**README — Importing the Eclipse Project (JavaFX) without Maven/Gradle**

This guide shows how to import the CountingGame Eclipse project and what you STILL NEED TO do so it runs reliably with JavaFX.

Target setup used in examples: JDK C:\Java\Oracle\_JDK-24, JavaFX SDK C:\javafx-sdk-24.0.2

1) Import the existing Eclipse project

File → Import… → General → Existing Projects into Workspace → Next

• Select root directory: choose the folder that contains the project’s .project file

• Check the project → Finish

Import completes, but you STILL NEED TO do the steps below.

2) STILL NEED TO — Point Eclipse at your JDK 24

• Window → Preferences → Java → Installed JREs → Add… → Standard VM

• JRE home: C:\Java\Oracle\_JDK-24

• Name it Oracle\_JDK-24 → Finish; check it as the default → Apply and Close

3) STILL NEED TO — Put JavaFX on the Modulepath (compile‑time)

• Project → Properties → Java Build Path → Libraries

• Add Library… → User Library → Next → User Libraries… → New…

• Name: JavaFX24\_24.0.2; Add External JARs… → select all JARs in C:\javafx-sdk-24.0.2\lib\; OK → Close

• Back in Add Library…, choose JavaFX24\_24.0.2 → Finish

• Confirm it appears under Modulepath (not Classpath). If on Classpath, remove and re‑add.

4) STILL NEED TO — Create the Run Configuration (runtime flags)

• Run → Run Configurations… → Java Application → New launch configuration

• Project: CountingGame

• Main class: countinggame.CountingGame (use Search… if needed)

Arguments → VM arguments (single line; no duplicates):

--module-path "C:\javafx-sdk-24.0.2\lib" --add-modules javafx.controls,javafx.graphics,javafx.media --enable-native-access=javafx.graphics

Use either --add-modules X,Y or --add-modules=X,Y. Do not include both forms. Do not duplicate --module-path. No trailing dot (.).

5) STILL NEED TO — Make sure resources are on the classpath

The code searches these classpath locations (first match wins):

/countinggame/resources/  
/countinggame/  
/resources/

Preferred layout (easiest to get right):

CountingGame/  
 src/  
 countinggame/  
 CountingGame.java  
 resources/  
 bluesparklesbackground.png  
 1.png … 10.png  
 1.mp3 … 10.mp3

Alternative layout: create a top-level resources/ folder → Right-click → Build Path → Use as Source Folder → put files inside.

After placing files: Refresh (F5) the project → Project → Clean…

6) STILL NEED TO — Verify package and names

First line of CountingGame.java must be exactly:

package countinggame;

File name must be CountingGame.java (matches public class CountingGame).

7) STILL NEED TO — Ensure Eclipse builds to bin/

• Project → Build Automatically (enable)

• Project → Clean… → select CountingGame → Clean

Check that bin/ appears with compiled classes:

\  
CountingGame/  
├─ bin/  
│ └─ countinggame/  
│ ├─ CountingGame.class  
│ └─ resources/  
│ ├─ bluesparklesbackground.png  
│ ├─ 1.png … 10.png  
│ └─ 1.mp3 … 10.mp3  
└─ src/  
 └─ countinggame/  
 ├─ CountingGame.java  
 └─ resources/  
 ├─ bluesparklesbackground.png  
 ├─ 1.png … 10.png  
 └─ 1.mp3 … 10.mp3

If bin/ is missing: Project → Properties → Java Build Path → Source → verify Default output folder is CountingGame\bin.

You can keep the VM arguments even with JPMS; it’s simple and works well in Eclipse.

**Common errors & quick fixes**

“import javafx … cannot be resolved” → JavaFX SDK jars weren’t added to the Modulepath (Step 3). Clean the project.

“Could not find or load main class countinggame.CountingGame” → Fix package line or main class in Run Config; Clean.

Images/Sounds not found → Put assets under src/countinggame/resources (preferred), Refresh, Clean; check exact filenames.

EGit “HOME not set” → optional: set HOME=C:\Users\<YourUser>.

Renderer “Unsafe” warnings → harmless for this app.

VM arguments (copy‑paste)

--module-path "C:\javafx-sdk-24.0.2\lib" --add-modules javafx.controls,javafx.graphics,javafx.media --enable-native-access=javafx.graphics