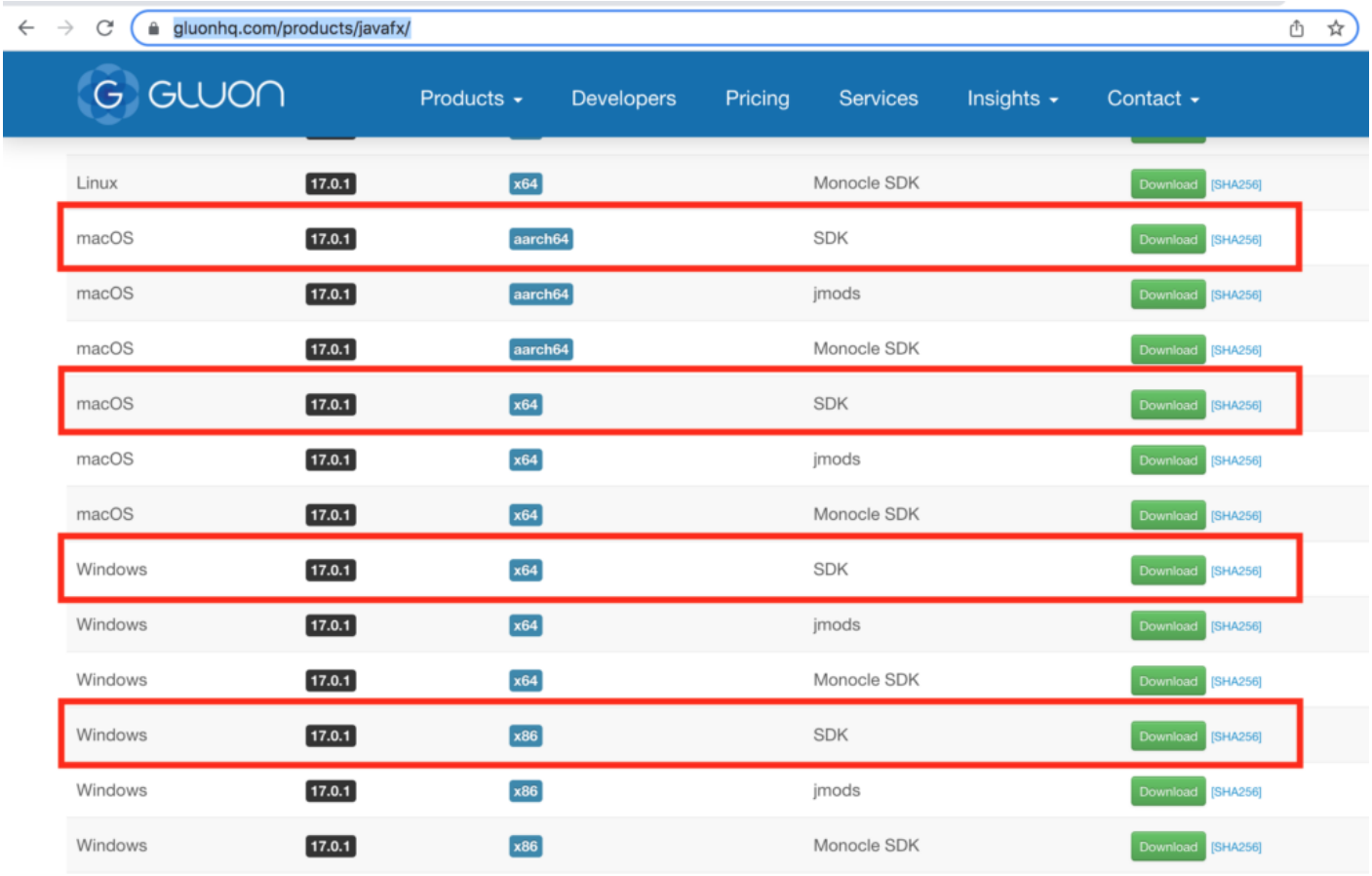
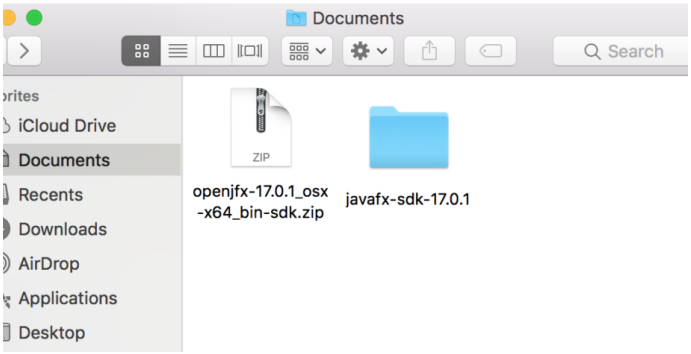
How to add JavaFX to Eclipse (the easy way)

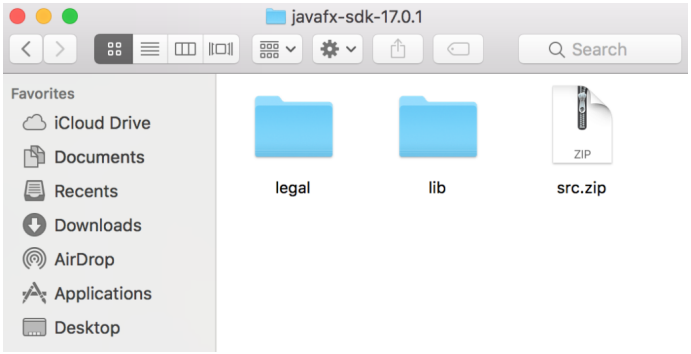
STEP 1. DOWNLOAD JAVAFX

Go to https://gluonhq.com/products/javafx/ and download the appropriate **SDK** for your operating system. Make sure you download the SDK and make sure you choose the correct operating system and architecture for your computer. More than likely, you'll be downloading one of the outlined options below.



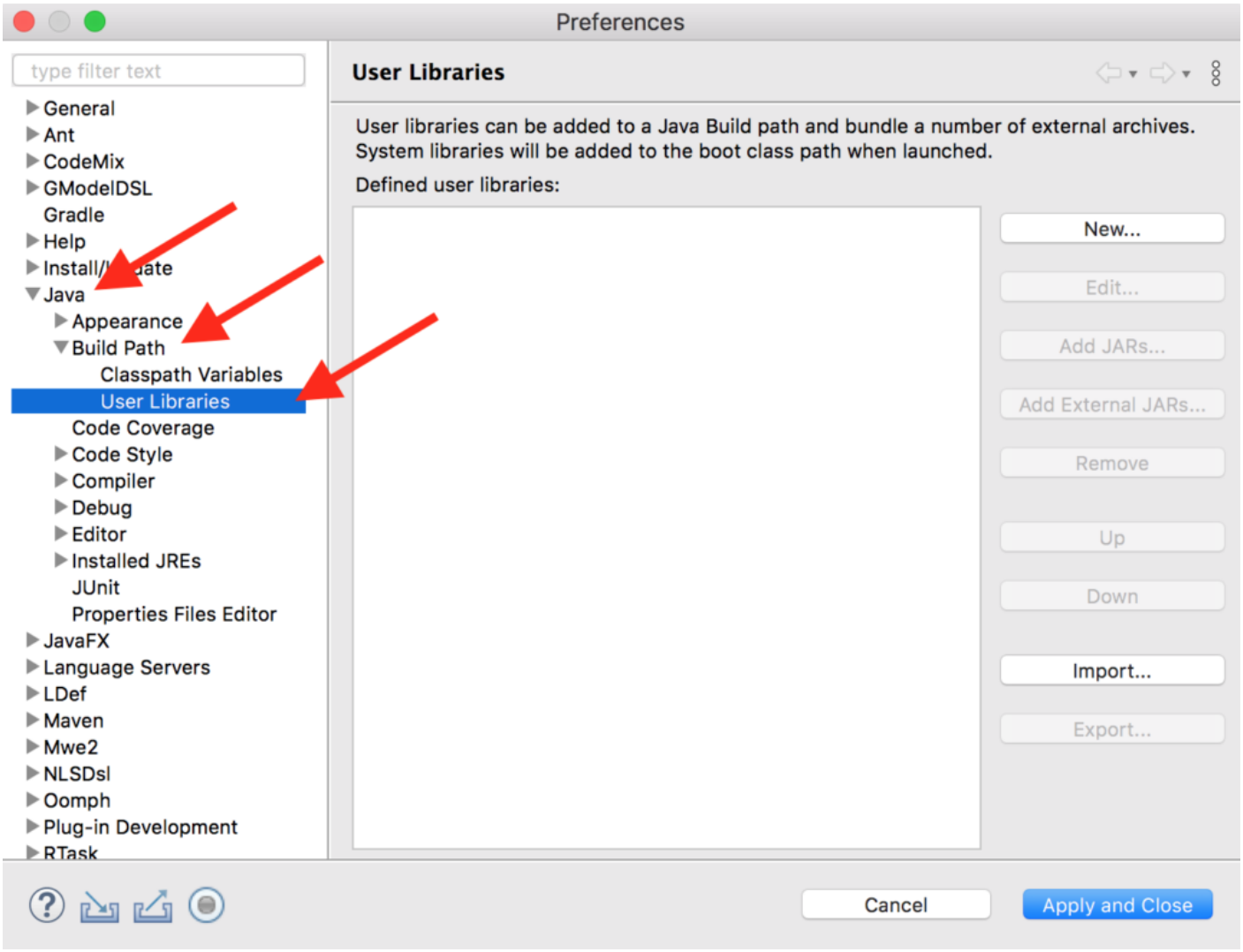
This will install a `.zip` file. Unzip this file and put the resulting folder in a memorable place, such as your `/Documents` folder. We'll need this later.



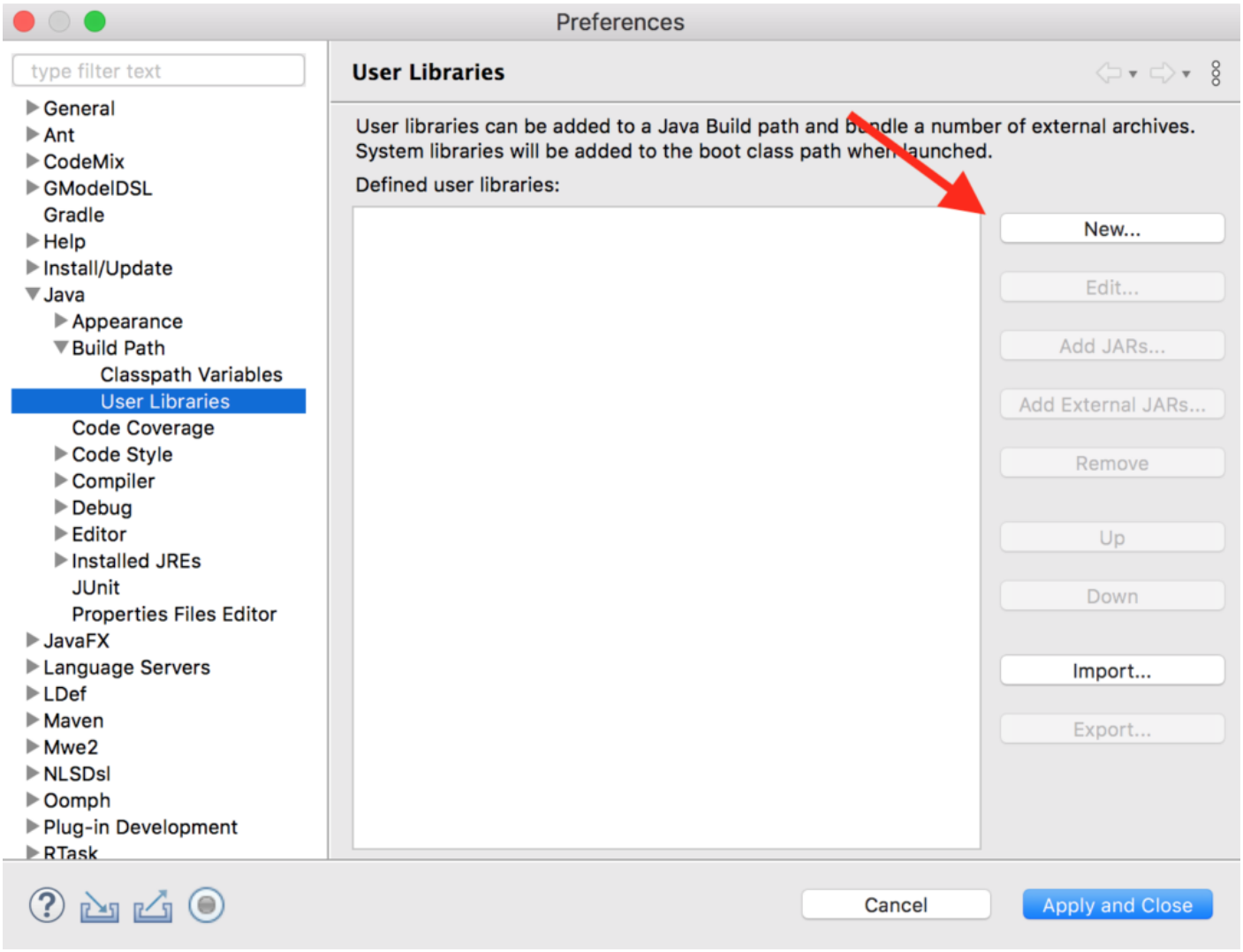


STEP 2. CREATE A USER LIBRARY IN ECLIPSE

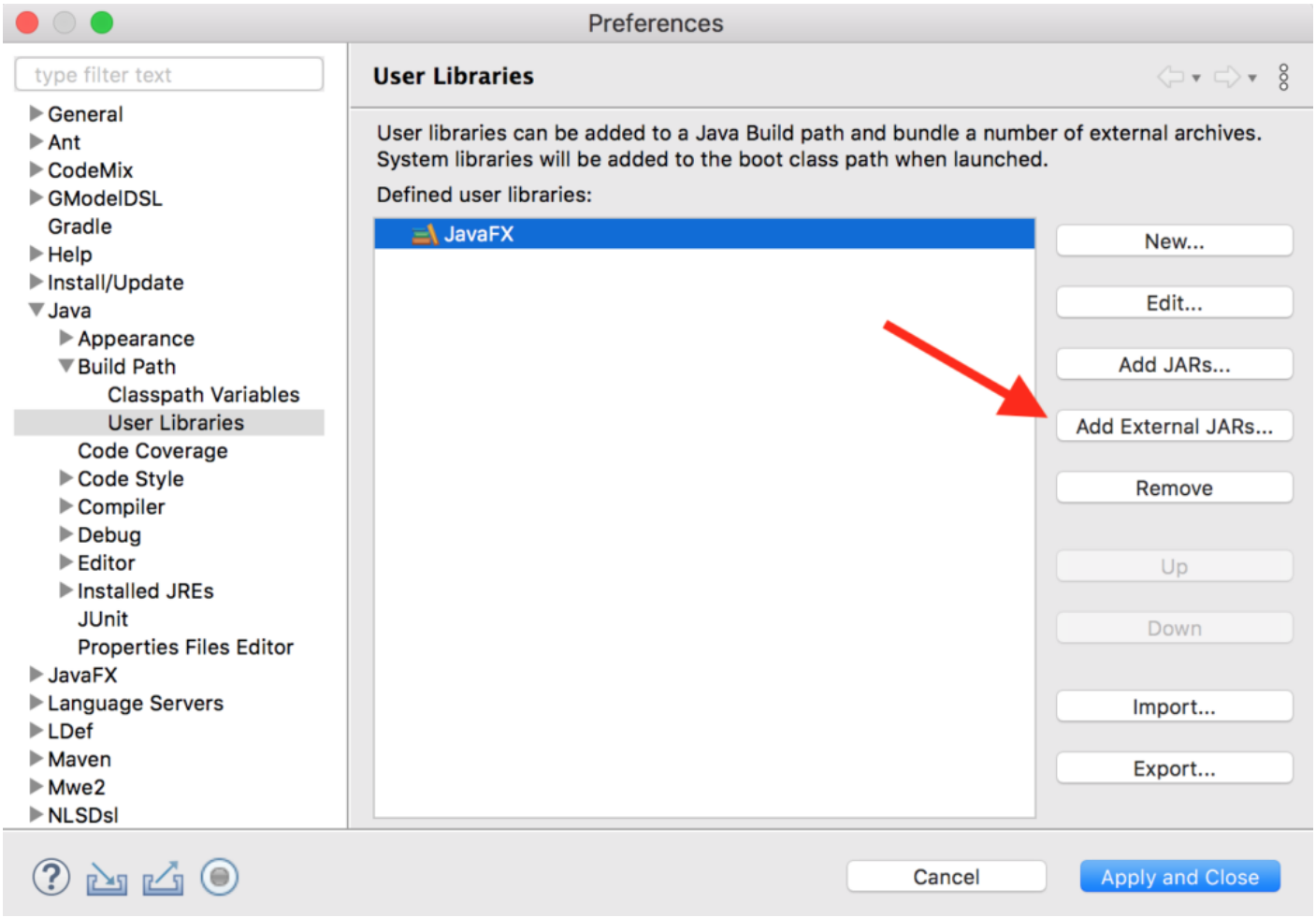
- Open up Eclipse and go to **Preferences**
- You should be able to get to Preferences from Eclipse > Preferences or Window > Preferences
- Go to **Java** > **Build Path** > **User Libraries**



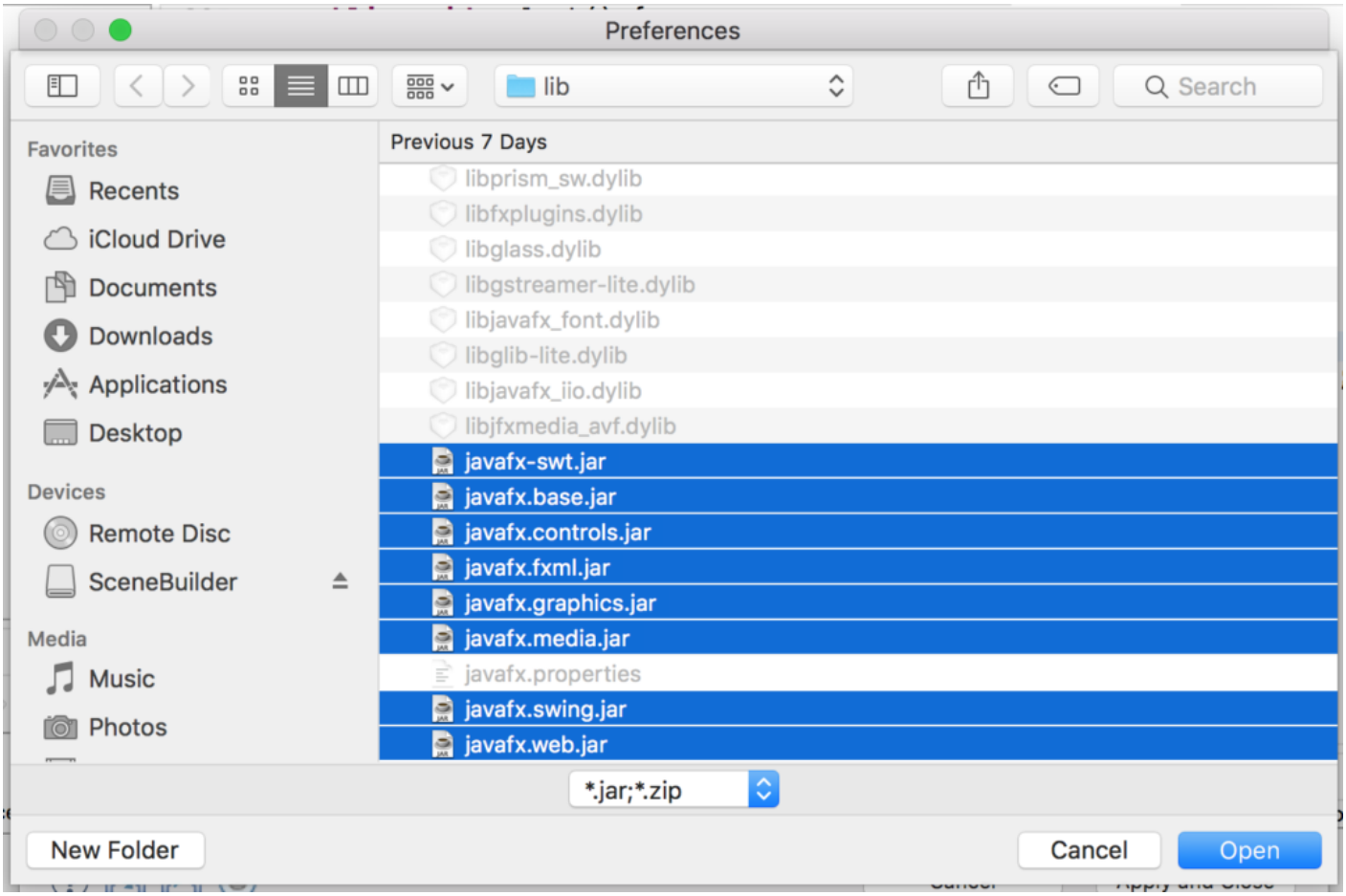
- Now click **New...**



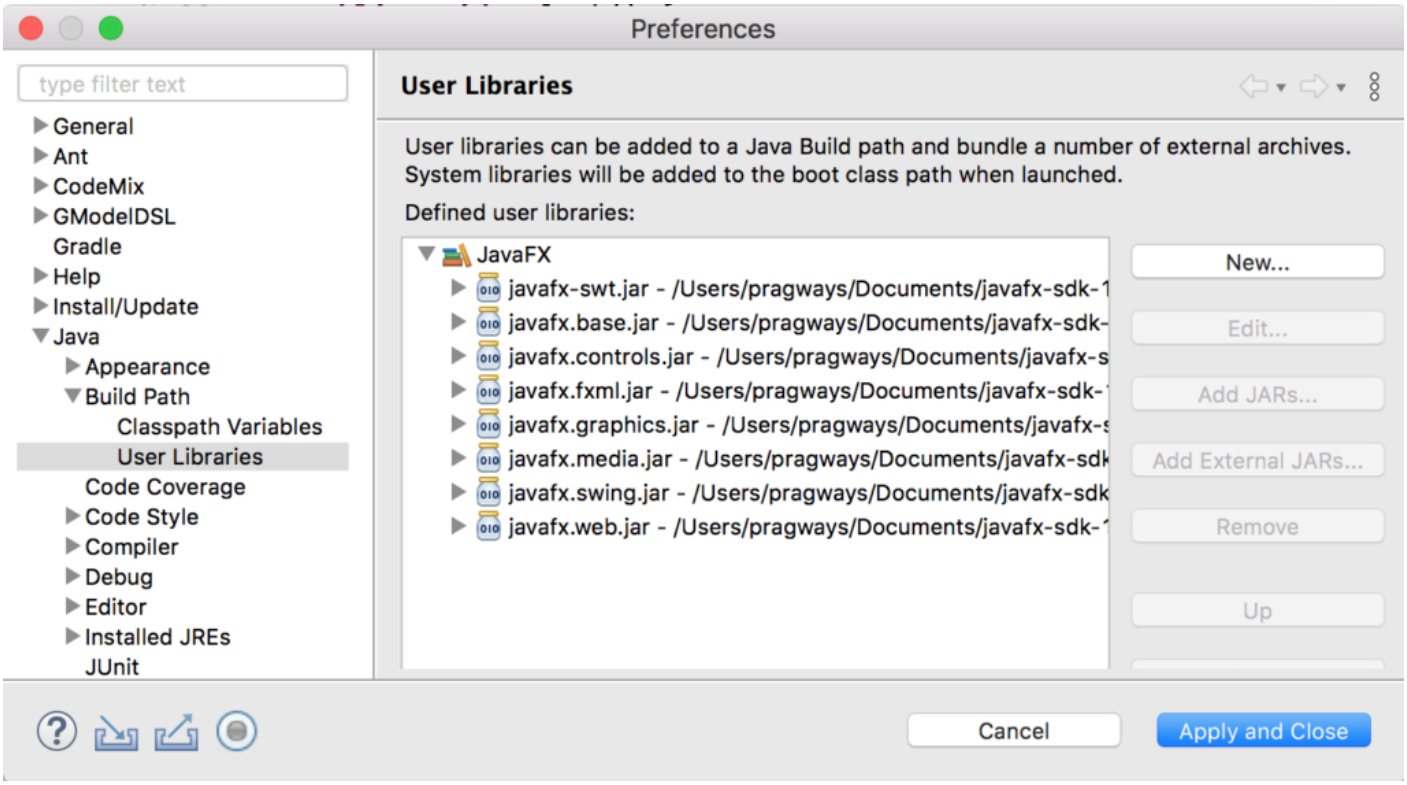
- Name it **JavaFX** and click **Ok**
- With JavaFX highlighted, click **Add External JARs...**



- Select all the `.jar` files from the `/lib/` folder of the unzipped JavaFX folder we saved earlier.



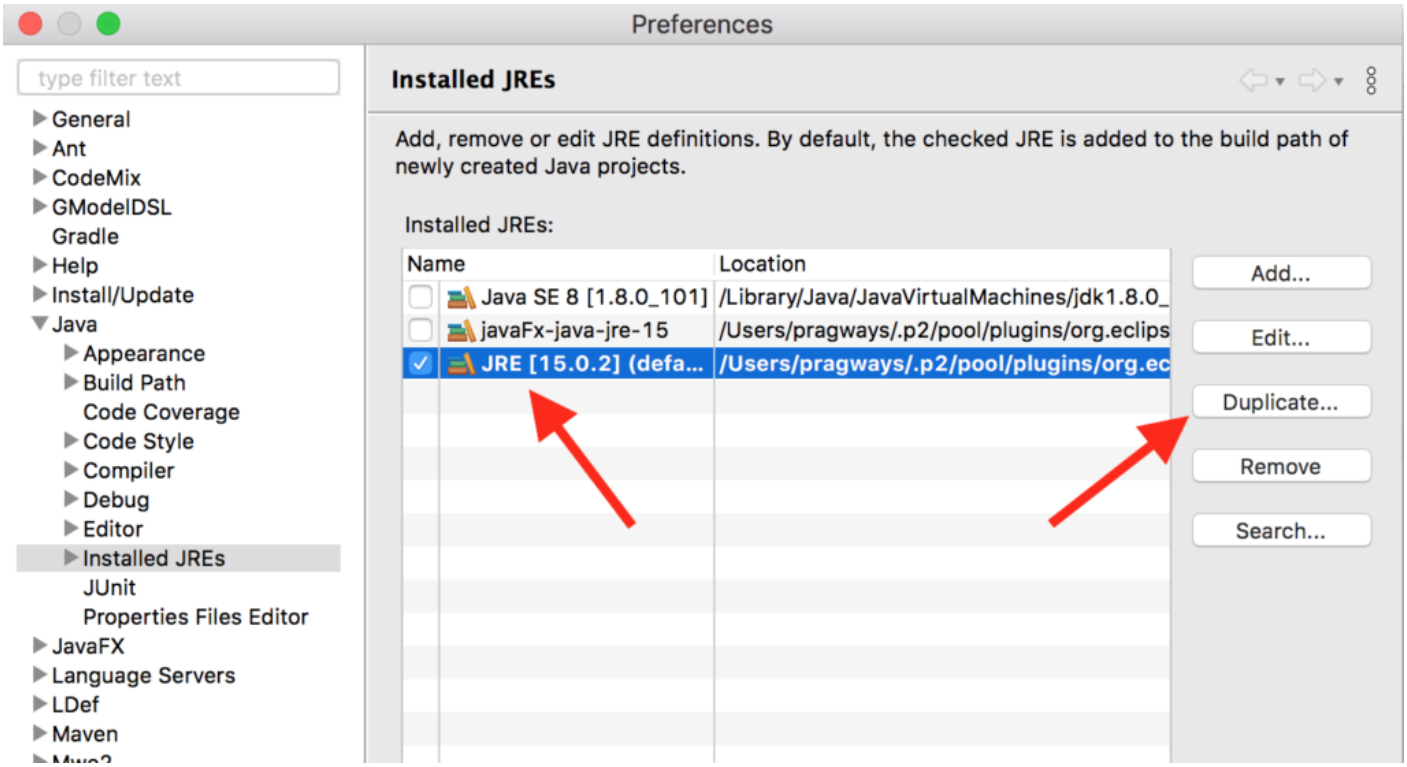
- Click **Open** and your new User Library should look something like this:



- Click **Apply and Close** to save your new User Library

STEP 3. CREATE A JRE CLONE WITH THE REQUIRED VM ARGUMENTS

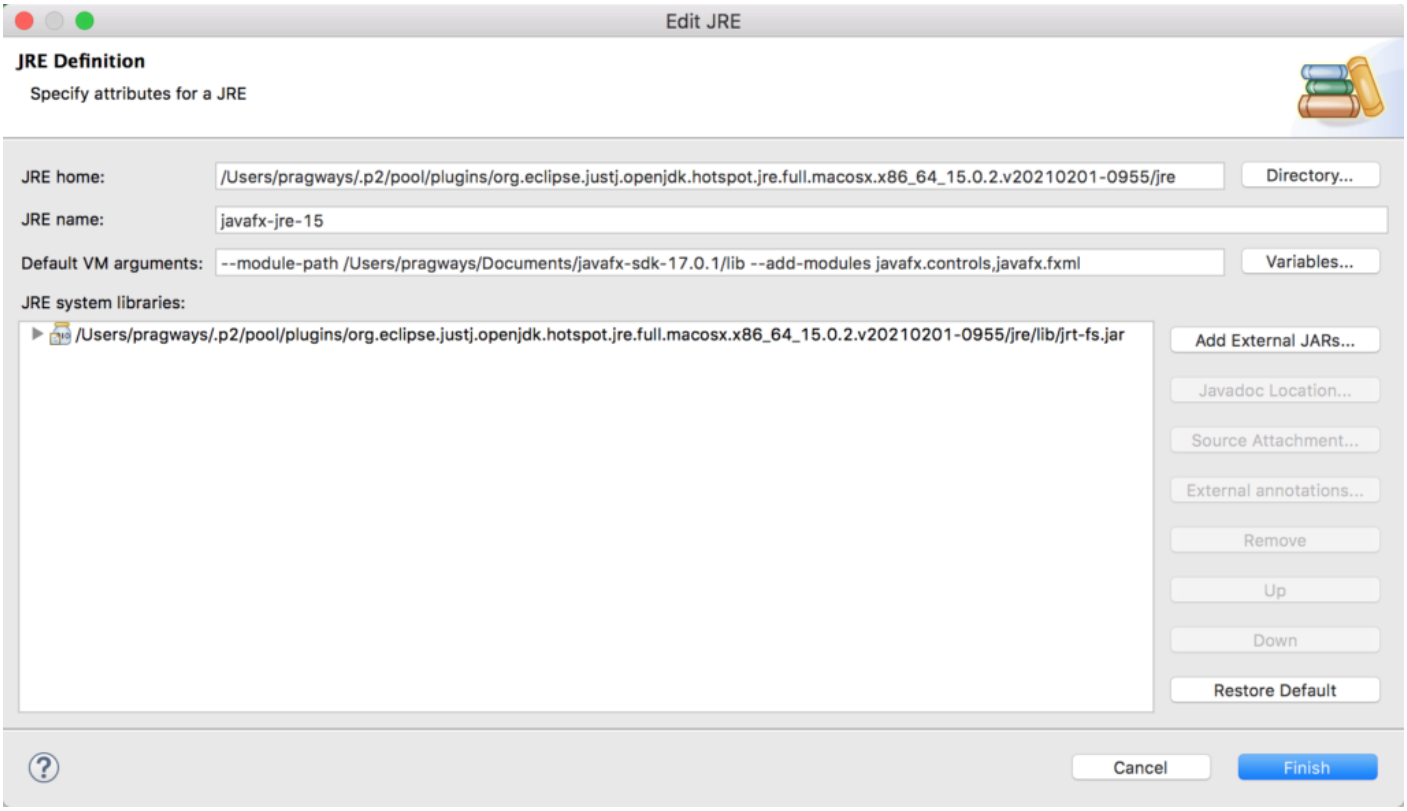
- Go back to **Preferences** in Eclipse
- Go to **Java** > **Installed JREs**
- Select your default JRE and click **Duplicate**



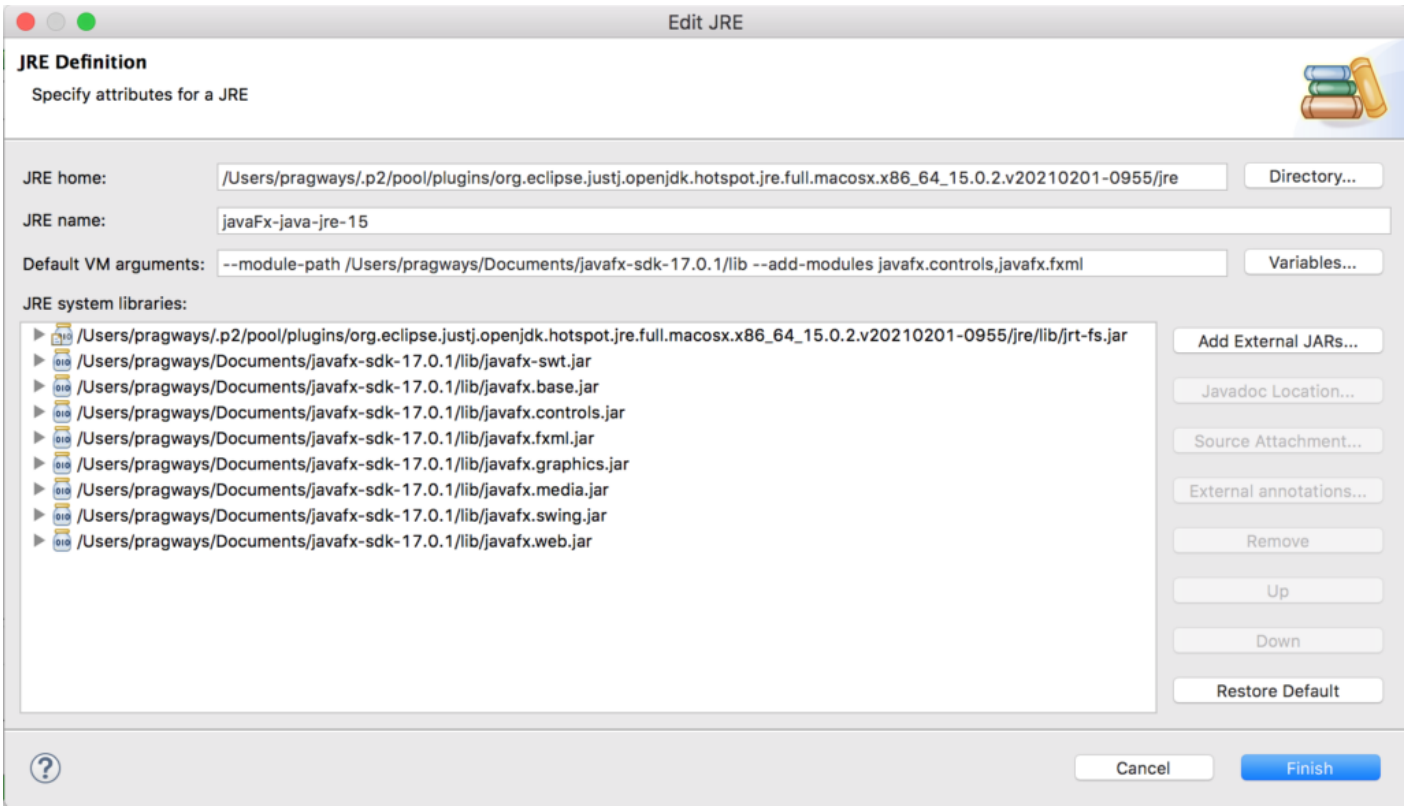
- Copy this line if you're on **Windows**
- `--module-path "\path\to\javafx-sdk-17\lib" --add-modules javafx.controls,javafx.fxml`
- Copy this line if you're on **Mac** or **Linux**
- `--module-path /path/to/javafx-sdk-17/lib --add-modules javafx.controls,javafx.fxml`

- **Important:** Make sure you replace the `/path/to/javafx-sdk-17/lib` with the path to where you placed your unzipped JavaFX folder from before. You will want the entire path all the way to the /lib/ folder. For instance, mine on a Mac looks like this:
- `--module-path /Users/pragways/Documents/javafx-sdk-17.0.1/lib --add-modules javafx.controls,javafx.fxml`

- Paste that line in the **Default VM arguments:** field
- Rename the **JRE name:** field to something memorable, such as **javafx-jre-15**



- Click **Add External JARs...**
- Select all the `.jar` files from the `/lib/` folder of the unzipped JavaFX folder we saved earlier and then click **Open**

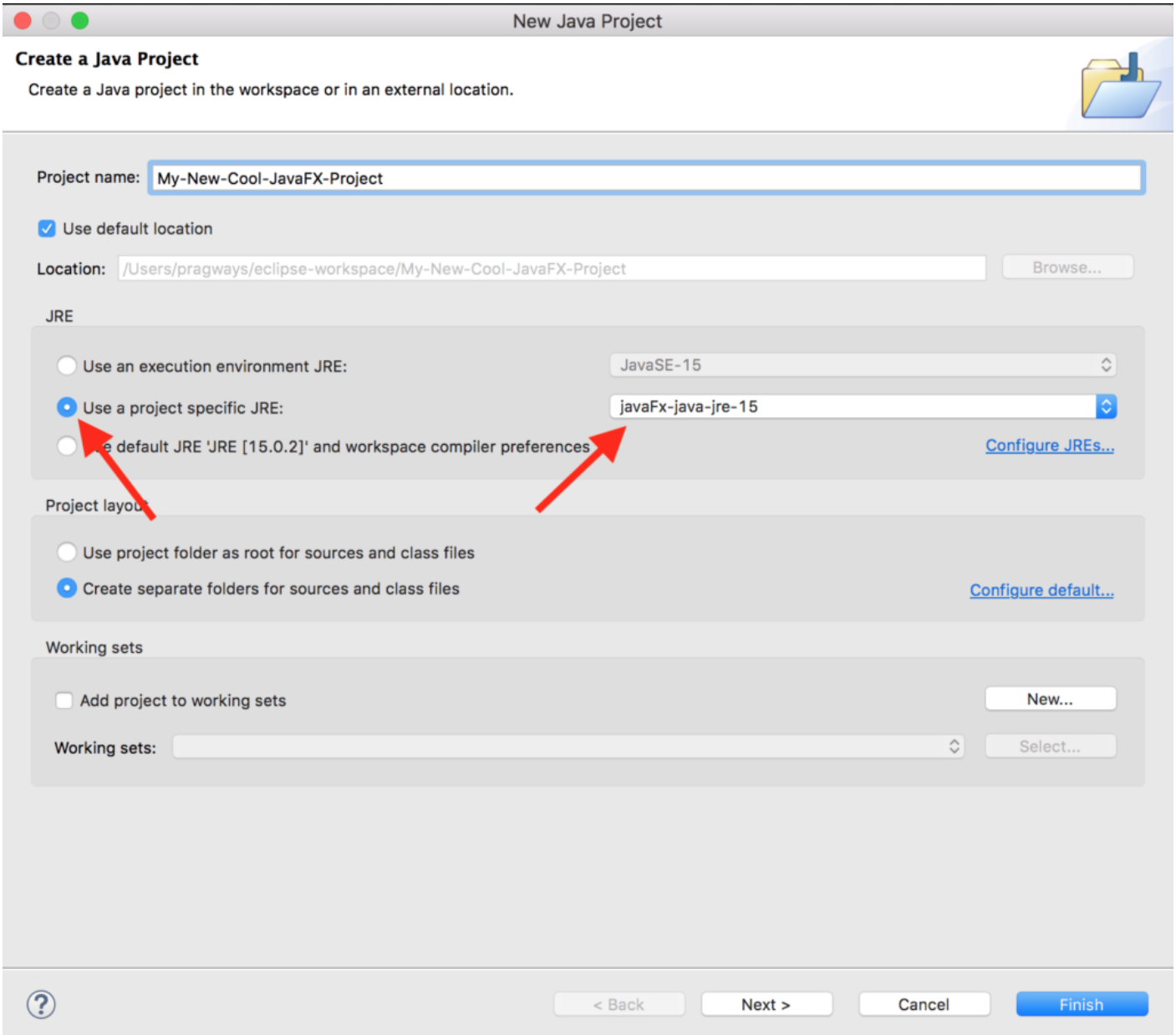


- Your JRE Definition window should now look something like this
- Click **Finish**

STEP 4. CREATE A NEW PROJECT WITH YOUR NEW JRE

Now whenever you want to create a new JavaFX project, it will be a lot quicker and easier!

All you need to do is make sure you select **Use a project specific JRE** when creating a new project, and then select your new JavaFX specific JRE that you just created.



Adam Allard

Hi, I'm Adam Allard. I'm a Full Stack Software Engineer for Northrop Grumman creating web applications for the DoD. At this time I'm primarily working with Java and Angular based applications, although I have some years of experience in various languages and frameworks from previous jobs where I dabbled with Python & Django, C# & Xamarin, PHP, and Bootstrap. My hobbies include time with my family, wondering when the Green Bay Packers will win their next Super Bowl, drinking over-priced beer, and of course learning and teaching.

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Save my name and email in this browser for the next time I comment.

POST COMMENT

Recent Posts

Unit Conversions

The tables below are all the unit conversion factors used in the Unit Converter App tutorial, where we walked through together building out a JavaFX and SceneBuilder interface. Click here to...

CONTINUE READING

Here's why you should use Deque instead of Stack or Queue

A Deque is a double ended queue. It can be used as a Queue or a Stack If you've studied your basic Data Structures and Algorithms, then I'm sure you're familiar with the Queue and the Stack data...

CONTINUE READING

ABOUT

Pragmatic Ways is dedicated to help developers write better code. Many developers in the real world either forgot the clean coding techniques they learned back in school or never learned them in the first place. On top of that, it's one thing to have learned about the basic Object Oriented Programming concepts, but it's another thing to know when, where, why, and how to actually use them writing complex, robust, and maintainable applications.

This blog focuses on Software Engineering best practices, design patterns, data structures and algorithms, refactoring legacy software, and other critical aspects every developer should know to help them write quality code.

LEGAL FOOBAR

Hi, my name is Adam Allard and I own and operate this site, Pragmatic Ways. Full Disclosure: I am a participant in the Amazon Services LLC Associates Program, an affiliate advertising program designed to provide a means for sites to earn advertising fees by advertising and linking to Amazon.com. I also participate in other affiliate programs with NameCheap, Clickbank, CJ, ShareASale, and other sites. I get compensated a few pennies every time you click on one of these links and purchase a product or service. I promise to only promote products and services I use, trust, and stand by myself. If you continue to click on these affiliate links and purchase services from these vendors, I can continue to purchase enough coffee to help me keep writing new content for this blog.

NEWSLETTER

Email address: Your email address

SUBSCRIBE NOW
