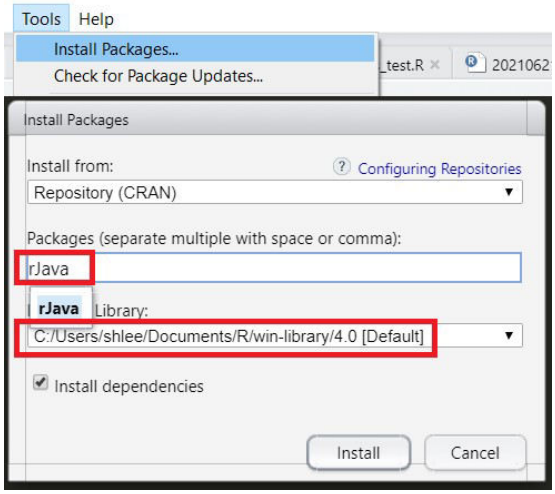
…Installing rJava

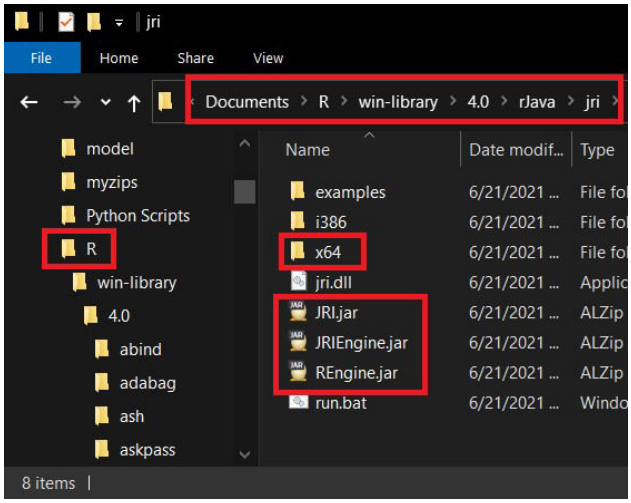
we need to install rJava package in R studio.



We need to know the directory where rJava package is installed. In our case, this directory is as follow. (this is dependent on your Windows system)

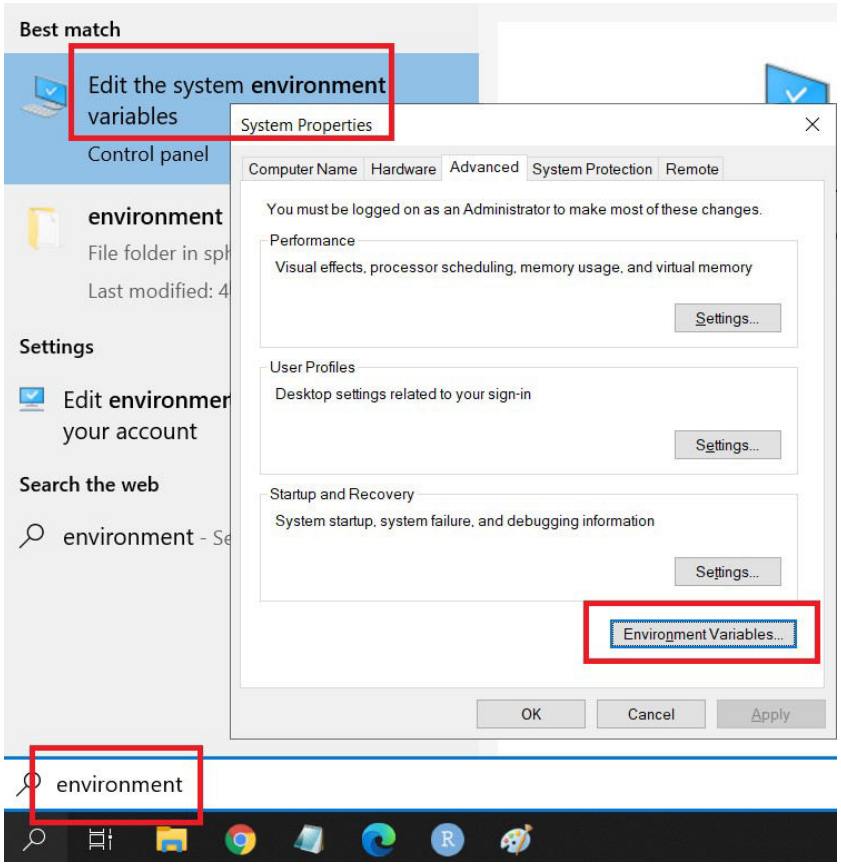
C:\Users\shlee\Documents\R\win-library\4.0

The following figure displays files of **rJava/jri** directory.



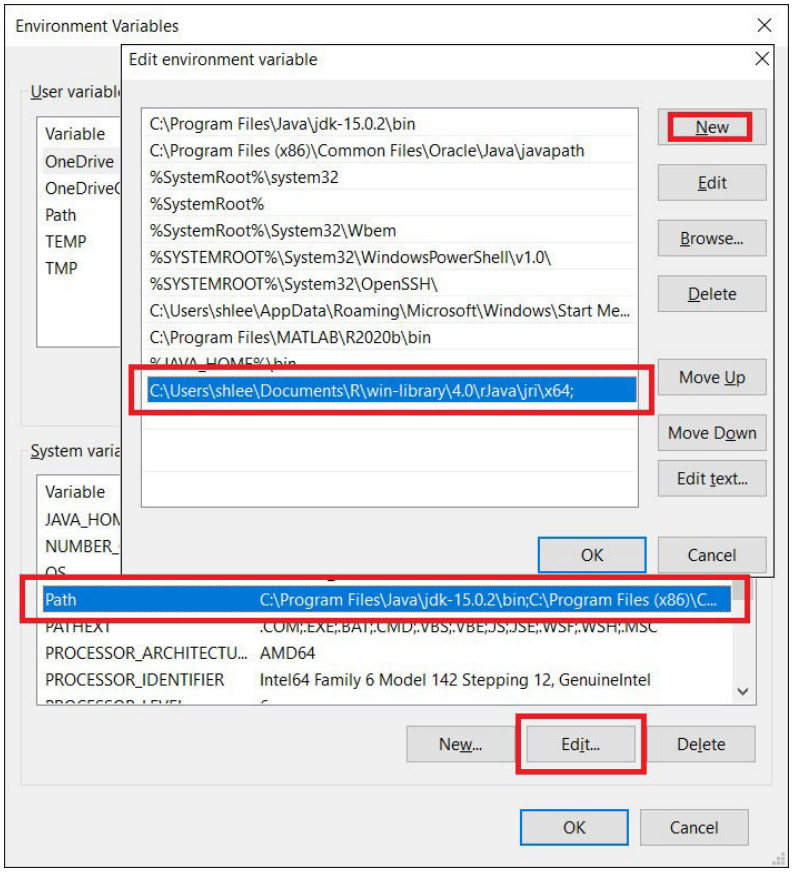
Environment Variables

Find **"Edit the system environment variables"** and open **System Properties** box, and click **Environment Variables …** button.



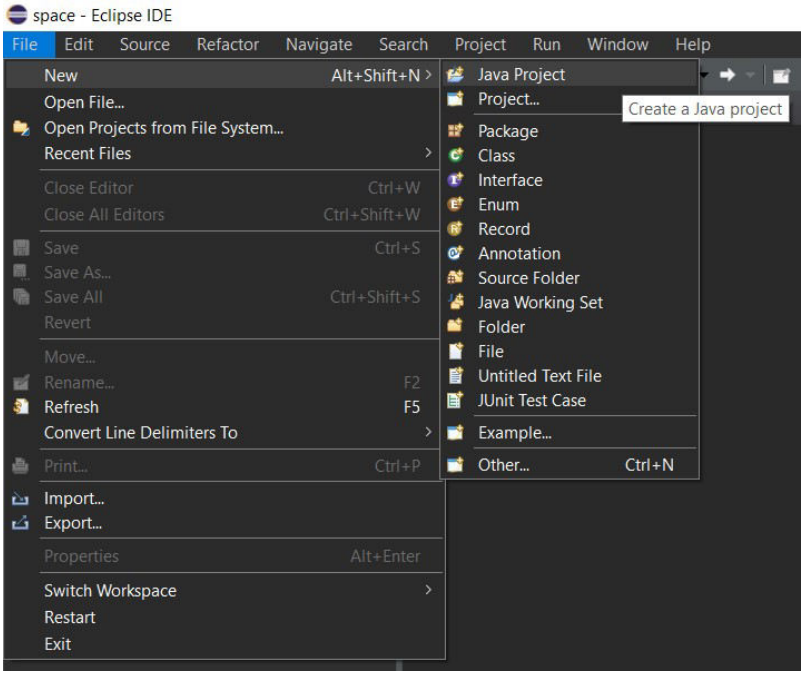
Add **Path** as a new variable in **System Variables** as follows.

- Path : C:\Users\shlee\Documents\R\win-library\4.0\rJava\jri\x64;



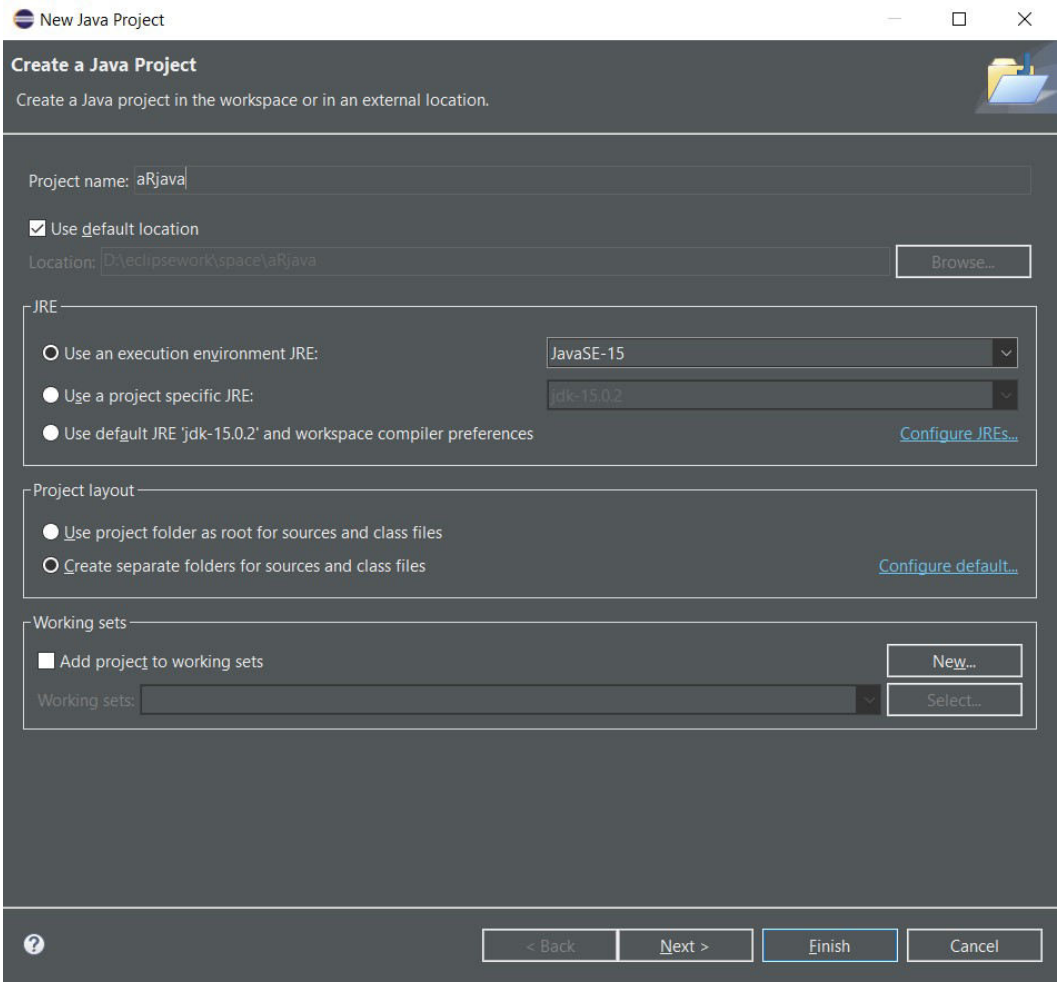
Eclipse Project and Class

Make a new project in Eclipse : **File** -> **New** -> **Java Project** .

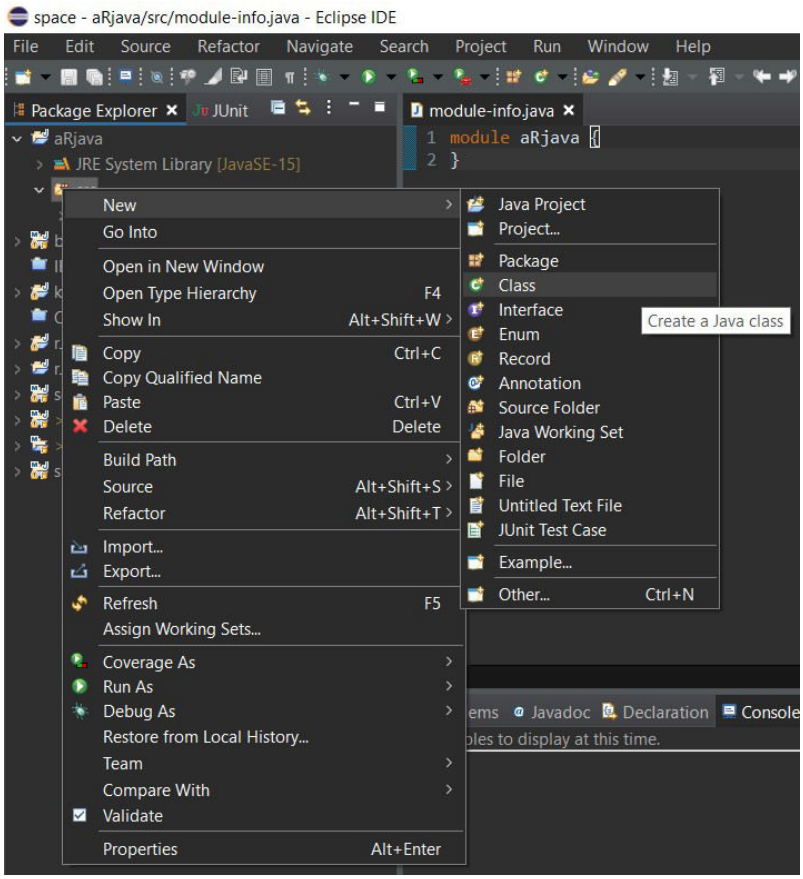


Insert **aRjava** or your favorite name as **Project name**.

- Project name : aRjava

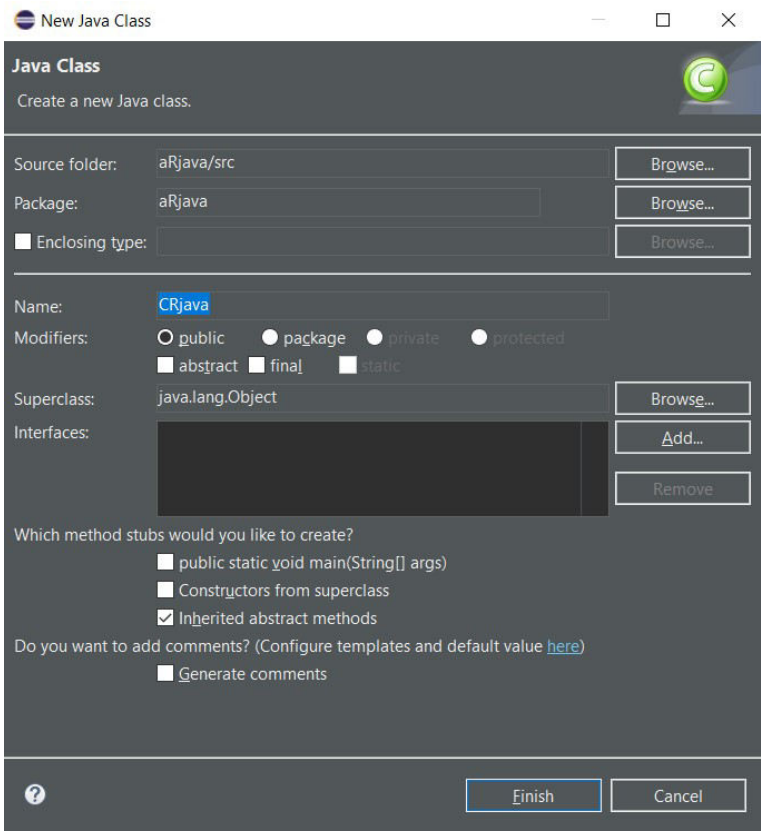


Make a new class : **File** -> **New** -> **Class** .



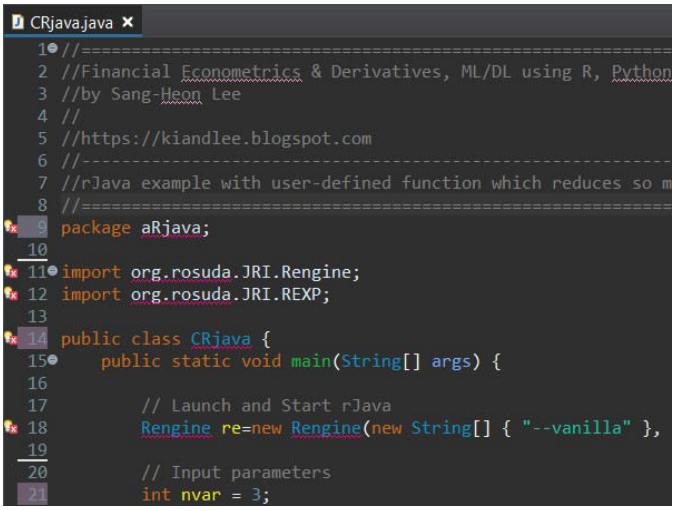
Insert **CRjava** or your favorite name as **Name**.

- Name : CRjava



Writing a Java code with rJava

Write a code using rJava in CRjava class file.

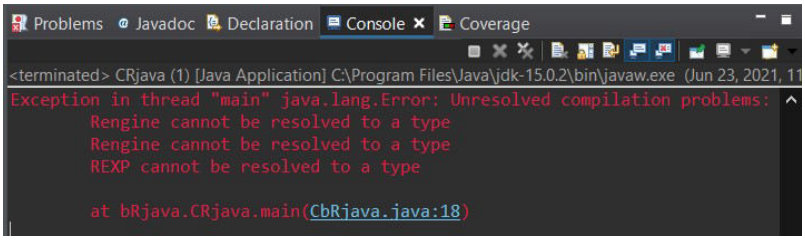


The above Java code is to call R command for the eigen decomposition and return eigenvector. Matrix multiplication with its transpose ensures the positive definite. The full Java code is as follows.

```
//============================================================================#
// Financial Econometrics & Derivatives, ML/DL using R, Python, Tensorflow
// by Sang-Heon Lee
//
// https://kiandlee.blogspot.com
//----------------------------------------------------------------------------#
// rJava example with built-in R commands
//============================================================================#
package aRjava;

import org.rosuda.JRI.Rengine;
import org.rosuda.JRI.REXP;

public class CRjava {
   public static void main(String[] args) {

      // Launch and Start rJava
      Rengine re=new Rengine(new String[] { "--vanilla" }, false, null);

      // Input parameters
      int nvar = 3;

      // R commands through rJava
      re.eval("ma <- matrix(rnorm("+nvar+","+nvar+"),"+nvar+","+nvar+")");
      re.eval("mb <- ma%*%t(ma)");
      re.eval("eg <- eigen(mb)");
      REXP x = re.eval("eg$vectors");

      // Result : raw output
      System.out.println("Result as a raw output");
      System.out.println(x+"\n");

      // R matrix --> Java 2D array
      double[][] mout = x.asDoubleMatrix();

      System.out.println("Result as a 2D array");
      for(int i = 0; i<nvar; i++) {
         for(int j = 0; j<nvar; j++) {
            System.out.print(String.format("%2.5f", mout[i][j]) + "   ");
         }
         System.out.println("");
      }

      // end rJava
      re.end();
   }
}
```

Colored by Color Scripter cs

When we run the above Java code, we encounter the following errors which are obstacles in the way. Hence, we need to some settings for Eclipse.



But this first running this project is important because after this trial, **Run Configuration** (which will be explained later) can identify this project.

Eclipse Project Setting

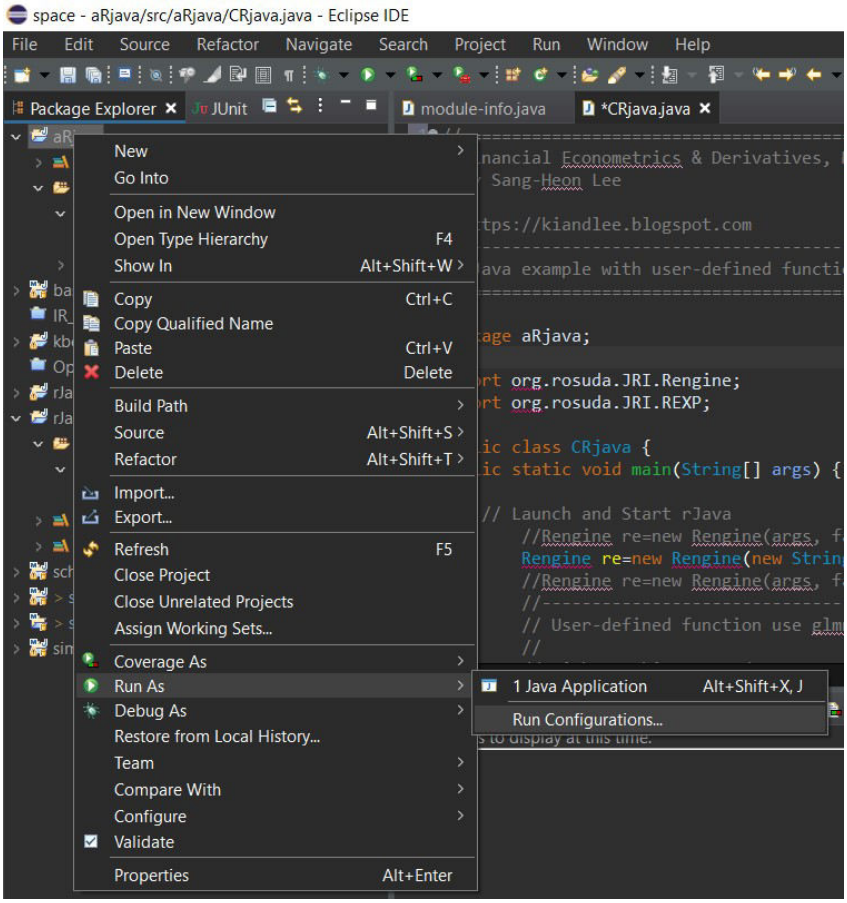
This is the highlight section of this post. This setting seems so complicated but let's do it.

The bottom line for the Eclipse setting is that we have only to **add two settings on our project not class files**.

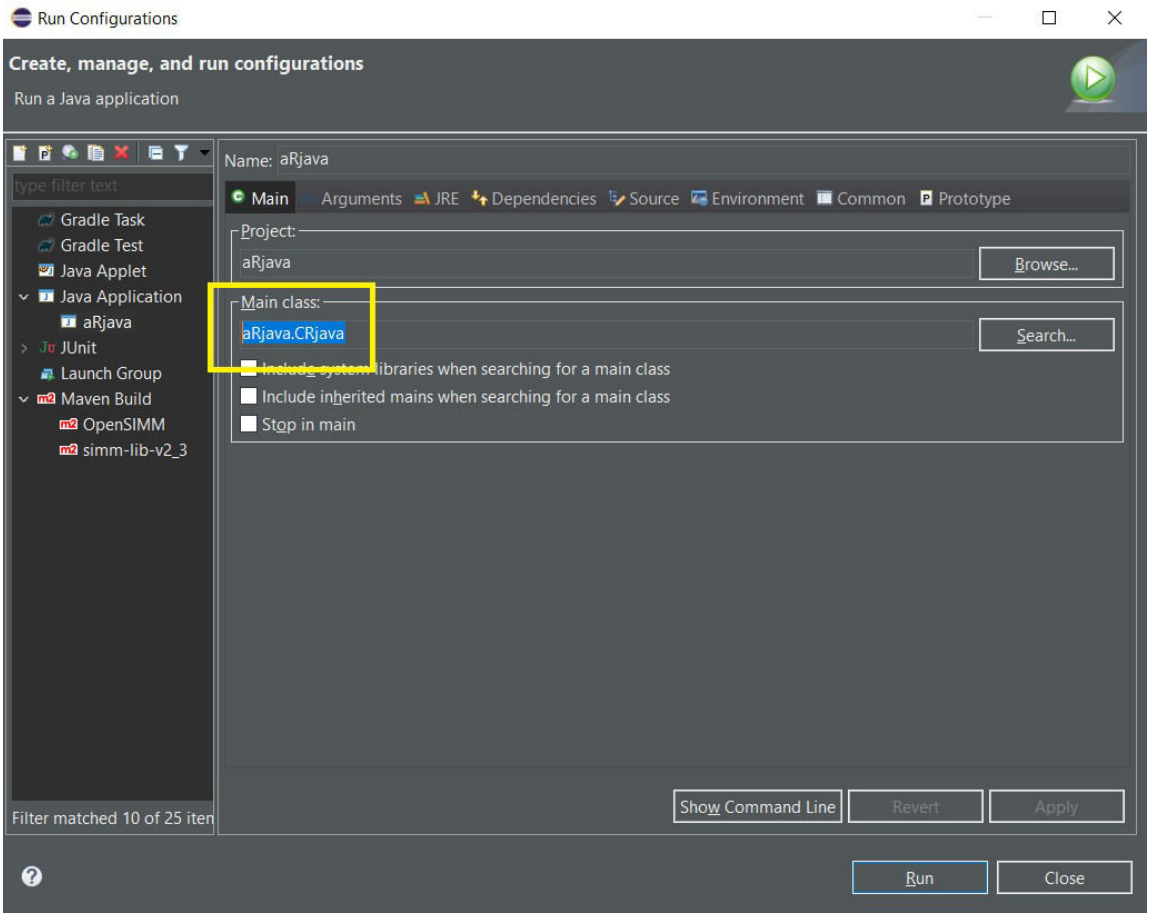
- Project –> Run As –> Run Configurations
- Project –> Build Path –> Configure Build Path…

Eclipse Project Settings – Run Configurations

Open **Project** –> **Run As** –> **Run Configurations**.

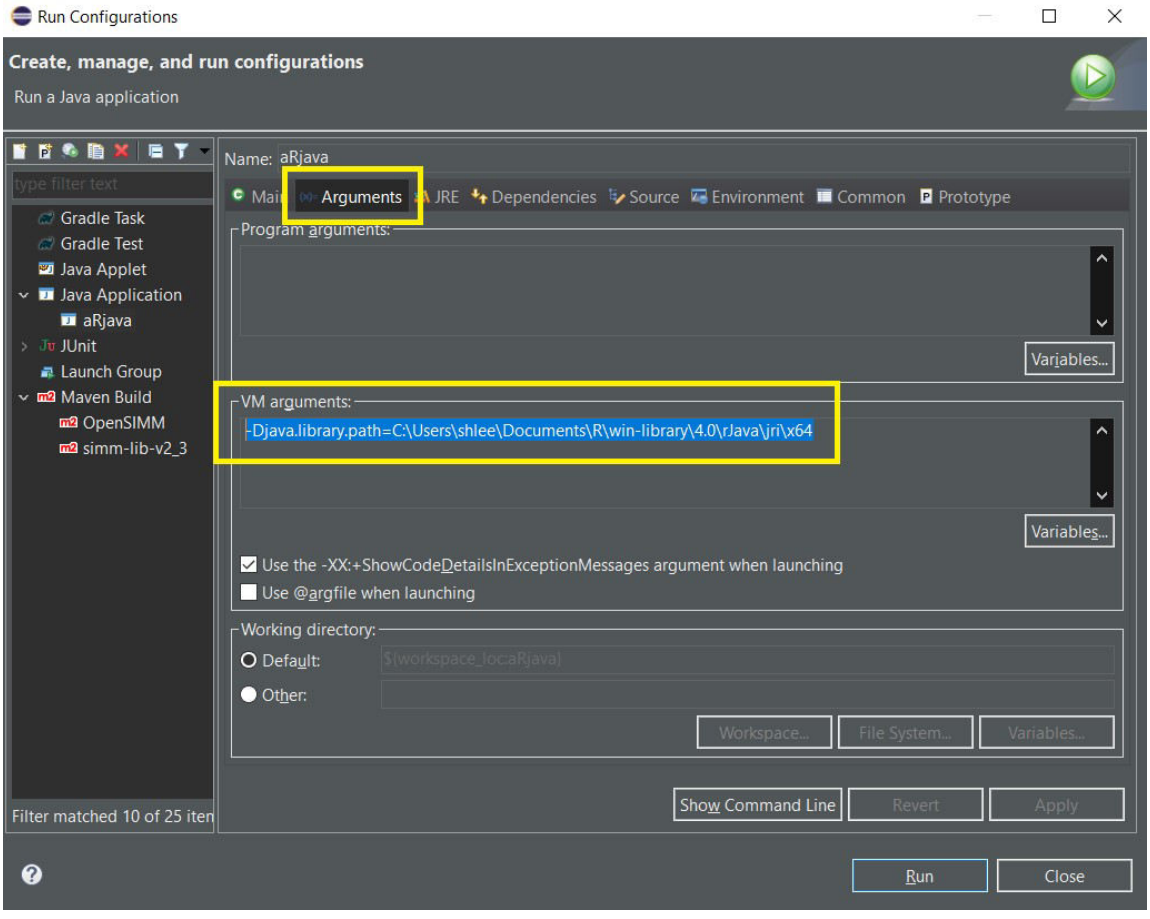


At first, we need to check whether **Project** and **Main class** are set to **aRjava** and **aRjava.CRjava** respectively.



In **Arguments** tab, **VM arguments** is filled as follows.

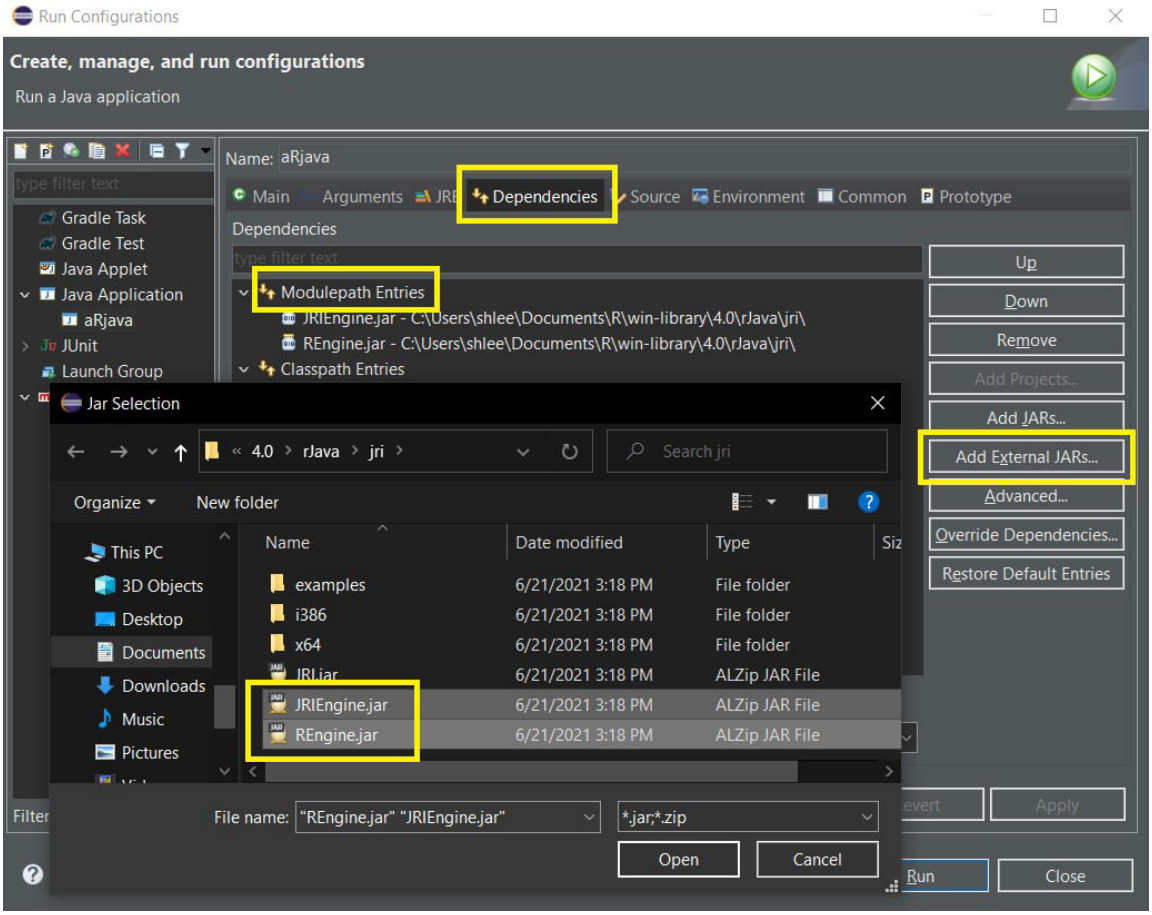
- VM arguments : -Djava.library.path=C:\Users\shlee\Documents\R\win-library\4.0\rJava\jri\x64



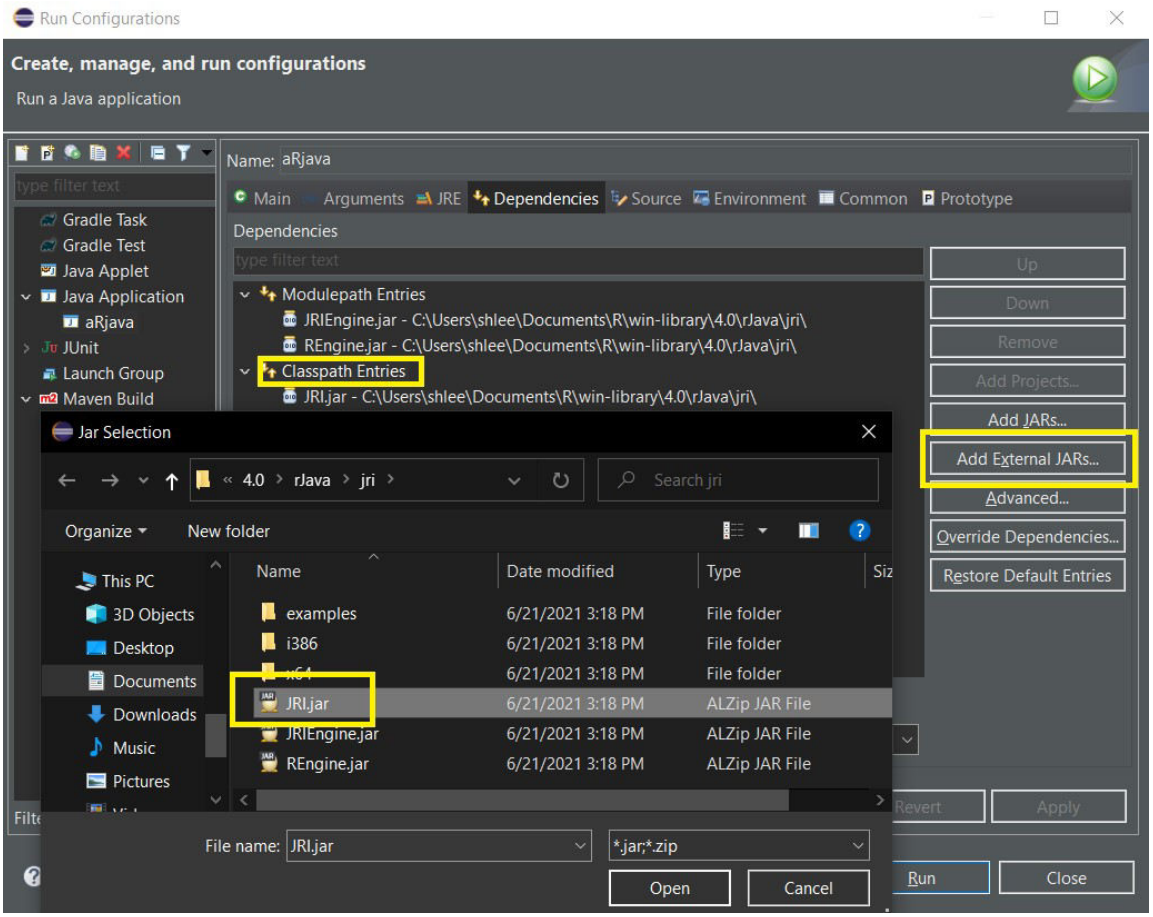
In **Dependencies** tab,

It is most important to distinguish between Modulepath Entries and Classpath Entries with respect to kind of jar file and project.

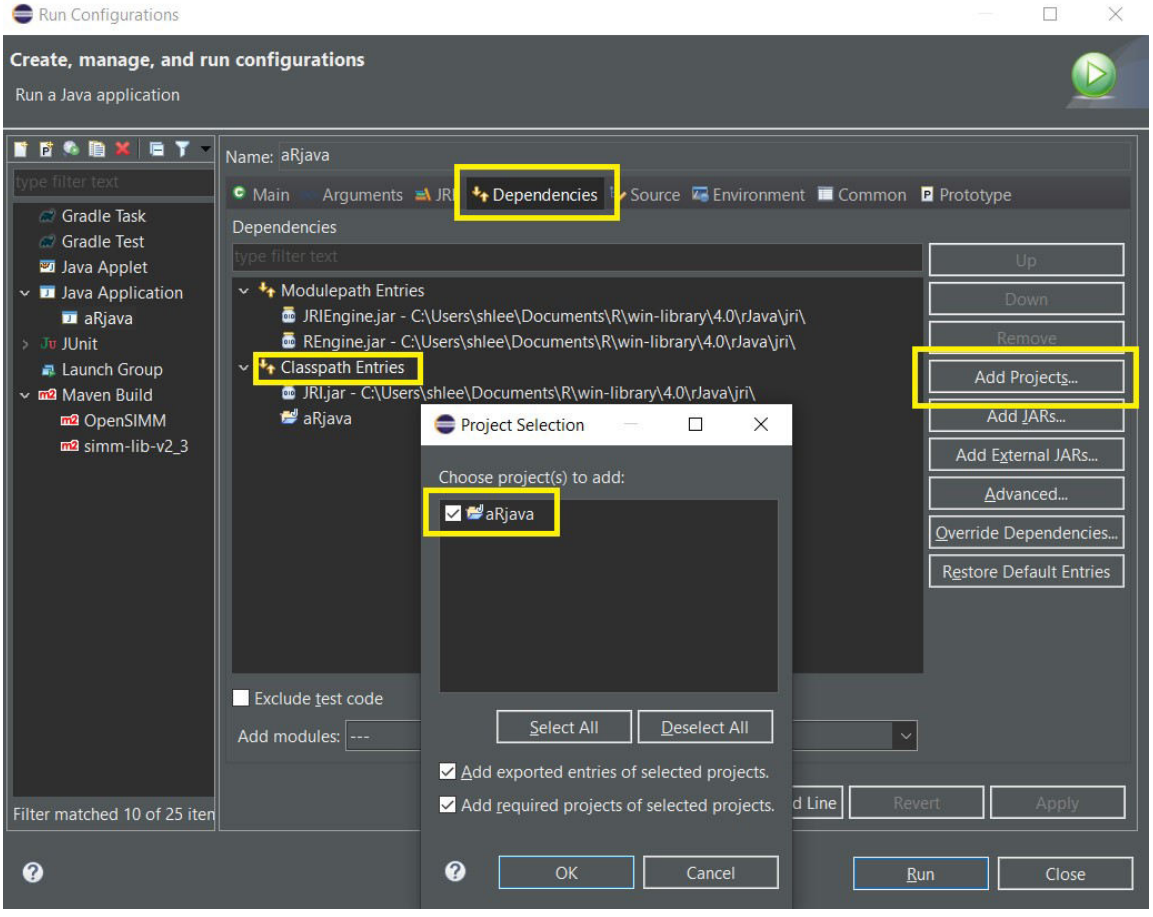
1) For **JRIEngine.jar** and **REngine.jar**, Add External JARs… to **Modulepath Entries.**



2) For **JRI.jar**, Add External JARs… to **Classpath Entries**



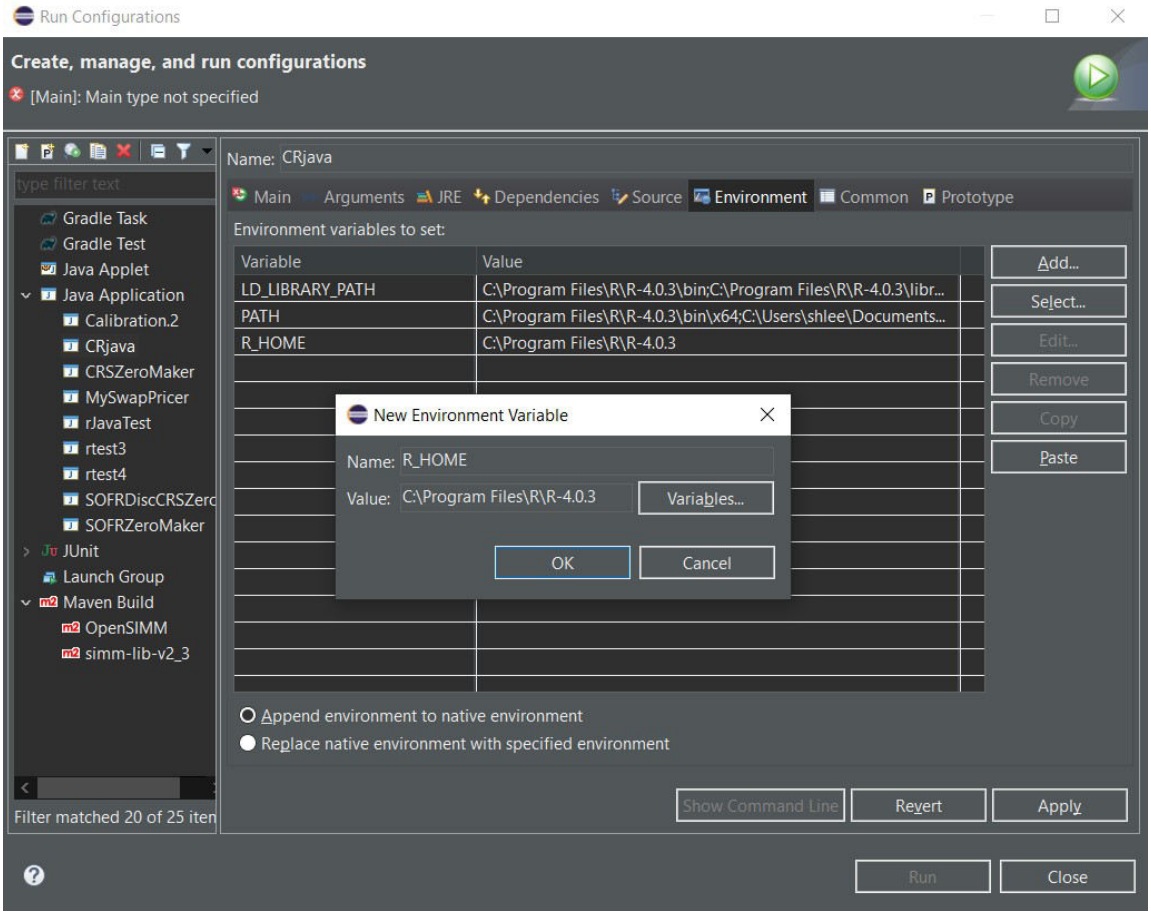
3) For **aRjava**, **Add Projects…** to **Classpath Entries** only if aRjava is absent in Classpath Entries.



In **Environment** tab, Add three directories in the following way.

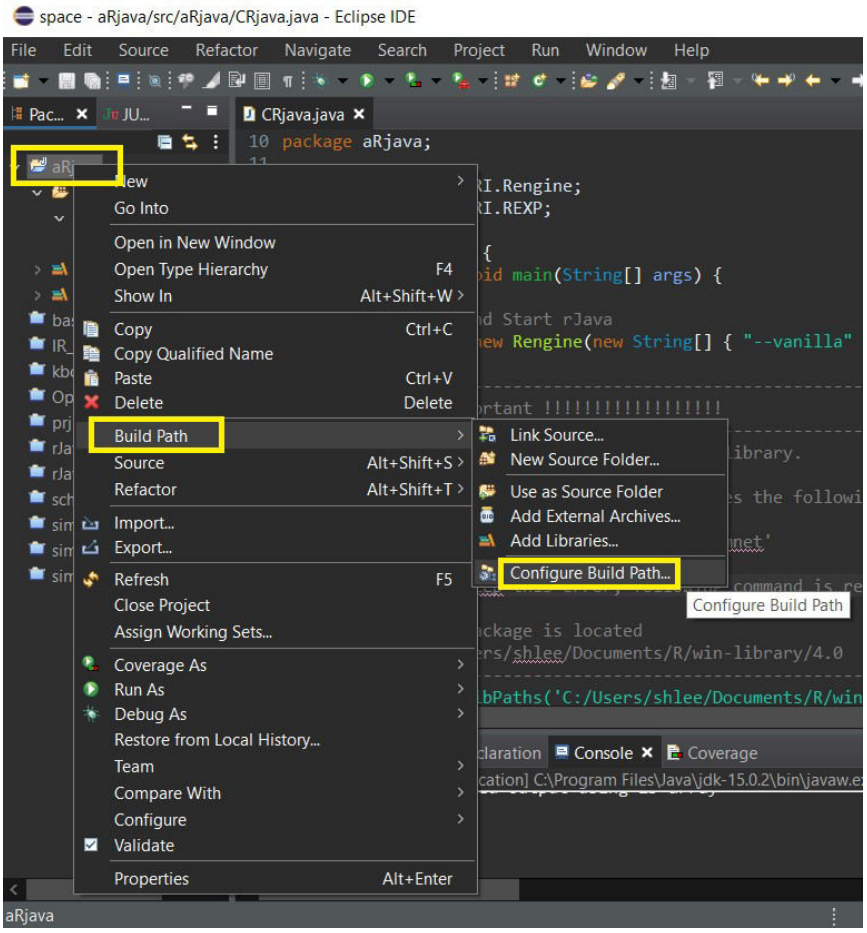
- LD_LIBRARY_PATH : C:\Program Files\R\R-4.0.3\bin;C:\Program Files\R\R-4.0.3\library;C:\Users\shlee\Documents\R\win-library;
- PATH : C:\Program Files\R\R-4.0.3\bin\x64;C:\Users\shlee\Documents\R\win-library\rJava\jri\x64;
- R_HOME : C:\Program Files\R\R-4.0.3

Here, in case of R_HOME, there is no ";".



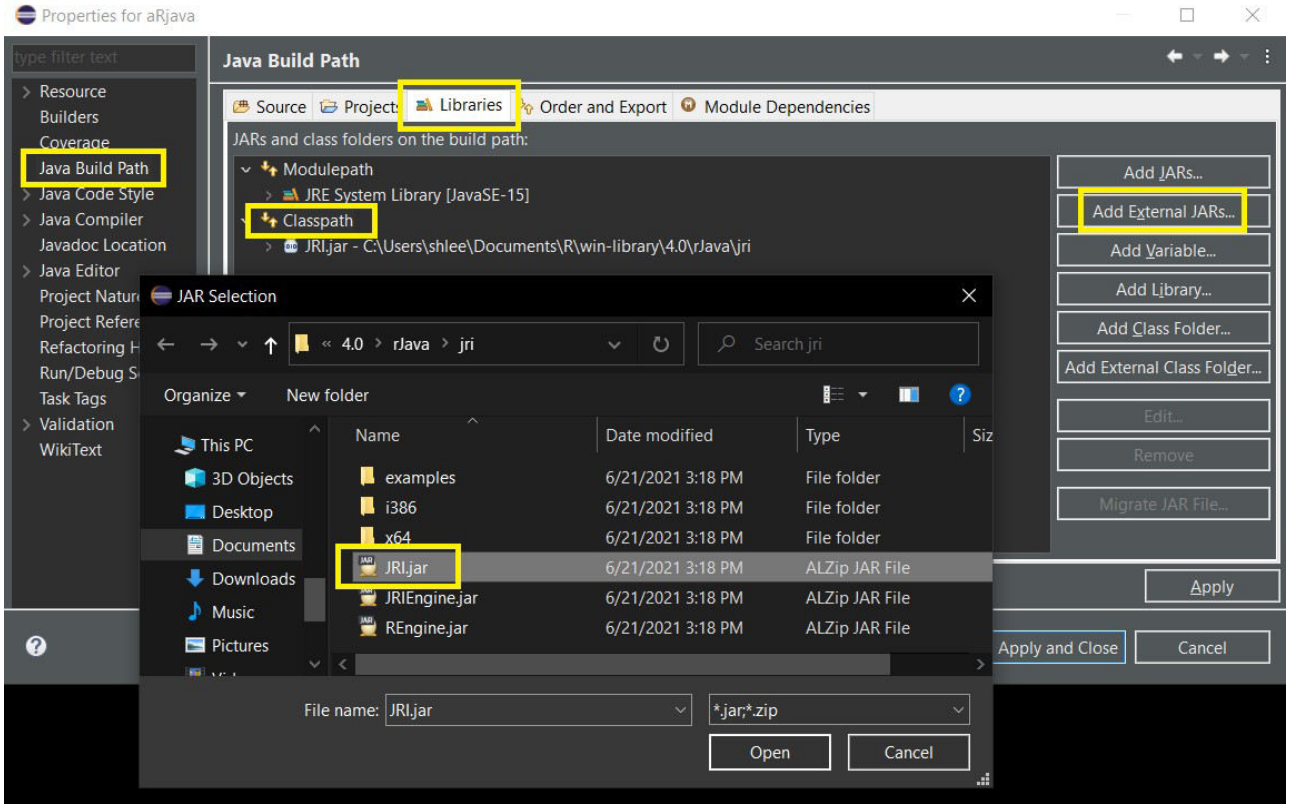
Eclipse Project Settings – Configure Build Path…

As the setting for **Run Configurations** is completed, it's time to add settings on **Configure Build Path…**.



In **Libraries** tab,

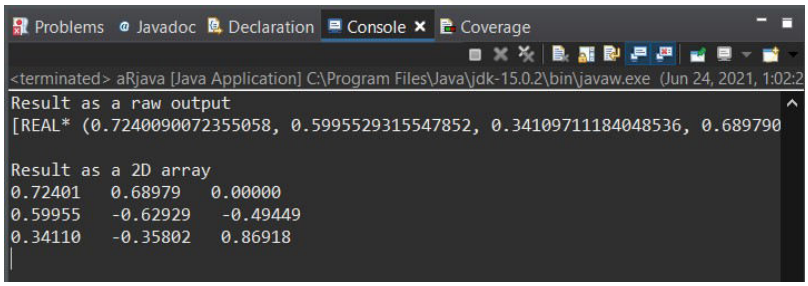
For **JRI.jar**, Add External JARs… to **Classpath.**



Now the setting for Eclipse Project is done completely.

Running and Results

When we rerun CRjava.class file, We can obtain correct results.



Although setting for rJava in Eclipse is so complicated, we try to summarize this setting process with many detailed screen captures for clear understanding.

We expect that rJava is quite useful when R is more suitable for some complicated calculations or estimations when Java is a main program.