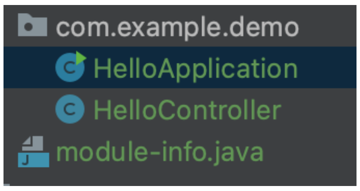
From IntelliJ JavaFX demo to JAR

Form intelliJ You can have



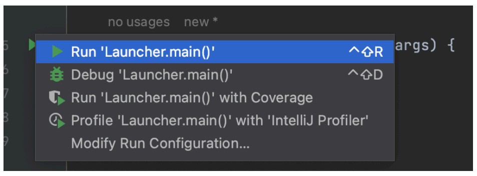
Steps:

1) add Launcher class

```
public class Launcher {

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

2) Pls check class if correct. (**HelloApplication**). We have to fake Java passing a std cli app that will trigger JavaFX.

3) Try run form IntelliJ from Launcher:


no usages   new *
Run 'Launcher.main()'   ^⇧R
Debug 'Launcher.main()'   ^⇧D
Run 'Launcher.main()' with Coverage
Profile 'Launcher.main()' with 'IntelliJ Profiler'
Modify Run Configuration...
args) {

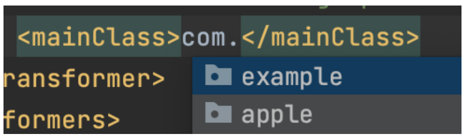
4)   Add support for shade plugin. n POM add: (in plug-in section...   <build> ---> <plugins>)

```
<!-- added -->

    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-shade-plugin</artifactId>
        <version>3.2.0</version>
        <executions>
            <execution>
                <phase>package</phase>
                <goals>
                    <goal>shade</goal>
                </goals>
                <configuration>
                    <shadedArtifactAttached>true</shadedArtifactAttached>
                    <shadedClassifierName>project-classifier</
shadedClassifierName>
                    <outputFile>shade\${project.artifactId}.jar</
outputFile>
                    <transformers>
                        <transformer implementation=

"org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                            <mainClass>com.ingconti.Launcher</mainClass>
                        </transformer>
                    </transformers>
                </configuration>
            </execution>
        </executions>
    </plugin>

    <!-- end of added -->
```

5) PLS change your class and or path.
Tip:
If in one:

<mainClass>com.example.demo.Launcher</mainClass>

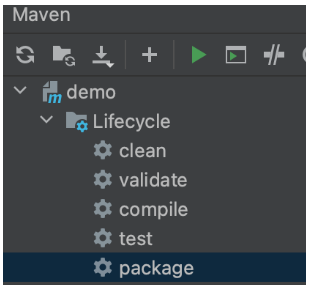
You remove ALL from "com" and start typing DOT, IntelliJ will suggest:



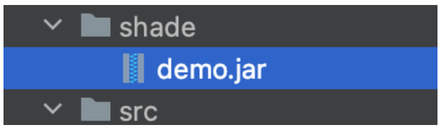
```
<mainClass>com.</mainClass>
ransformer>   example
formers>      apple
```

Go on selecting And typing dots!

6) On maven tab:



Click on package. You will see a new folder, shade: this will be You ready final JAR:



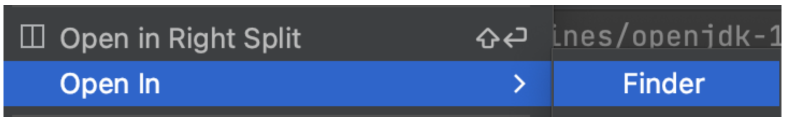
Right click to run.

Note:
If You got

```
WARNING: Unsupported JavaFX configuration: classes were loaded from
'unnamed module @3c5d1c1b'
```

IT'S OK!

7) from console
Open folder of JAR; on right click on shell:



(On windows will be open in Explorer..)

8) Open a std terminal / console (for windows shell... or better WSL2 console) and type **cd**
9) Drag folder in Your console (so You dont haver to type path...)

You will se something similar to :

```
cd /Users/ingconti/Downloads/samples-master/HelloFX/Maven/demo/shade/
```

11) hit return..

10) with **ls -la** or **dir** You should see:

```
-rw-r--r--@  1 ingconti  staff  8271988 Apr 11 12:11 demo.jar
```

Go on running:
  java -jar demo.jar

Et voilà!

PS it also run from double click.

12) Some cleanup

You can remove all there section: <executions>

It's ONLY to run inside intellJ: now You run from jar.

13) If You rename package or folder, pls FIX POM! (Refactor should work..)

14) **BUT I need console, too**!

It's already ok.. we do start from console!

See main:

```java
public class HelloApplication {

    public static void main(String[] args) {

        HelloApplication.main(args);

    }

}
```

main is ALREADY a console app. (DO not use Launcher, but modify HelloApplication, use of Launcher is only a. Dirty patch).

Suppose You want to run as server for example with some switch on run:

```
java -jar demo.jar --server
```

(PLS every time rebuild using package from MAVEN! We are using console!)

On main:

```
public static void main(String[] args) {
    for (String arg: args) {
        System.out.println(arg);
    }

    launch();
}
```

Run:

```
java -jar demo.jar --server
```

```
--server
Apr 11, 2023 2:19:04 PM com.sun.javafx.application.PlatformImpl startup
```

SO:

```
public static void main(String[] args) {
    // for (String arg: args) { System.out.println(arg);}
    if (args.length > 0) {
        String param0 = args[0];
        if (param0.equals( "--server") )
            runAsServer();
    }else {
        launch();
    }
}

static void runAsServer(){
    System.out.println("started as server.. bye!");
}
```

....

You will get:

```
java -jar JavaFXMinimalClickableDemo.jar --server

started as server.. bye!
```

Final POM:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/
xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <name>demo</name>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <junit.version>5.9.1</junit.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-controls</artifactId>
            <version>19</version>
        </dependency>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-fxml</artifactId>
            <version>19</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.10.1</version>
                <configuration>
                    <source>19</source>
                    <target>19</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.openjfx</groupId>
                <artifactId>javafx-maven-plugin</artifactId>
                <version>0.0.8</version>

            </plugin>


            <!-- added -->

            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>3.2.0</version>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
```

```
                            <shadedArtifactAttached>true</shadedArtifactAttached>
                            <shadedClassifierName>project-classifier</
shadedClassifierName>
                            <outputFile>shade\${project.artifactId}.jar</outputFile>
                            <transformers>
                                <transformer implementation=

"org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                                    <mainClass>com.example.demo.Launcher</mainClass>
                                </transformer>
                            </transformers>
                        </configuration>
                    </execution>
                </executions>
            </plugin>

            <!-- end of added -->

        </plugins>
    </build>
</project>
```