

---

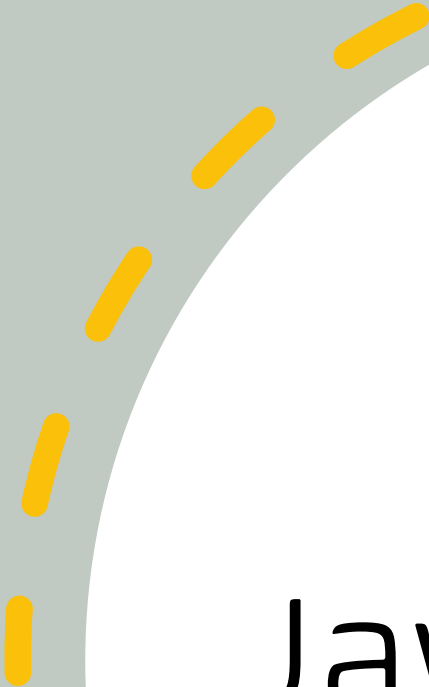
# CS 213 SOFTWARE METHODOLOGY

Lily Chang

CS Department @ Rutgers New Brunswick

FALL 2023





# JavaFX Basics

Lecture Note #8



# What is JavaFX?

- JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms
- JavaFX library is written as a Java API, JavaFX application code can reference APIs from any Java library: <https://openjfx.io/javadoc/21/>
- Download:  
<https://gluonhq.com/products/javafx/>
- Getting started with JavaFX:  
<https://openjfx.io/openjfx-docs/>



# JavaFX Applications

---

- The look and feel of JavaFX applications can be customized
- Cascading Style Sheets (CSS) **separate appearance and style from implementation** so that developers can concentrate on coding
- Graphic designers can easily customize the appearance and style of the application through the CSS
- Develop the presentation aspects of the UI in the FXML scripting language and use Java code for the application logic
- To design UIs without writing code, then use **JavaFX Scene Builder**.
- Scene Builder creates **FXML markup** that can be ported to an Integrated Development Environment (IDE) so that developers can add the business logic
- Download: <https://gluonhq.com/products/scene-builder/>
- Configuring the Scene Builder in IntelliJ:  
<https://www.jetbrains.com/help/idea/opening-fxml-files-in-javafx-scene-builder.html>



# JavaFX Key Features

- **Java APIs.**
  - A Java library that consists of classes and interfaces that are written in Java code
- **FXML and Scene Builder**
  - FXML is an XML-based declarative markup language for constructing a JavaFX application user interface
- **WebView**
  - A web component that uses WebKitHTML technology to make it possible to embed web pages within a JavaFX application
  - **JavaScript running in WebView can call Java APIs**, and Java APIs can call JavaScript running in WebView  
[Adding HTML Content to JavaFX Applications.](#)



# JavaFX Key Features

## Swing interoperability

- Existing Swing applications can be updated with JavaFX features, such as rich graphics media playback and embedded Web content. The `SwingNode` class, which enables you to embed Swing content into JavaFX applications

## Built-in UI controls and CSS

- Provides all the major UI controls that are required to develop a full-featured application

## 3D Graphics Features

- API classes for `Shape3D` (`Box`, `Cylinder`, `MeshView`, and `Sphere` subclasses), `SubScene`, `Material`, `PickResult`, `LightBase` (`AmbientLight` and `PointLight` subclasses), and `SceneAntialiasing`

# JavaFX Key Features



## Canvas API

Enables drawing directly within an area of the JavaFX scene that consists of one graphical element (node).



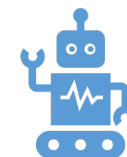
## Printing API

The `javafx.print` package provides the public classes for the [JavaFX Printing API](#).



## Rich Text Support

Enhanced text support to JavaFX, including bi-directional text and complex text scripts, and multi-line, multi-style text in text nodes



## Multitouch Support for handheld devices

Provides support for multitouch operations, based on the capabilities of the underlying platform.

# What can we build with JavaFX?

You can build many types of applications

Typically, they are network-aware applications that are deployed across multiple platforms and display information in a high-performance modern user interface that features audio, video, graphics, and animation



# JavaFX vs. Swing and AWT

When Java was introduced, the GUI classes were bundled in a library known as the **Abstract Windows Toolkit (AWT)**, which is prone to platform-specific bugs

**Swing** replaced the AWT user-interface components where components are painted directly on canvases using Java code; it is designed for developing **desktop GUI** applications

**JavaFX** replaced Swing and is a newer GUI platform that incorporates modern GUI technologies to enable you to develop rich GUI applications

Provides a multitouch support for **touch-enabled devices** such as tablets and smart phones, 2D, 3D, animation, and video and audio playback

Oracle no longer supports JavaFX, which is now OpenFX:  
<https://openjfx.io/>

# Working on JavaFX apps with IntelliJ

If you don't use Maven or Gradle, you can

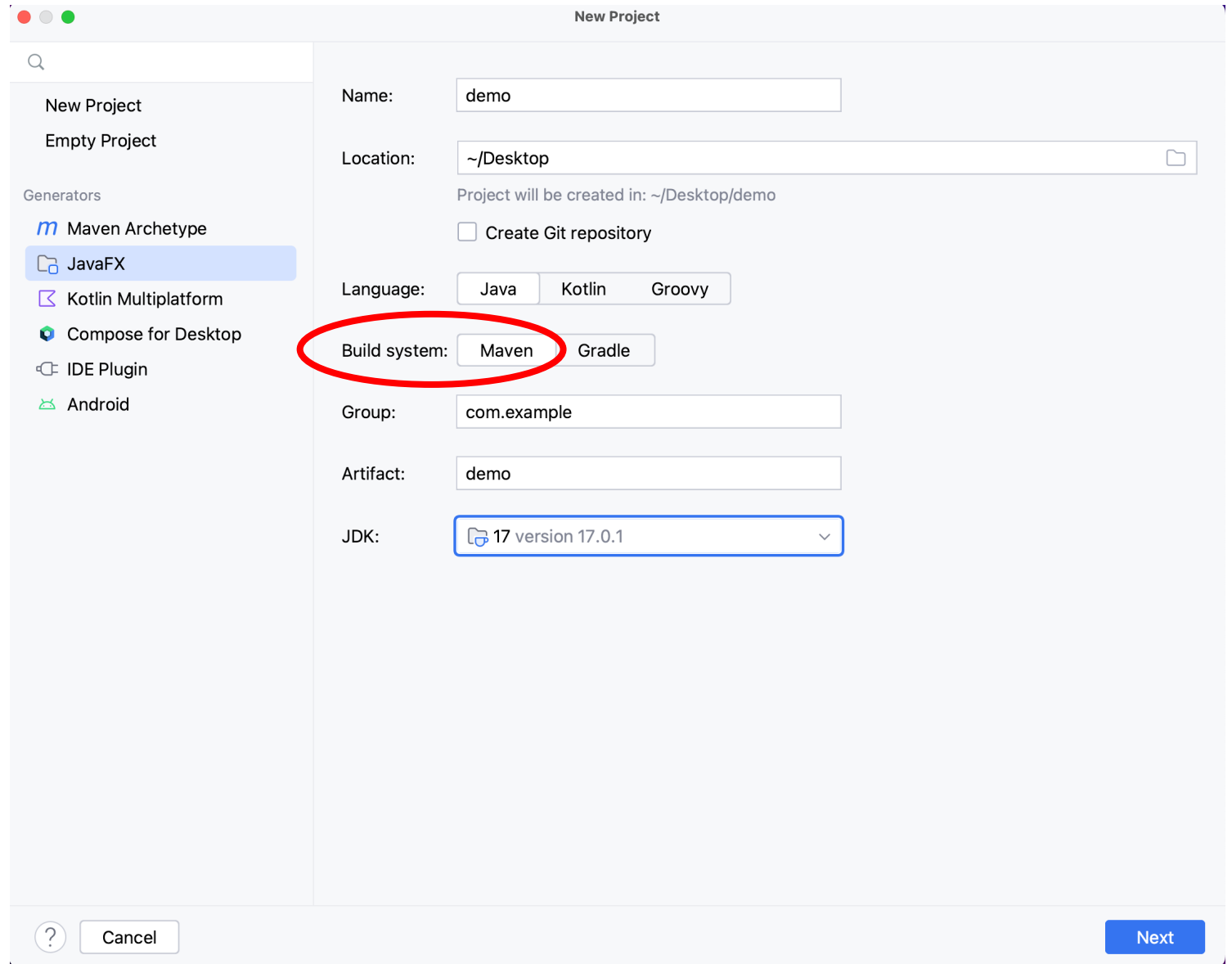
Download JavaFX APIs - <https://gluonhq.com/products/javafx>

- Need to edit Run Configuration
- Add JavaFX .jar to the classpath
- Add VM Options (<https://openjfx.io/openjfx-docs/>)

Download SceneBuilder to help you with GUI design

- <https://gluonhq.com/products/scene-builder/#download>
- Preferences->Languages and Frameworks->JavaFX->Path to SceneBuilder

# INTELLIJ – CREATE A JAVAFX PROJECT WITH MAVEN



The image shows the 'New Project' dialog in IntelliJ IDEA. On the left, under 'Generators', 'JavaFX' is selected. On the right, the 'Build system' is set to 'Maven', which is circled in red. Other fields include 'Name' (demo), 'Location' (~/Desktop), 'Language' (Java), 'Group' (com.example), 'Artifact' (demo), and 'JDK' (17 version 17.0.1). A 'Next' button is at the bottom right.

New Project

Empty Project

Generators

- Maven Archetype
- JavaFX**
- Kotlin Multiplatform
- Compose for Desktop
- IDE Plugin
- Android

Name: demo

Location: ~/Desktop  
Project will be created in: ~/Desktop/demo

☐ Create Git repository

Language: Java Kotlin Groovy

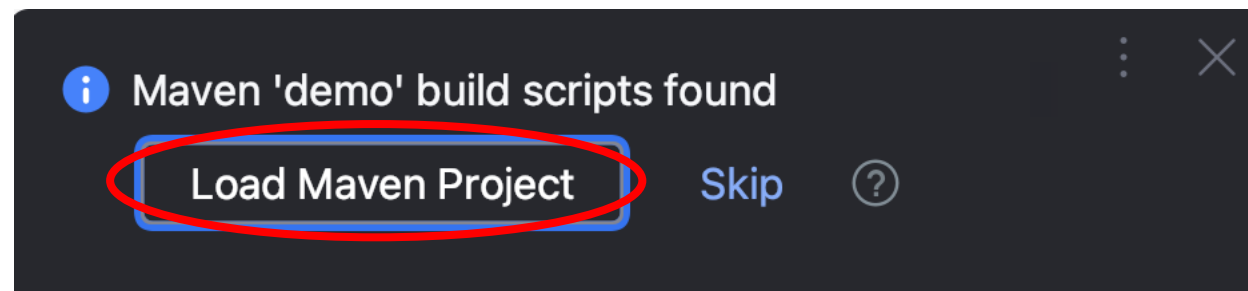
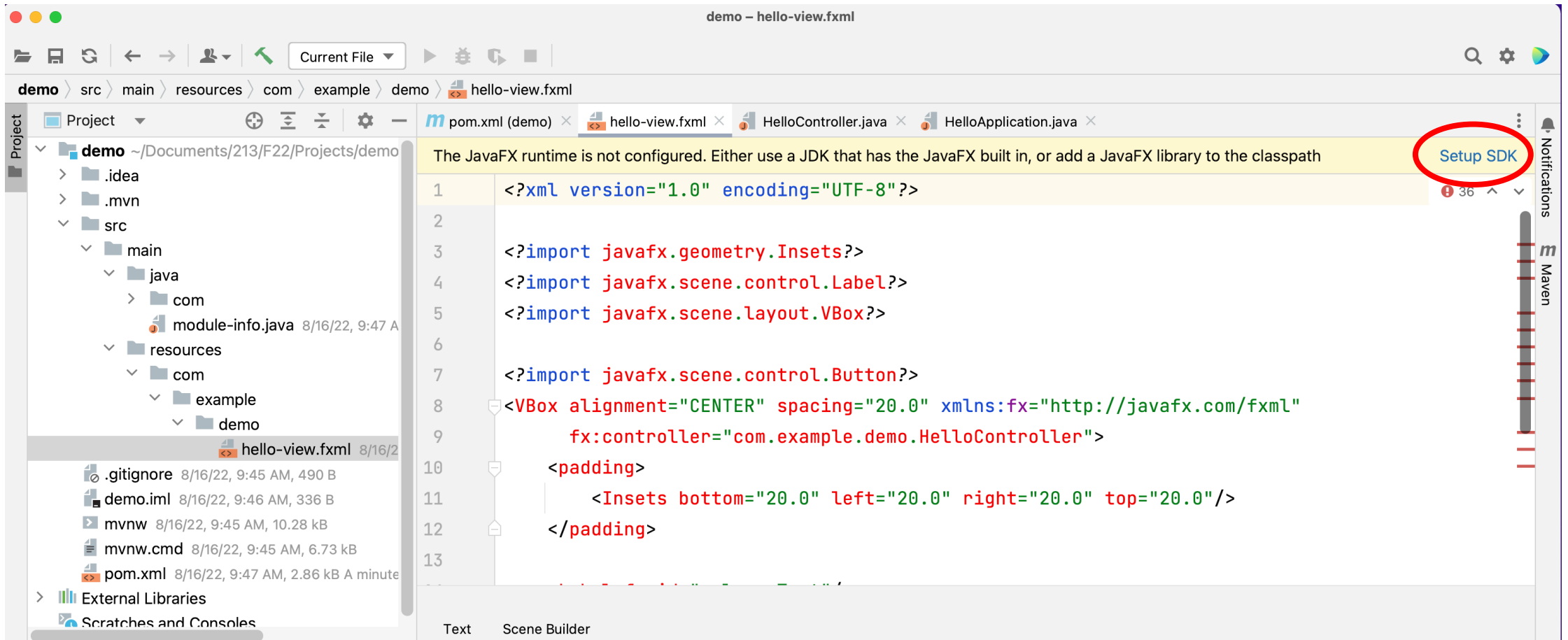
**Build system: Maven** Gradle

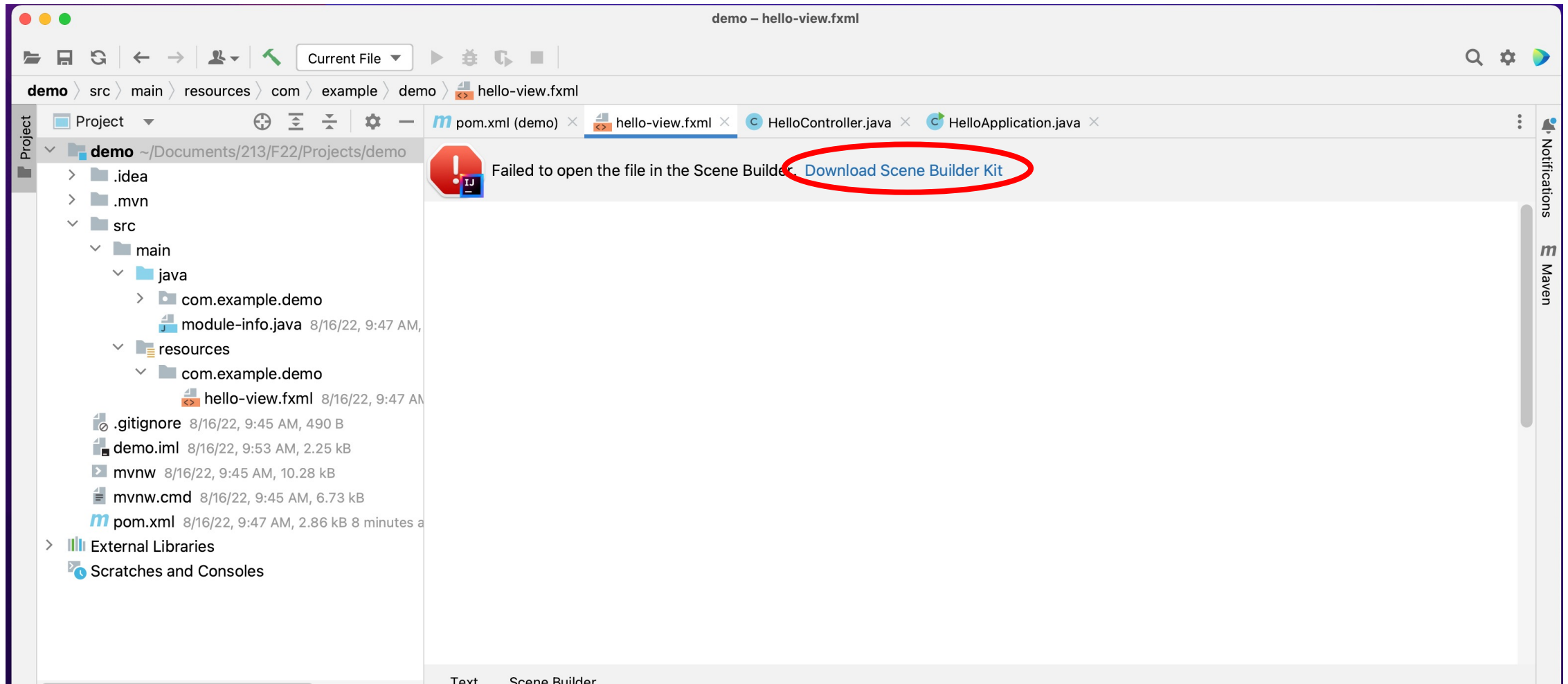
Group: com.example

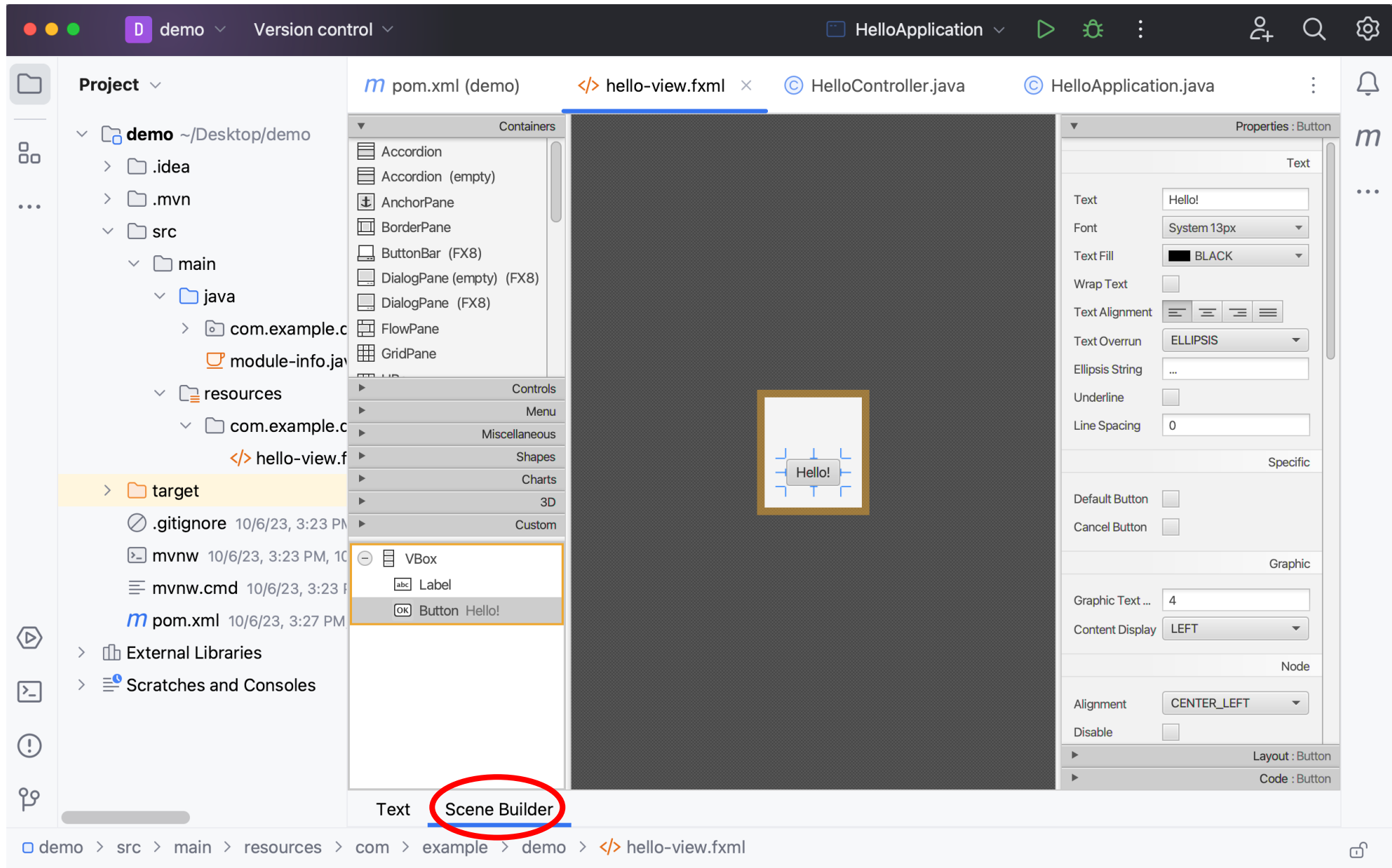
Artifact: demo

JDK: 17 version 17.0.1

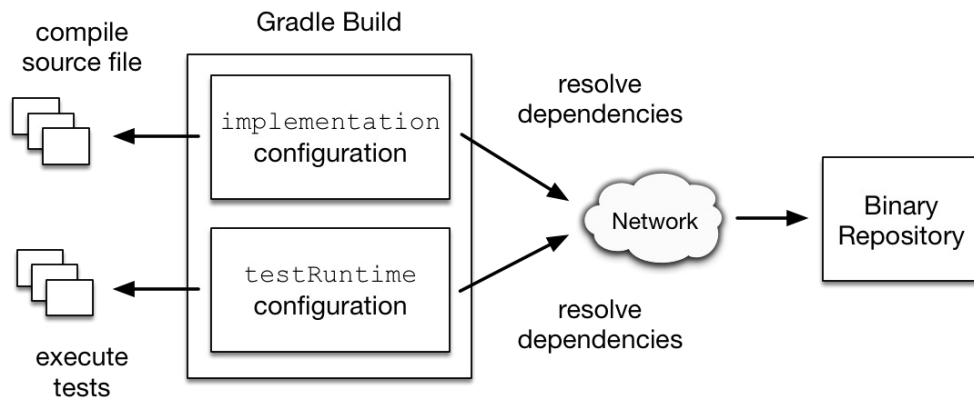
Cancel Next







# Build Systems – Gradle and Maven



- What is a build system?
  - software project management
  - manage dependency configuration
  - automate the process of generating a software build for delivery
  - with the build system, you don't need to add the JavaFX library classes (the jar files) to the classpath