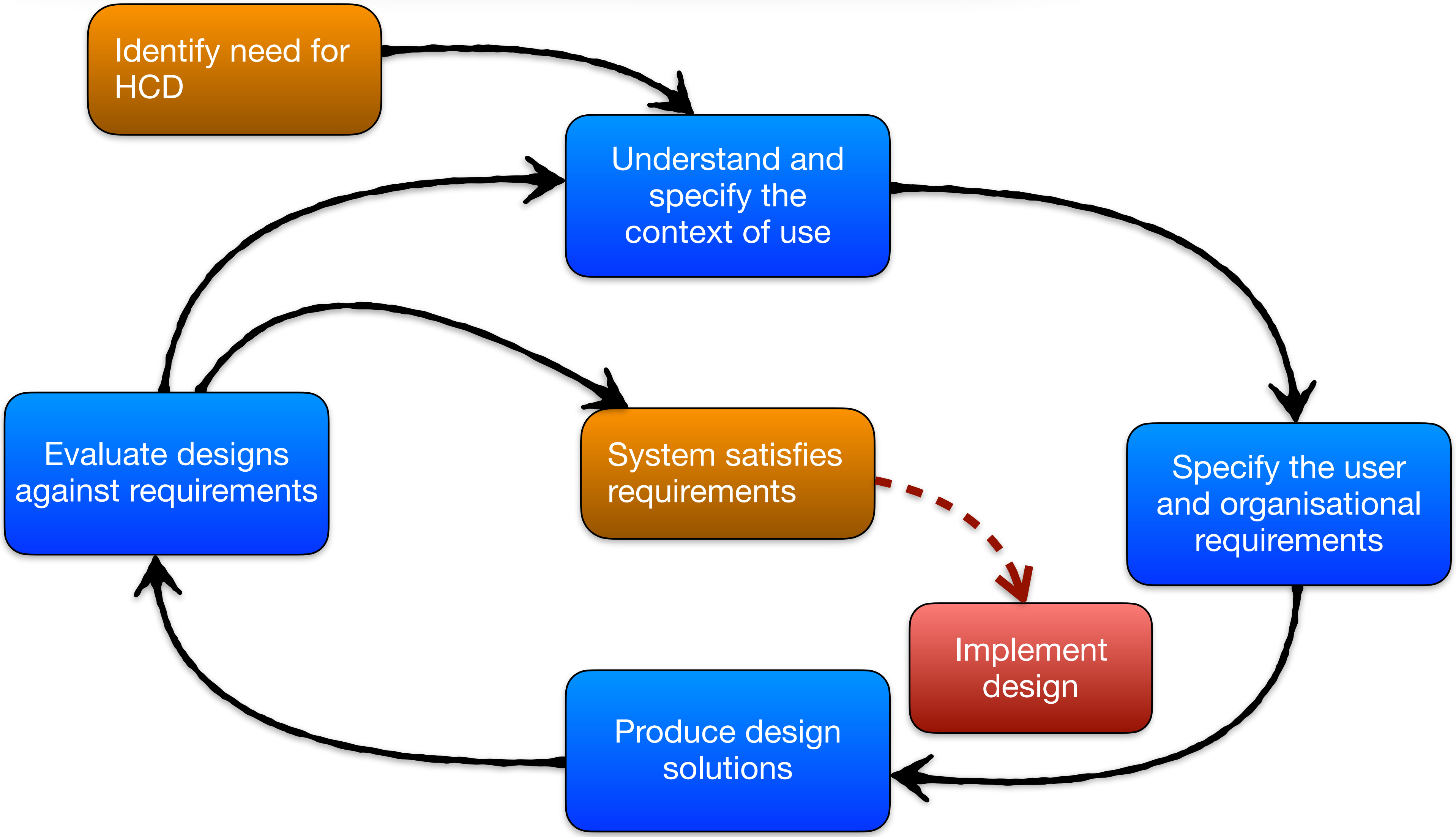
Módulo 8

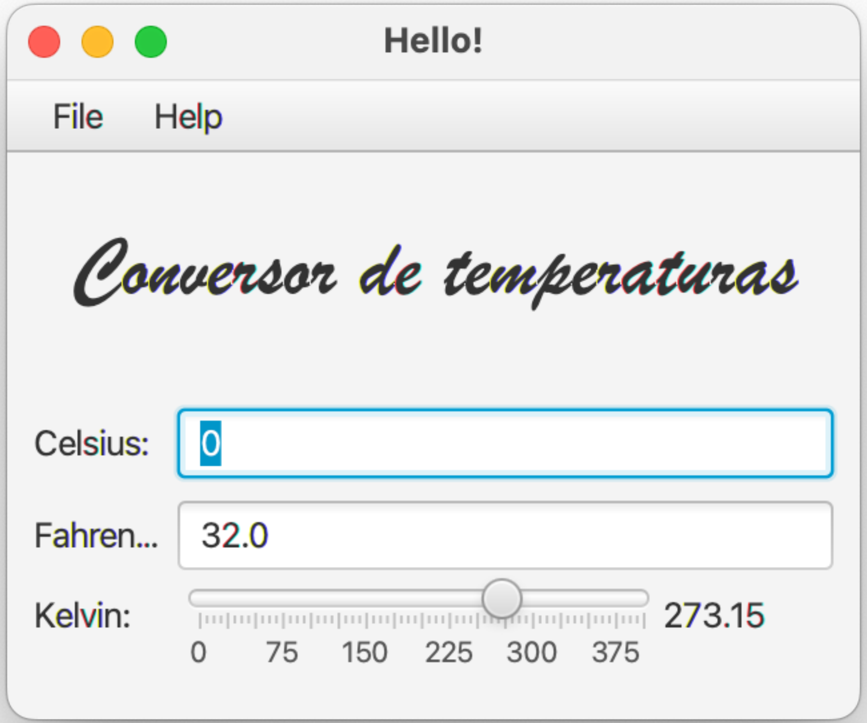
JAVAFX

Human-centred design (HCD)

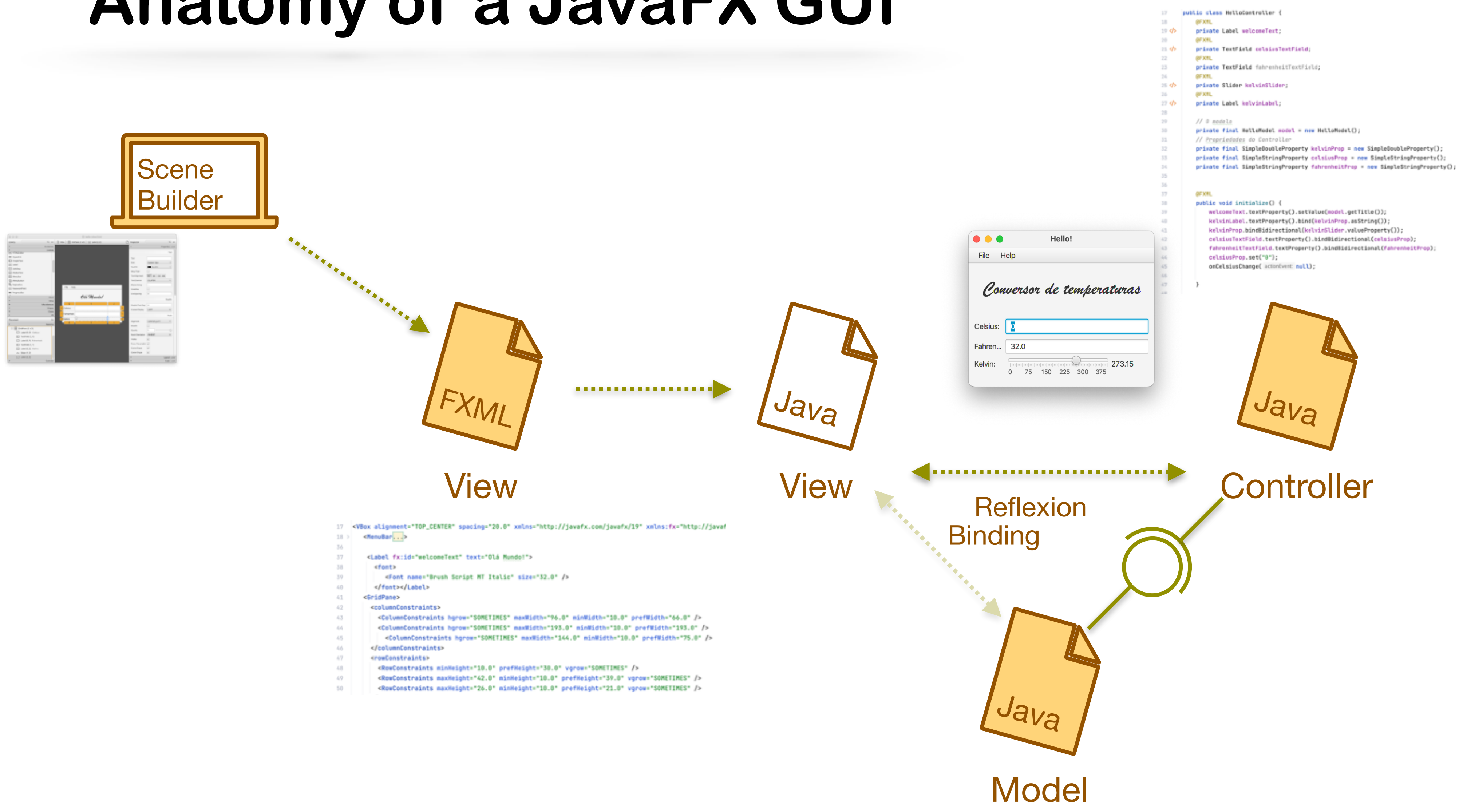

Identify need for HCD
Understand and specify the context of use
Specify the user and organisational requirements
Produce design solutions
Evaluate designs against requirements
System satisfies requirements
Implement design

JavaFX GUIs — an example

- We will develop a graphical user interface (GUI)
  - There are other types of user interfaces (UI)
- GUIs are built from GUI components (widgets or controls)
  - Example Java FX GUI:



Anatomy of a JavaFX GUI



```java
public class HelloController {
    @FXML
    private Label welcomeText;
    @FXML
    private TextField celsiusTextField;
    @FXML
    private TextField fahrenheitTextField;
    @FXML
    private Slider kelvinSlider;
    @FXML
    private Label kelvinLabel;

    // O modelo
    private final HelloModel model = new HelloModel();
    // Propriedades do Controller
    private final SimpleDoubleProperty kelvinProp = new SimpleDoubleProperty();
    private final SimpleStringProperty celsiusProp = new SimpleStringProperty();
    private final SimpleStringProperty fahrenheitProp = new SimpleStringProperty();


    @FXML
    public void initialize() {
        welcomeText.textProperty().setValue(model.getTitle());
        kelvinLabel.textProperty().bind(kelvinProp.asString());
        kelvinProp.bindBidirectional(kelvinSlider.valueProperty());
        celsiusTextField.textProperty().bindBidirectional(celsiusProp);
        fahrenheitTextField.textProperty().bindBidirectional(fahrenheitProp);
        celsiusProp.set("0");
        onCelsiusChange( actionEvent: null);

    }
```

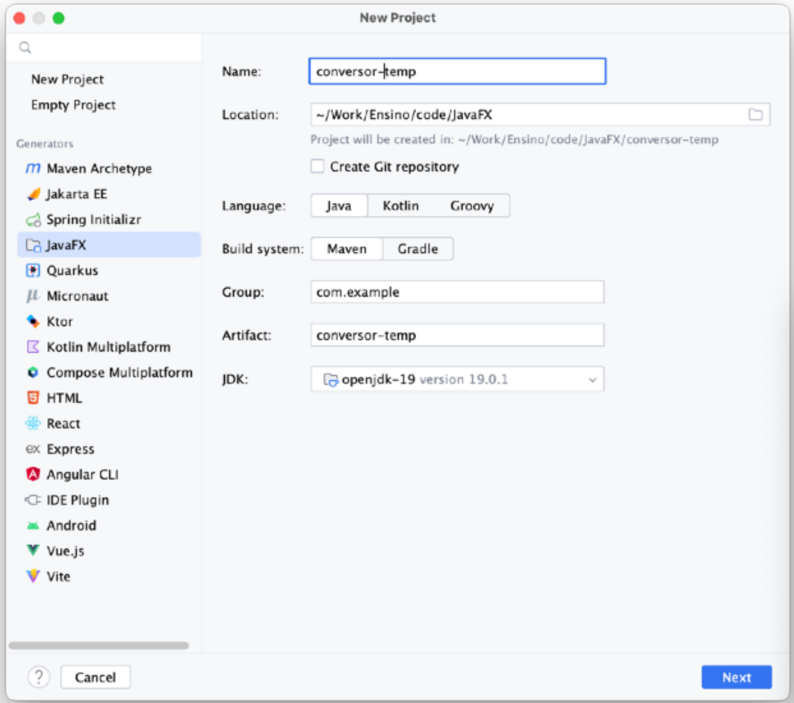
```xml
<VBox alignment="TOP_CENTER" spacing="20.0" xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javaf
  <MenuBar...>

   <Label fx:id="welcomeText" text="Olá Mundo!">
     <font>
        <Font name="Brush Script MT Italic" size="32.0" />
     </font></Label>
  <GridPane>
    <columnConstraints>
      <ColumnConstraints hgrow="SOMETIMES" maxWidth="96.0" minWidth="10.0" prefWidth="66.0" />
      <ColumnConstraints hgrow="SOMETIMES" maxWidth="193.0" minWidth="10.0" prefWidth="193.0" />
        <ColumnConstraints hgrow="SOMETIMES" maxWidth="144.0" minWidth="10.0" prefWidth="75.0" />
    </columnConstraints>
    <rowConstraints>
      <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
      <RowConstraints maxHeight="42.0" minHeight="10.0" prefHeight="39.0" vgrow="SOMETIMES" />
      <RowConstraints maxHeight="26.0" minHeight="10.0" prefHeight="21.0" vgrow="SOMETIMES" />
```

Anatomy of a JavaFX GUI

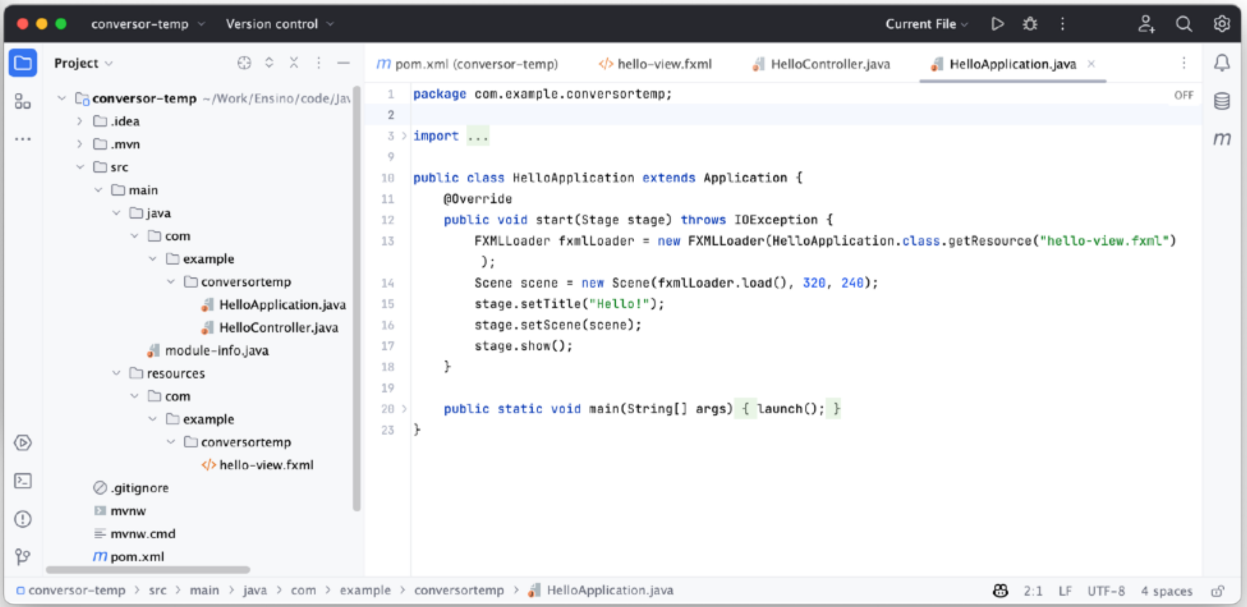
- Building a JavaFX UI involves
  - SceneBuilder – a GUI builder (drag-and-drop manipulation of widgets)
  - FXML – a configuration language (records the widgets in the GUI, their visible attributes and their relationship to each other)
  - A Controller class – defines the behaviour of the GUI (must be written by the programmer)
  - A Model class — provides access to the business logic

Structure of the code

- In IntelliJ IDEA…



Name:	conversor-temp
Location:	~/Work/Ensino/code/JavaFX
	Project will be created in: ~/Work/Ensino/code/JavaFX/conversor-temp
	Create Git repository
Language:	Java / Kotlin / Groovy
Build system:	Maven / Gradle
Group:	com.example
Artifact:	conversor-temp
JDK:	openjdk-19 version 19.0.1



```
conversor-temp ~/Work/Ensino/code/Jav
  .idea
  .mvn
  src
    main
      java
        com
          example
            conversortemp
              HelloApplication.java
              HelloController.java
        module-info.java
      resources
        com
          example
            conversortemp
              hello-view.fxml
  .gitignore
  mvnw
  mvnw.cmd
  pom.xml
```

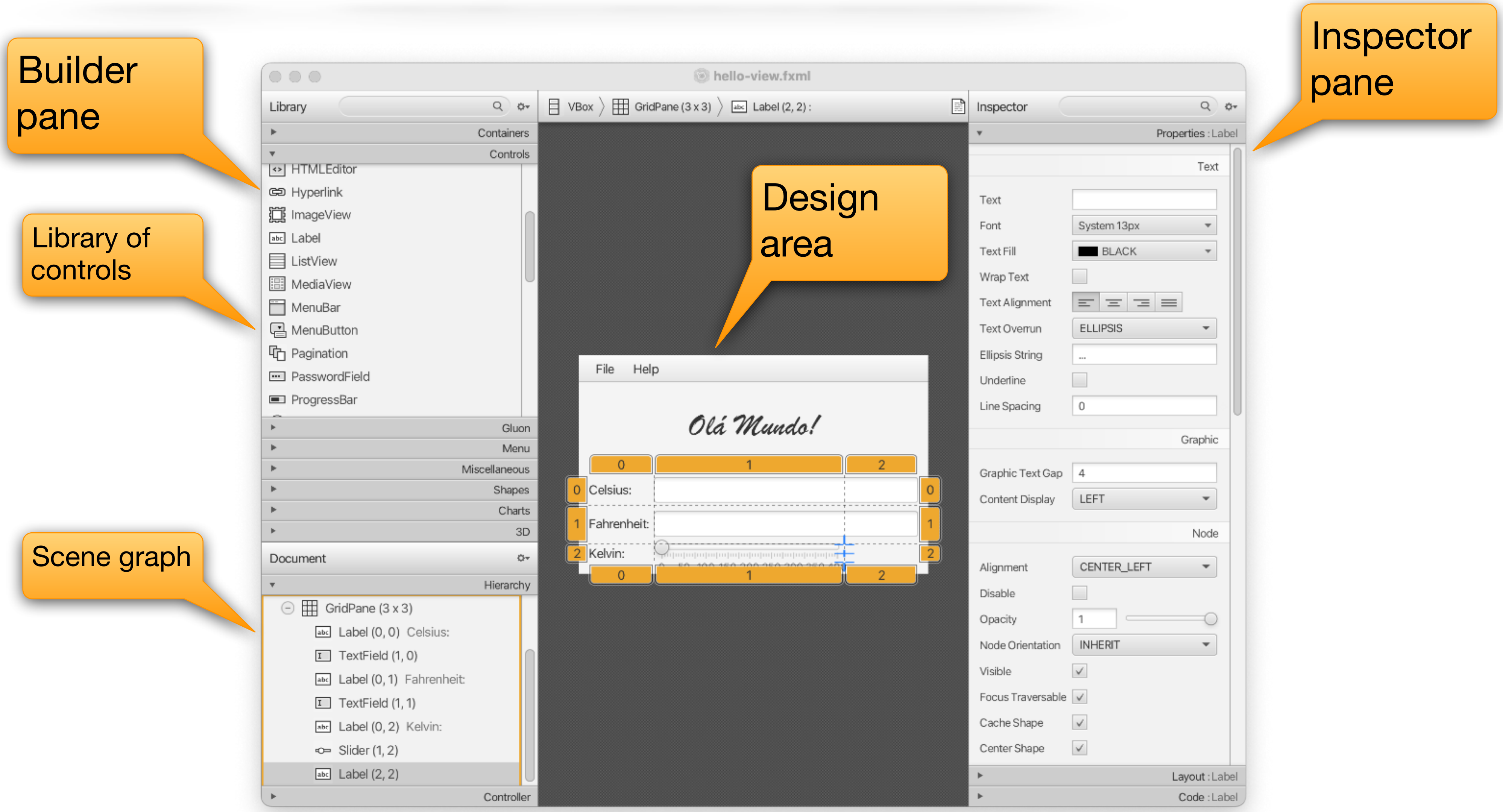
```java
package com.example.conversortemp;

import ...

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml")
         );
        Scene scene = new Scene(fxmlLoader.load(), 320, 240);
        stage.setTitle("Hello!");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) { launch(); }
}
```

conversor-temp > src > main > java > com > example > conversortemp > HelloApplication.java
2:1 LF UTF-8 4 spaces

View — JavaFX Scene Builder

- A standalone JavaFX GUI visual layout tool
    - GUI creation by drag&drop of GUI components onto a design area
- Generates FXML (FX Markup Language)
    - An XML vocabulary for defining and arranging JavaFX GUI controls declaratively
- Gluon Scene Builder
    - Scene Builder - Gluon (gluonhq.com)

View — JavaFX Scene Builder



Text

Text	
Font	System 13px
Text Fill	■ BLACK
Wrap Text	☐
Text Alignment	
Text Overrun	ELLIPSIS
Ellipsis String	...
Underline	☐
Line Spacing	0

Graphic

Graphic Text Gap	4
Content Display	LEFT

Node

Alignment	CENTER_LEFT
Disable	☐
Opacity	1
Node Orientation	INHERIT
Visible	✓
Focus Traversable	✓
Cache Shape	✓
Center Shape	✓

View — FXML

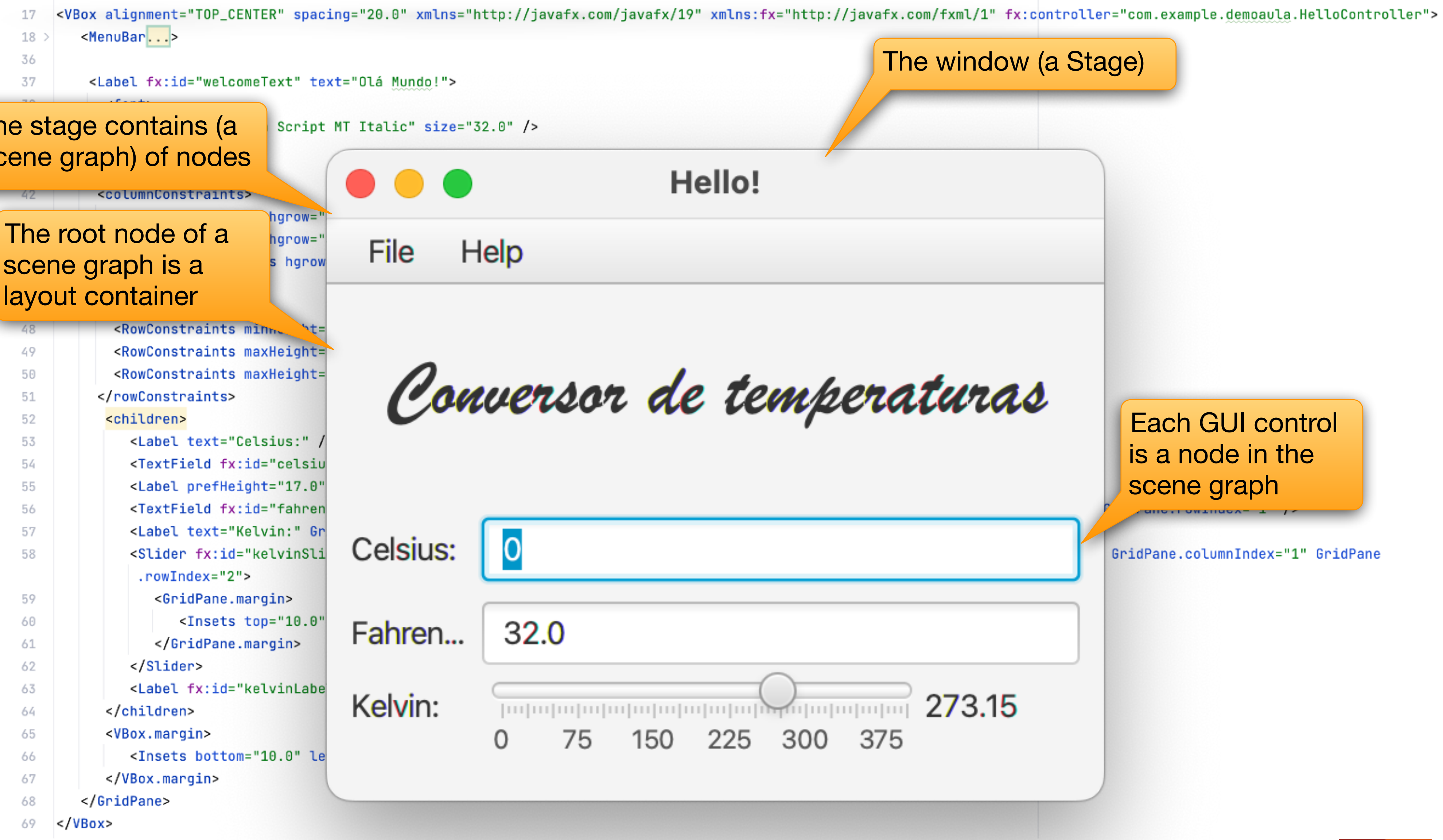
- FXML code is separate from program logic (Java source code)
- Makes it easier to create, debug, modify and maintain JavaFX GUI apps

- Programming the layout imperatively can be tedious
    - Doing it declaratively is easier
    - Scene builder further simplifies the process
    - Layout can be adjusted without having to compile (unless changes are needed in the Controller)

View — Anatomy of a JavaFX window

- An app window is know as **Stage**
    - an instance of class `javafx.stage.Stage`
- A Stage contains one active **Scene**
    - an instance of class `javafx.scene.Scene`
    - defines the GUI as a scene graph
- A **Scene Graph** is a tree data structure of nodes (`javafx.scene.Node`)
    - Nodes with children are **layout containers**
    - Leaf nodes are visual elements (GUI controls, shapes, images, video, etc.)

View — Anatomy of a Window



```
<VBox alignment="TOP_CENTER" spacing="20.0" xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.demoaula.HelloController">
  <MenuBar...>

   <Label fx:id="welcomeText" text="Olá Mundo!">
                              Script MT Italic" size="32.0" />

     <columnConstraints>
                                   hgrow="
                                   hgrow="
                                 s hgrow
      <RowConstraints min
      <RowConstraints maxHeight=
      <RowConstraints maxHeight=
     </rowConstraints>
      <children>
        <Label text="Celsius:" /
        <TextField fx:id="celsiu
        <Label prefHeight="17.0"
        <TextField fx:id="fahren
        <Label text="Kelvin:" Gr
        <Slider fx:id="kelvinSli                                                          GridPane.columnIndex="1" GridPane
         .rowIndex="2">
           <GridPane.margin>
              <Insets top="10.0"
           </GridPane.margin>
        </Slider>
        <Label fx:id="kelvinLabe
      </children>
      <VBox.margin>
        <Insets bottom="10.0" le
      </VBox.margin>
   </GridPane>
</VBox>
```

The window (a Stage)

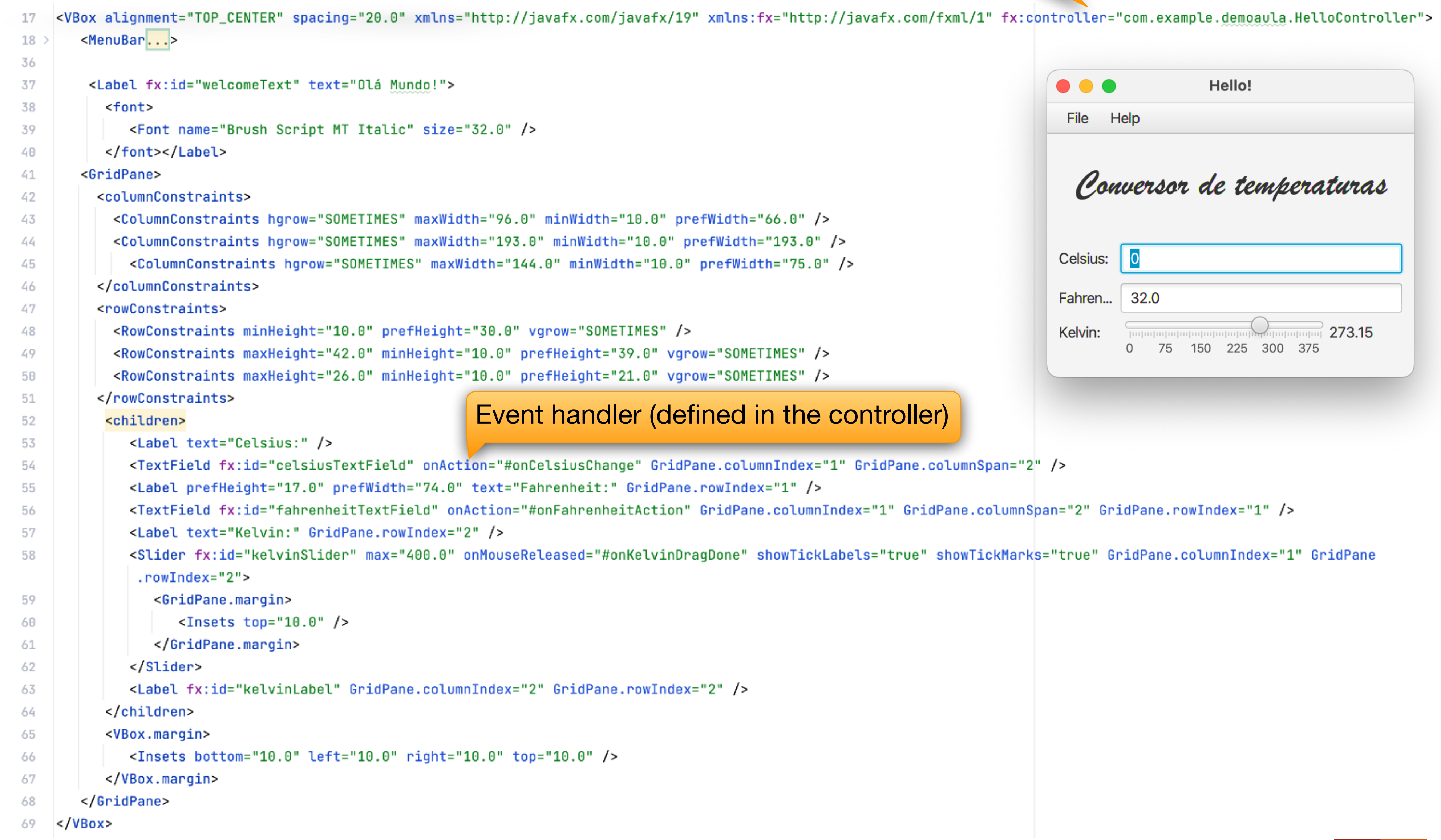
The stage contains (a Scene graph) of nodes

The root node of a scene graph is a layout container

Each GUI control is a node in the scene graph

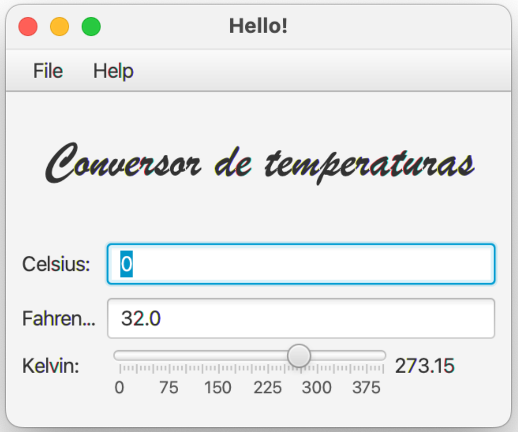
View — FXML

The controller



```xml
<VBox alignment="TOP_CENTER" spacing="20.0" xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.demoaula.HelloController">
   <MenuBar...>

    <Label fx:id="welcomeText" text="Olá Mundo!">
      <font>
         <Font name="Brush Script MT Italic" size="32.0" />
      </font></Label>
   <GridPane>
     <columnConstraints>
       <ColumnConstraints hgrow="SOMETIMES" maxWidth="96.0" minWidth="10.0" prefWidth="66.0" />
       <ColumnConstraints hgrow="SOMETIMES" maxWidth="193.0" minWidth="10.0" prefWidth="193.0" />
         <ColumnConstraints hgrow="SOMETIMES" maxWidth="144.0" minWidth="10.0" prefWidth="75.0" />
     </columnConstraints>
     <rowConstraints>
       <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
       <RowConstraints maxHeight="42.0" minHeight="10.0" prefHeight="39.0" vgrow="SOMETIMES" />
       <RowConstraints maxHeight="26.0" minHeight="10.0" prefHeight="21.0" vgrow="SOMETIMES" />
     </rowConstraints>
      <children>
         <Label text="Celsius:" />
         <TextField fx:id="celsiusTextField" onAction="#onCelsiusChange" GridPane.columnIndex="1" GridPane.columnSpan="2" />
         <Label prefHeight="17.0" prefWidth="74.0" text="Fahrenheit:" GridPane.rowIndex="1" />
         <TextField fx:id="fahrenheitTextField" onAction="#onFahrenheitAction" GridPane.columnIndex="1" GridPane.columnSpan="2" GridPane.rowIndex="1" />
         <Label text="Kelvin:" GridPane.rowIndex="2" />
         <Slider fx:id="kelvinSlider" max="400.0" onMouseReleased="#onKelvinDragDone" showTickLabels="true" showTickMarks="true" GridPane.columnIndex="1" GridPane
          .rowIndex="2">
            <GridPane.margin>
               <Insets top="10.0" />
            </GridPane.margin>
         </Slider>
         <Label fx:id="kelvinLabel" GridPane.columnIndex="2" GridPane.rowIndex="2" />
      </children>
      <VBox.margin>
         <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
      </VBox.margin>
   </GridPane>
</VBox>
```

Event handler (defined in the controller)

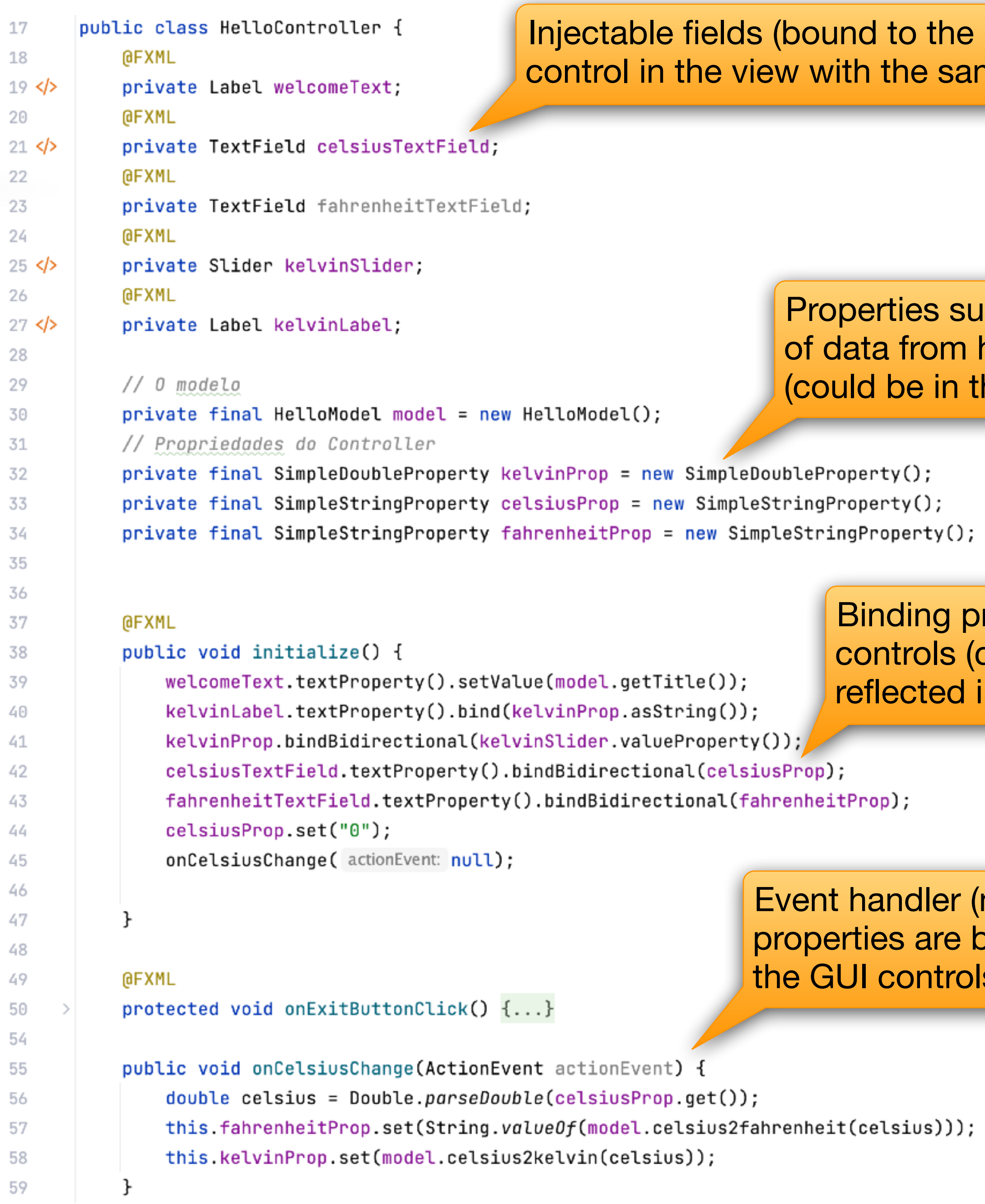


JavaFX controls and events

- **Controls** are GUI components, such as
    - Labels that display text,
    - TextFields that enable a program to receive user input,
    - Buttons that users click to initiate actions, etc.

- When the user interacts with a control, the control generates an **event**
    - The program can respond to this event through an **event handler**
    - The event handler defines what should happen when that specific user interaction occurs
    - Event handlers are defined in the Controller

Controller



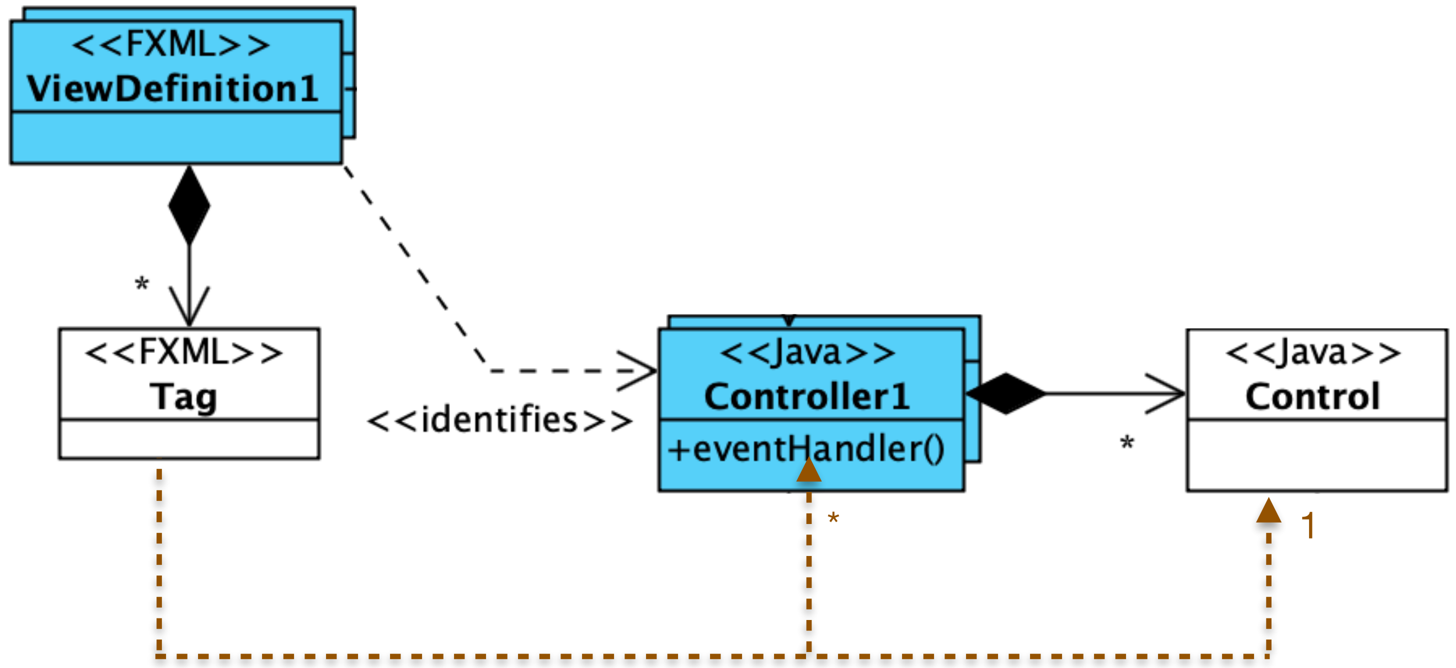
```java
public class HelloController {
    @FXML
    private Label welcomeText;
    @FXML
    private TextField celsiusTextField;
    @FXML
    private TextField fahrenheitTextField;
    @FXML
    private Slider kelvinSlider;
    @FXML
    private Label kelvinLabel;

    // O modelo
    private final HelloModel model = new HelloModel();
    // Propriedades do Controller
    private final SimpleDoubleProperty kelvinProp = new SimpleDoubleProperty();
    private final SimpleStringProperty celsiusProp = new SimpleStringProperty();
    private final SimpleStringProperty fahrenheitProp = new SimpleStringProperty();


    @FXML
    public void initialize() {
        welcomeText.textProperty().setValue(model.getTitle());
        kelvinLabel.textProperty().bind(kelvinProp.asString());
        kelvinProp.bindBidirectional(kelvinSlider.valueProperty());
        celsiusTextField.textProperty().bindBidirectional(celsiusProp);
        fahrenheitTextField.textProperty().bindBidirectional(fahrenheitProp);
        celsiusProp.set("0");
        onCelsiusChange( actionEvent: null);

    }

    @FXML
    protected void onExitButtonClick() {...}

    public void onCelsiusChange(ActionEvent actionEvent) {
        double celsius = Double.parseDouble(celsiusProp.get());
        this.fahrenheitProp.set(String.valueOf(model.celsius2fahrenheit(celsius)));
        this.kelvinProp.set(model.celsius2kelvin(celsius));
    }
```

Injectable fields (bound to the control in the view with the same ID)

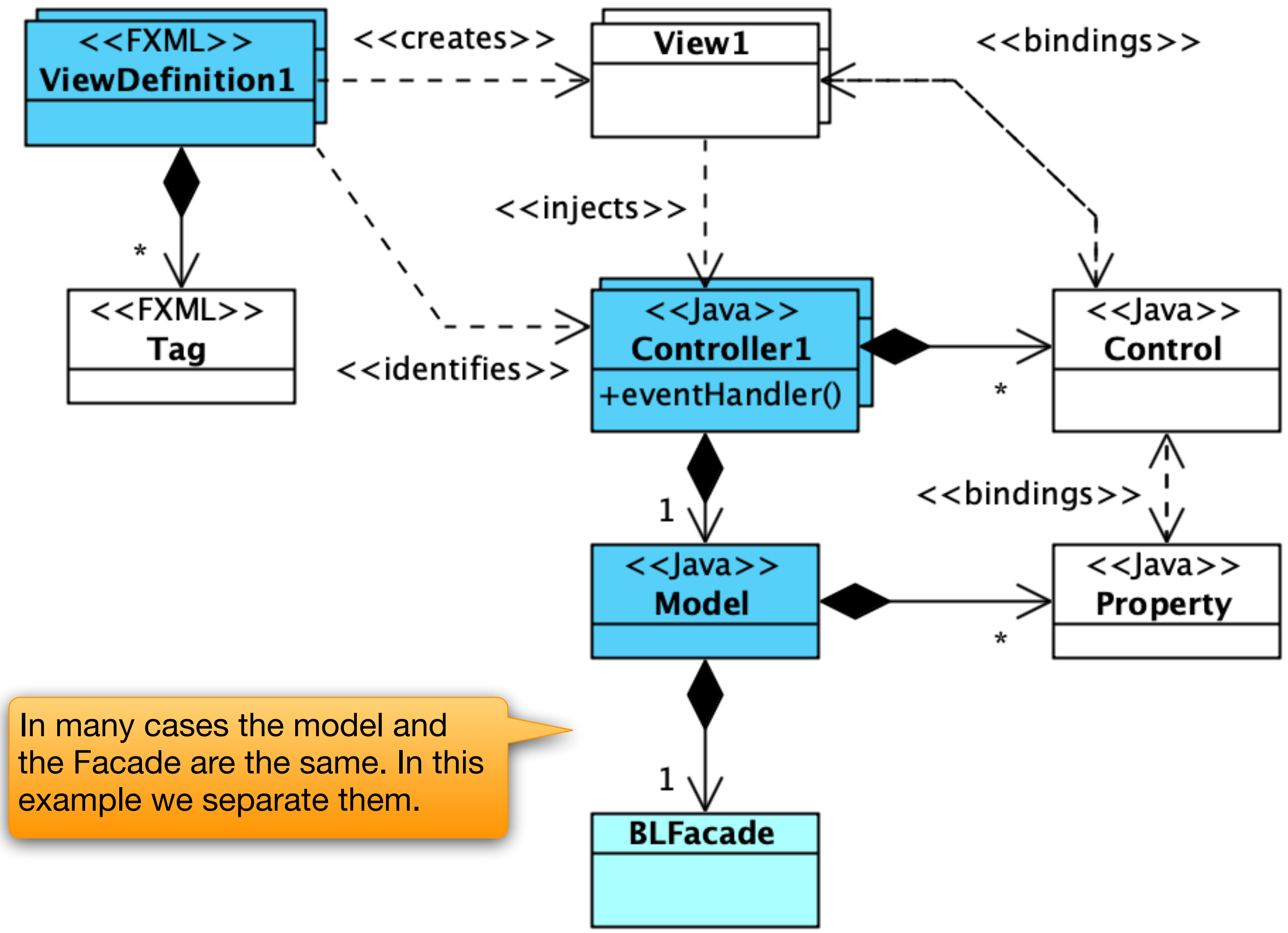
Properties support separation of data from how it is displayed (could be in the model)

Binding properties to view GUI controls (changes in one are reflected in the other)

Event handler (note that the properties are being used, not the GUI controls)



The IDs and action attributes on the FXML tags are mapped to methods and instance variables on the Controller



In many cases the model and the Facade are the same. In this example we separate them.