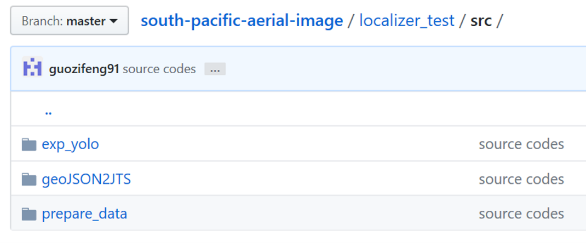
Here is a simple guide of how to run the JAVA localizer:

First you need to have a java develop platform, for example eclipse. Install the platform and you are ready to code in JAVA.

Then, start a new project and copy-paste all the code in “localizer\_text/src” from github to your JAVA project. Note that the prepare\_data folder is not necessary.



Then, install three libraries: tensorflow, processing 2.2.1, and OpenCV 3.3.:

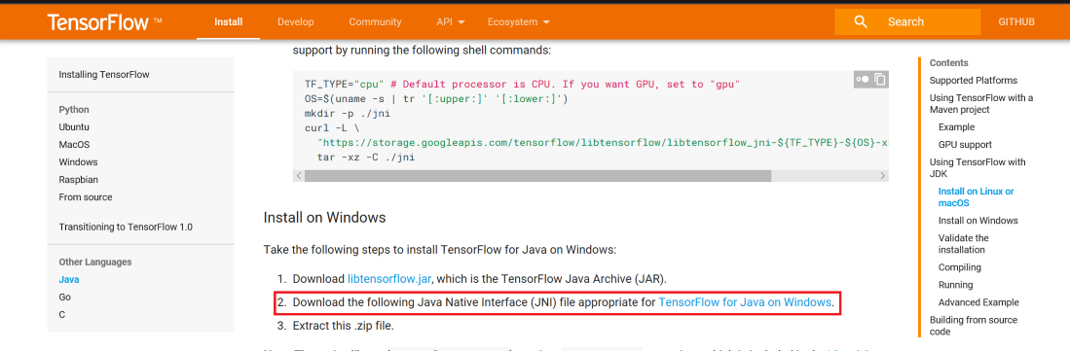
<https://www.tensorflow.org/install/install_java#using_tensorflow_with_jdk> (tensorflow)

<https://processing.org/download/> (processing)

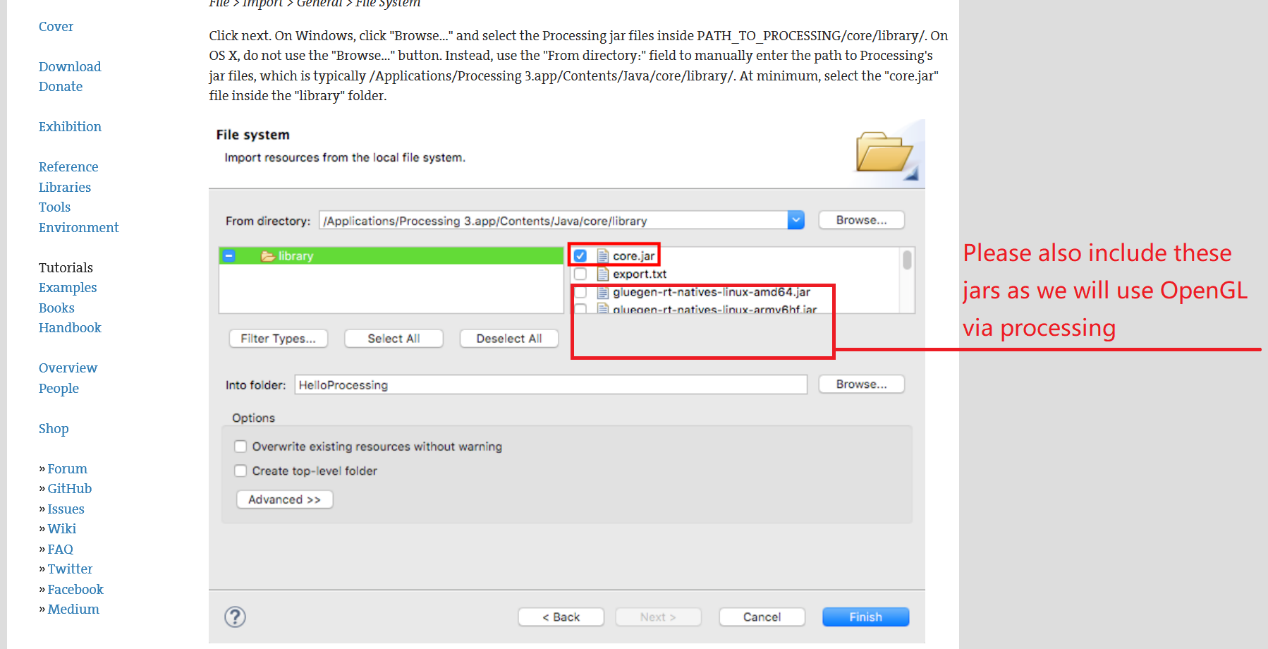
<https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.3.1/opencv-3.3.1-vc14.exe/download> (open cv)

The jar files of these libraries must be included into the build path of your java project, in addition, they have some native runtime files (.dll or .jni) that must be included as well. For tensorflow and processing, follow these links

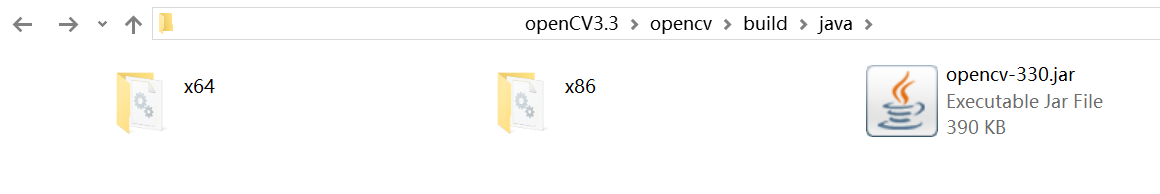
<https://www.tensorflow.org/install/install_java#using_tensorflow_with_jdk> (tensorflow)



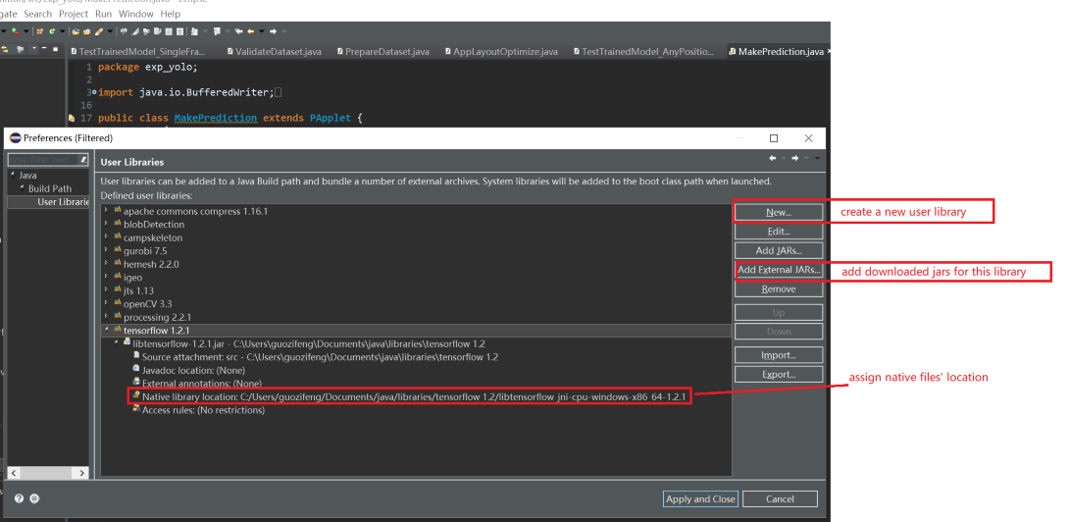
<https://processing.org/tutorials/eclipse/> (processing)



for opencv, after download and install it on your computer, add opencv-330.jar and opencv\_java330.dll (from one of the folders on the left, depend on if the system is 32 bit or 64 bit) into the build path (below).



It is recommended to use “User Library” to import the jars and the native runtime files: right click project name – building path – user libraries – next – user libraries, a window like this will show up:



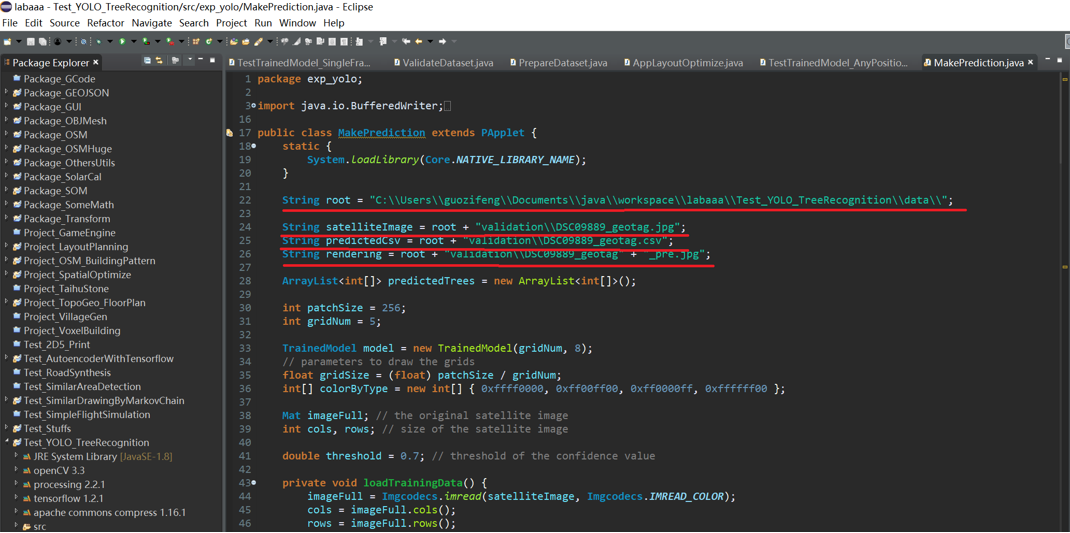
Click New to create a new user library, where you need to give a name.

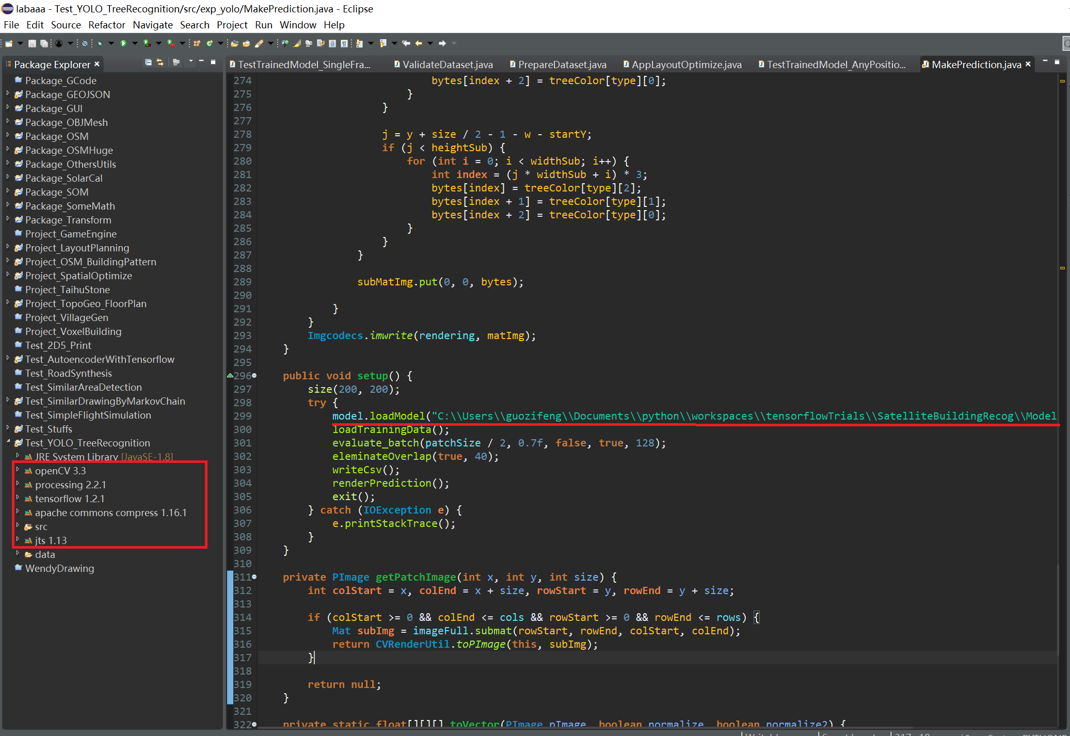
Then, choose Add External JARs to include all the necessary jar files to the user library

**Create one user library for each library, not for all of them**

Then, set the native files’ location to where they are on your drive, click Apply and Close

Now the code is ready, one more step before launch them: you need to change some paths in the code (**MakePrediction.java** and **TestTrainedModel\_AnyPosition\_ForVideo.java**) to where you store your jpeg imagery and the .pb file, the paths will look different if you are using Mac not Windows (below, red lines in the code)





Then that’s it, click run and the program should run properly.

**MakePrediction.java** takes a full-size imagery as input, and exports coordinates and types of all recognized trees as csv file, and renders an equal-size bounding box image which can be overlapped with the input one.

**TestTrainedModel\_AnyPosition\_ForVideo.java** shows a user interface where you can navigate through the input imagery and see the recognition result of the place of interest.