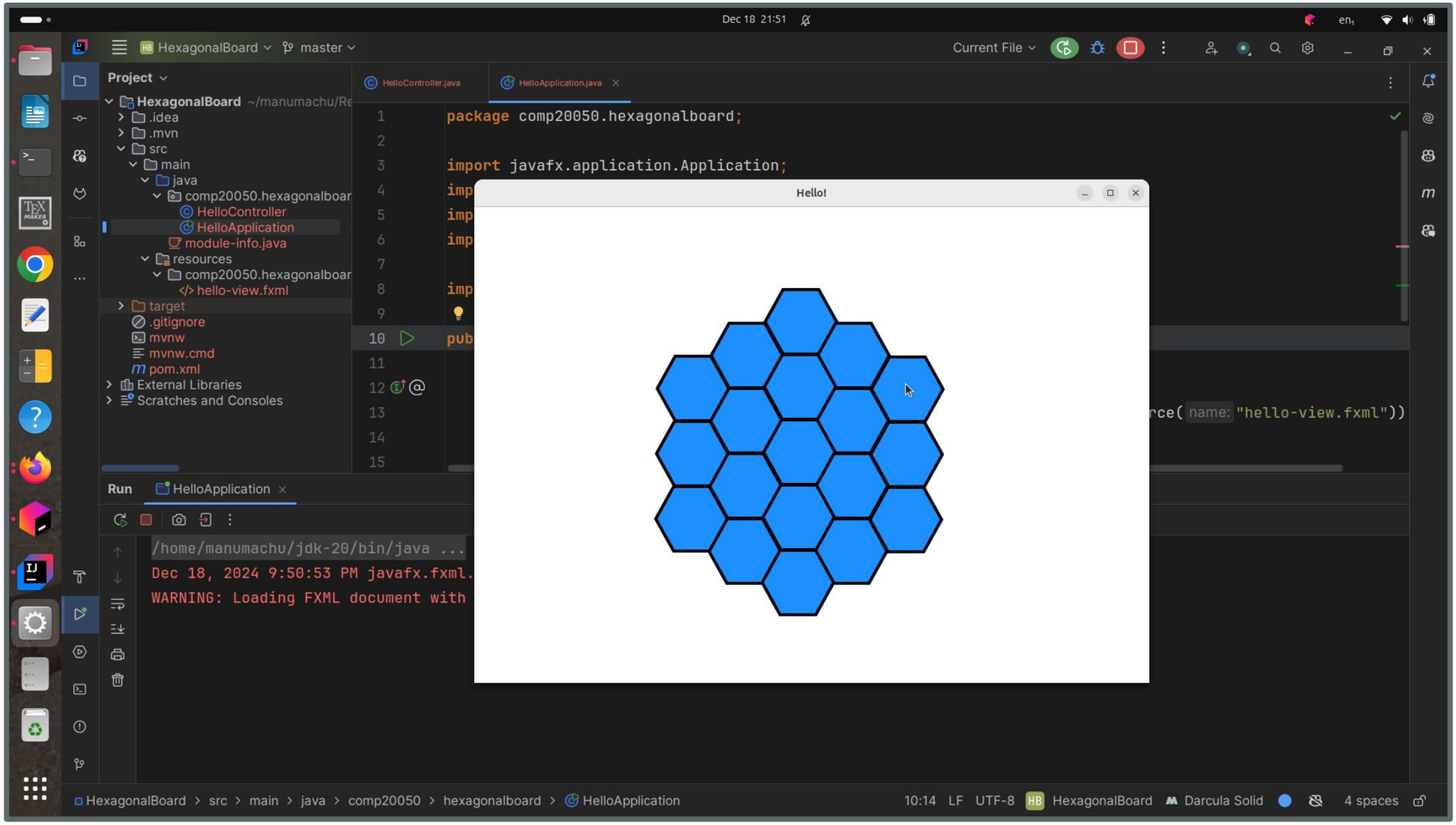
Build Application

- Build the application before you add any more functionality.



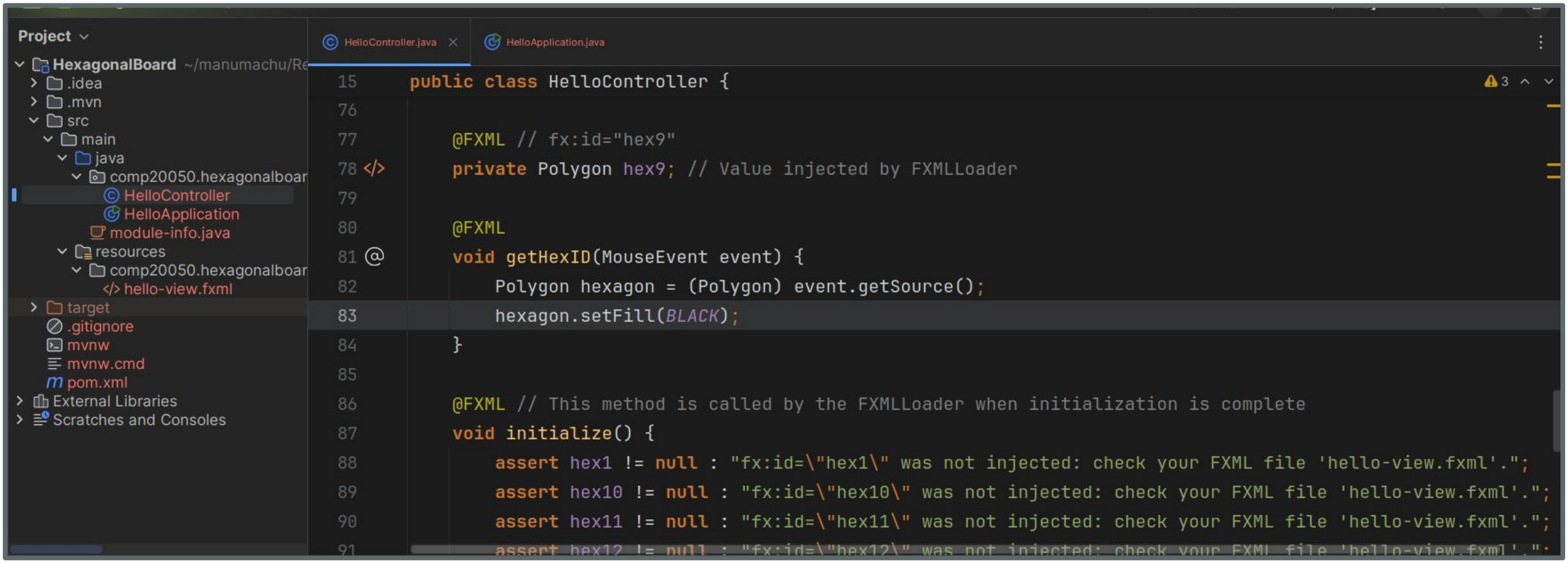
Run Application

- Run the application.



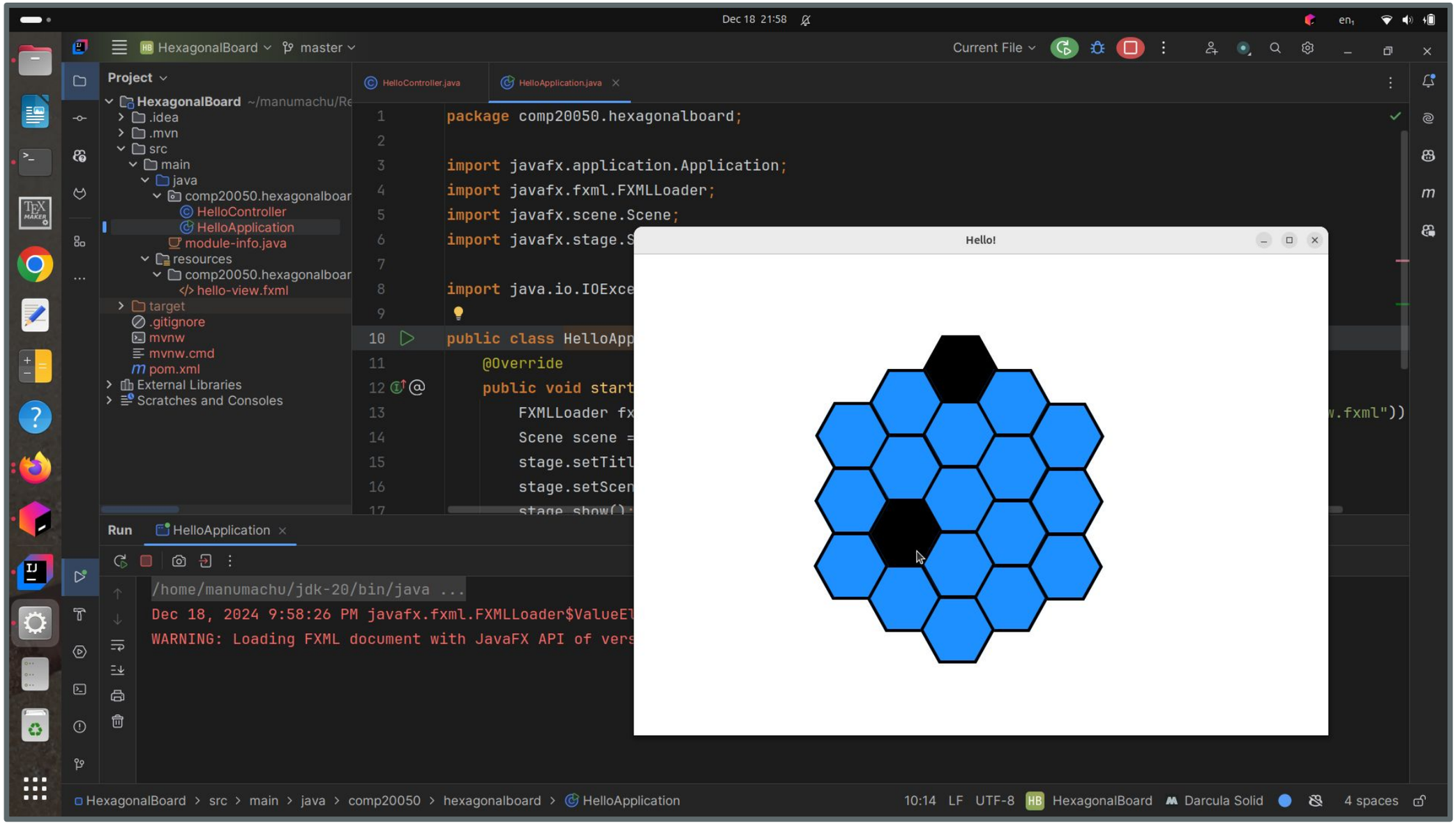
Handle Events

- We will set the hexagon to **BLACK** when clicked.
- The **getHexID()** implementation is shown below.



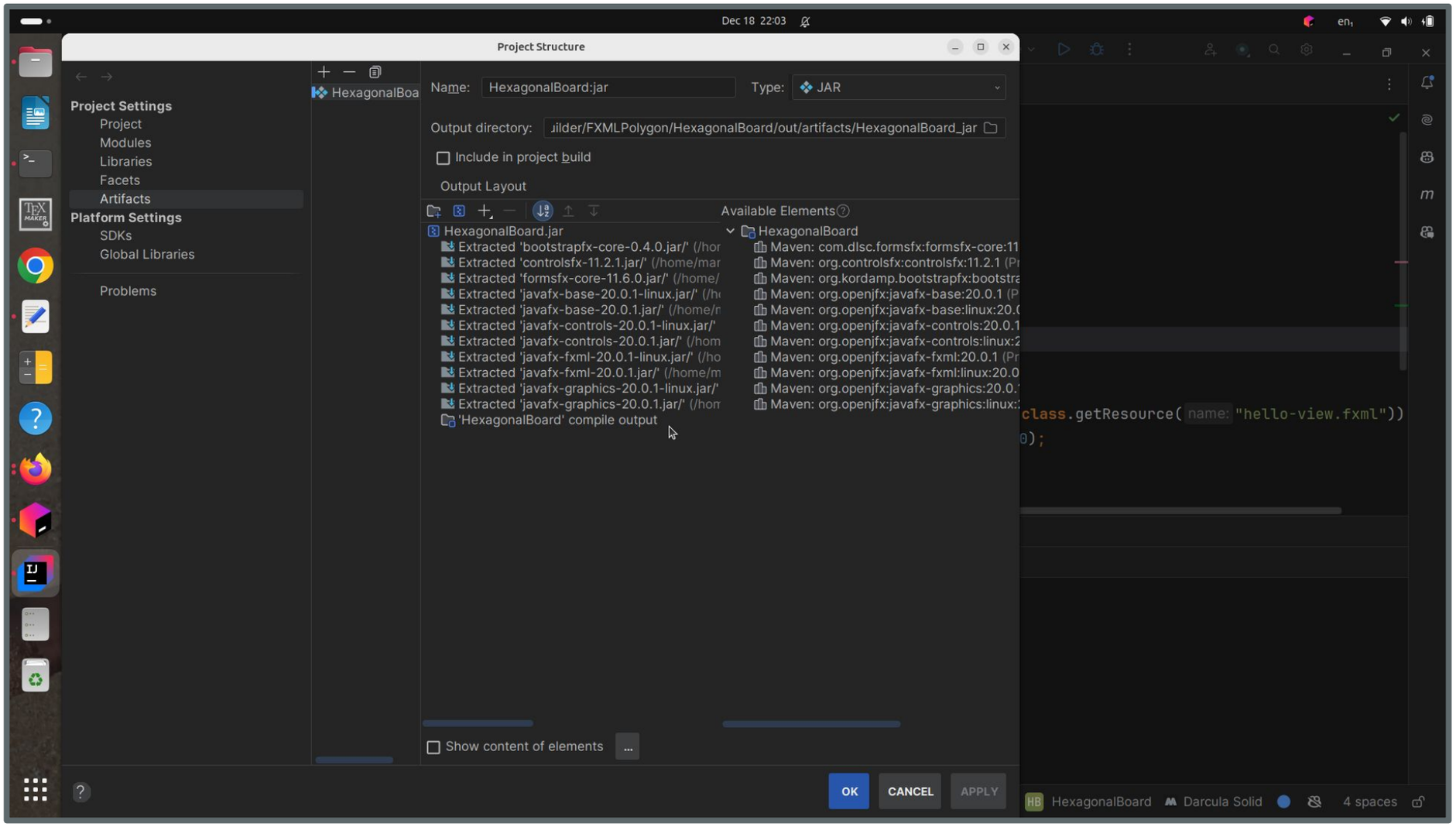
Rebuild and Run

- Rebuild and Run the application.



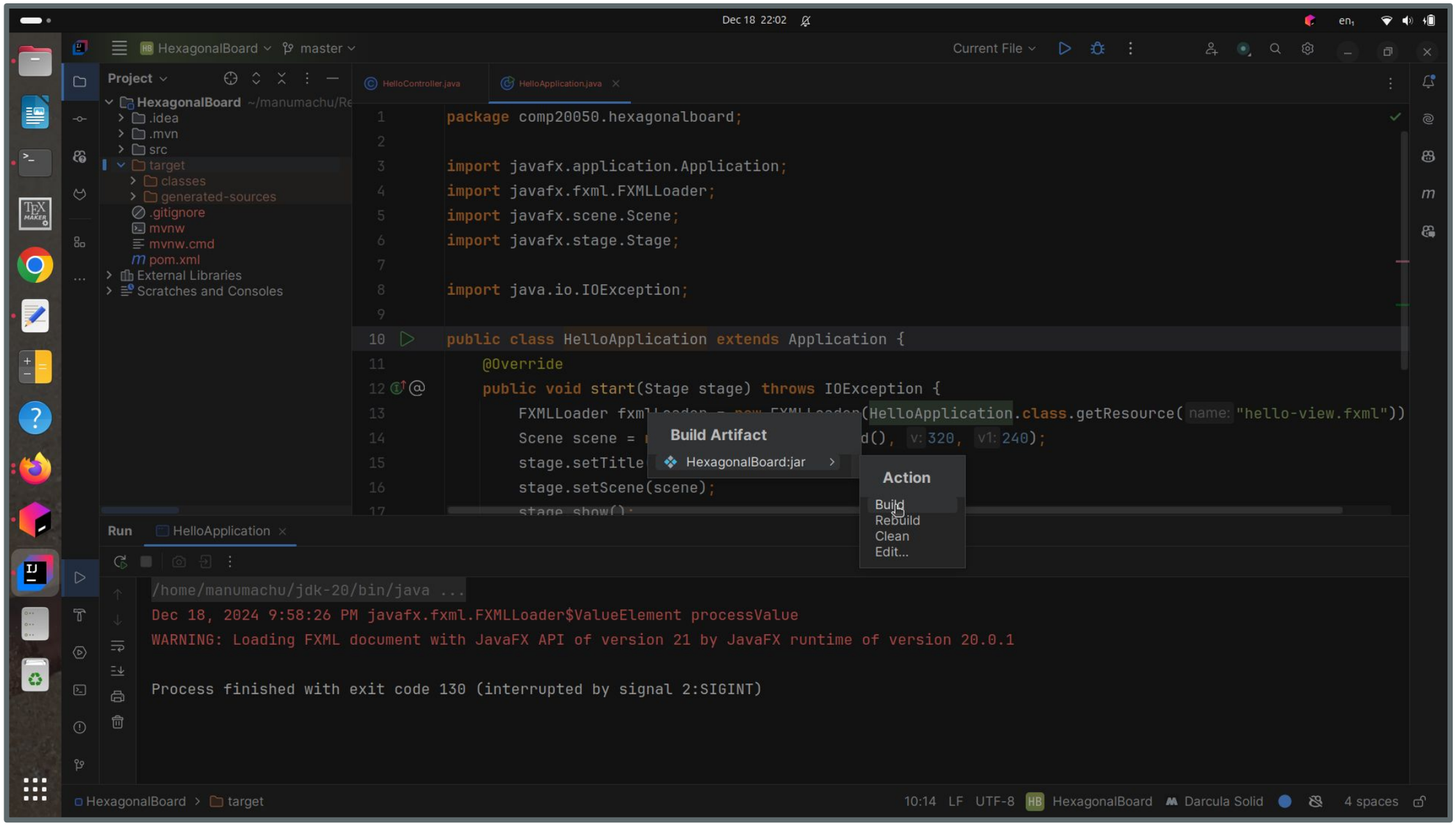
Build a Release Package

- Follow the instructions at the URL below:
  https://www.jetbrains.com/help/idea/compiling-applications.html#package_into_jar
  File -> Project Structure -> Click Artifacts -> Click + -> JAR ->
  From modules with dependencies -> Select the main class -> Click Apply



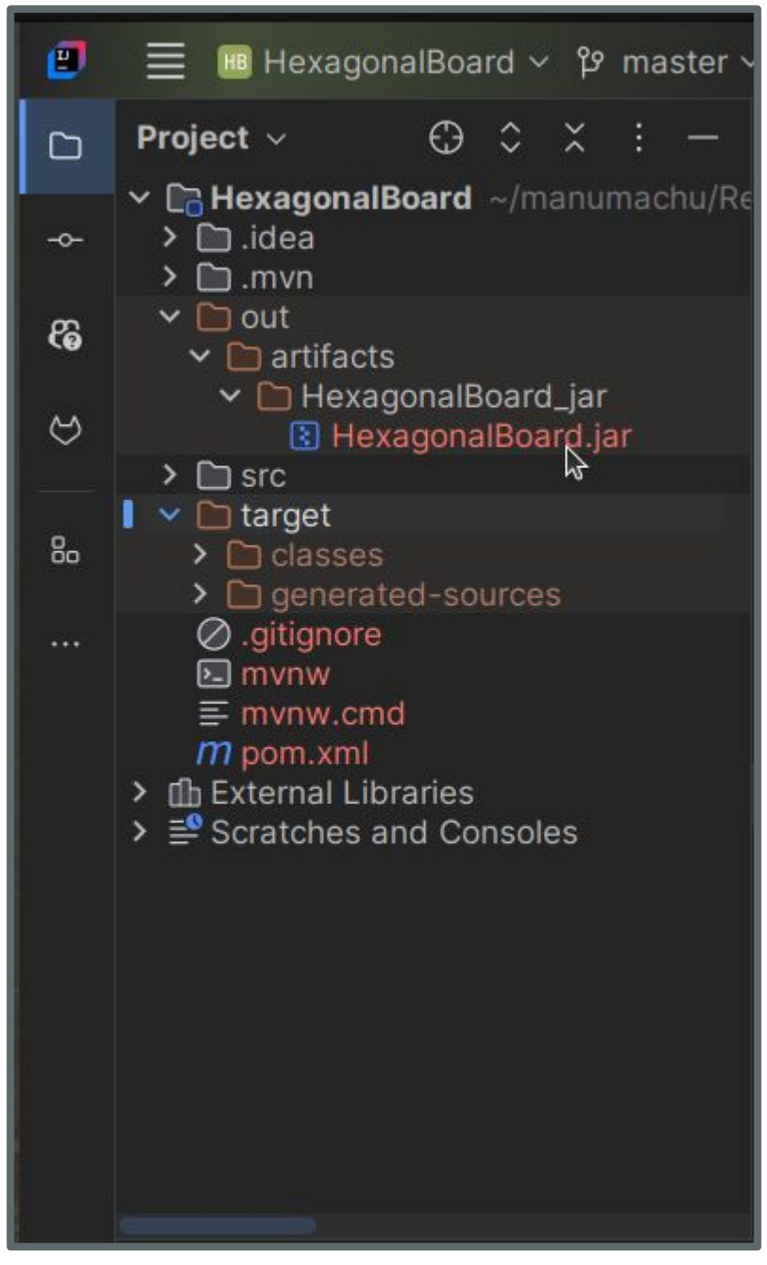
Build Artifact

- Menu -> Build -> Build Artifacts… -> Build



Release JAR File

- The jar file **HexagonalBoard.jar** is in **artifacts** folder.



Q&A

To follow...

Introduction to libGDX