Gradle

The following will show you a couple ways to use gridle to add external dependencies into your code one of the best recourses for using something like Gradle will be https://mvnrepository.com/ It will give you the scripts for importing the dependencies into your project. such as

```
implementation group: 'org.apache.poi', name: 'poi', version: '5.0.0'
```

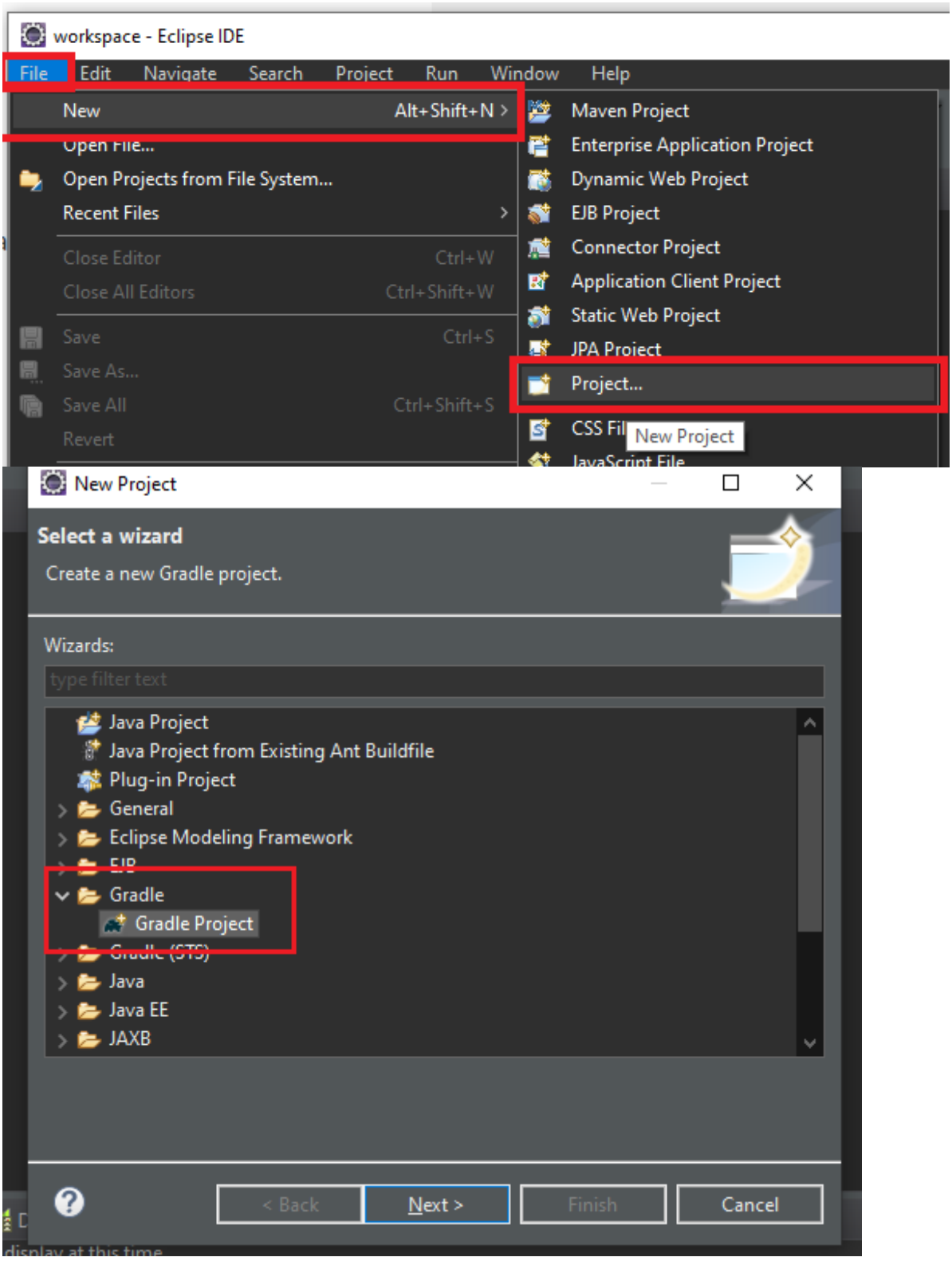
Introduction

For everything we will be using eclipse. It is one of the easiest free and functional IDE's.
For Gradle Specifically there are 2 methods of creating a project. The first is to use the IDE 's built in functionality and the other is to build it manually by installing Gradle and using windows cmd or bash for linux. We are only going to go over the Eclipse built in method

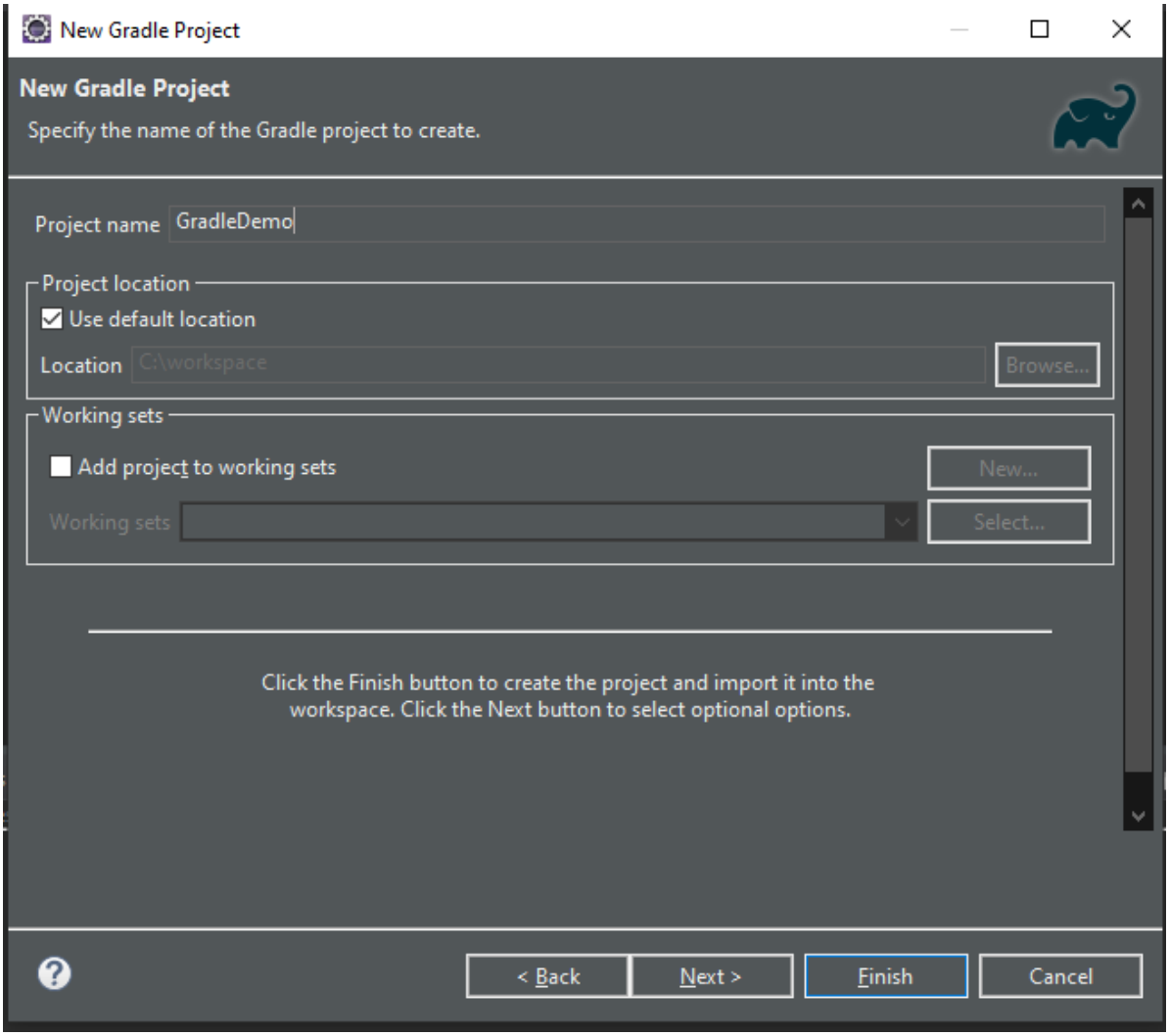
Eclipse

First thing is to make sure you have Eclipse for Java Enterprise Developers. The standard install doesn't have some of the tools that will make this much more simple.

To start with create a new Gradle project.

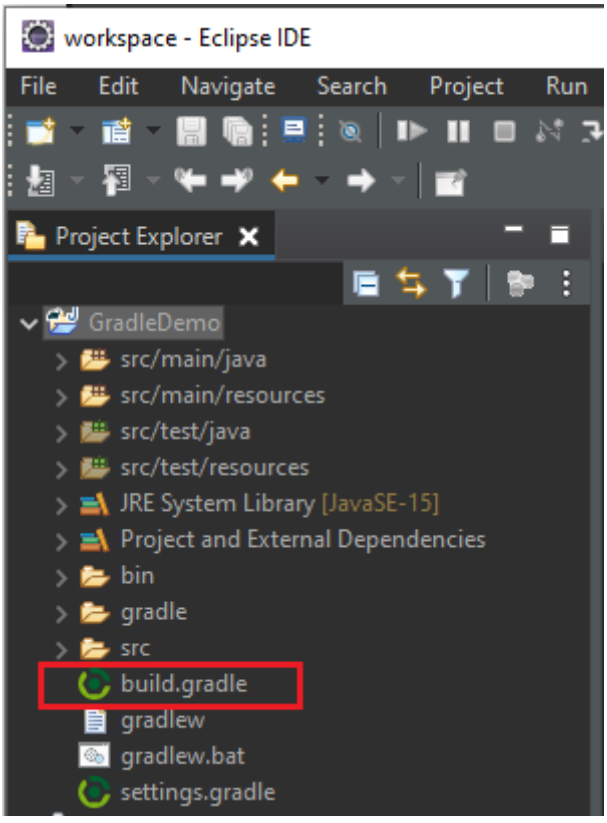


Click next until you get to the naming screen



Give it a simple name. In this case **GradleDemo** And then Click Finish.

Then Open up the new project and open the file build.gradle



For this tutorial we are going to stick with simple things that can be useful in the programing environment just to make sure that we have Gradle working. In your build.gradle file place the following in the dependencies block

```
    // This is a Package for a CSV Reader And writer.
    // https://mvnrepository.com/artifact/com.opencsv/opencsv
    implementation group: 'com.opencsv', name: 'opencsv', version: '4.0'```
```

This will add OpenCSV to your project if you want to know more there is a Users Guide And A Tutorial The next Step will be to take a CSV and read it, Process it and output it. into another csv. I have included a file. ***timepunches.csv***. Copy and paste this file into your Project. (You can copy the file and then Right Click on GradleDemo and click paste.)

timepunches.csv contains data with start and stop time. where row 1 is the persons name. row 2 is start and row 3 is the stop time. For this problem all the times will be presented in military time ***hh:ss*** and we are going to do the math to see the total time that each person worked. To keep this example simple only one record per person.

Employee	Start	Stop
Adam	8:00	17:00
James	7:30	17:00

Now to make the code.

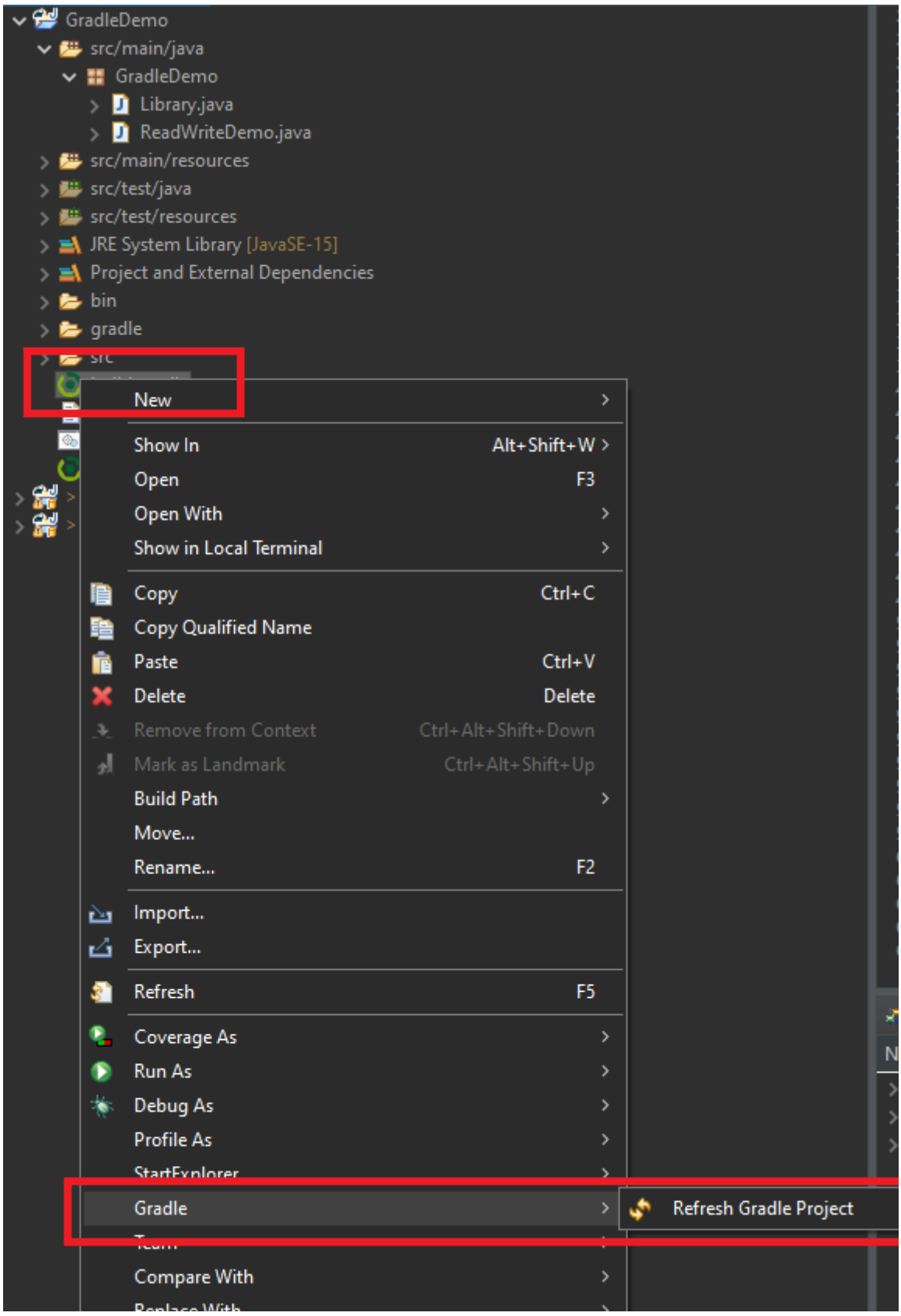
Right click on src/main/java And create a new class. Call the class ReadWriteDemo.

Now inside of your main body put.

```
// Create a new Java.io.Reader This is included in all java installs
Reader reader = Files.newBufferedReader(Paths.get("timepunches.csv"));

// Create a new com.opencsv.CSVReader This is an external library
CSVReader csvReader = new CSVReader(reader);

// Read The CSV file into list of String arrays. Each array represnts a row.
List<String[]> records = csvReader.readAll();

// Close Our readers to prevent a resource leak
csvReader.close();
reader.close();

// Create empty List of String arrays
List<String[]> newCsv = new ArrayList<String[]>();

// Add headers to the newCsv
String[] headers = {"Name,Time Worked"};
newCsv.add(headers);

// iterate through the records
int x=0;
for(String[] row:records)
{
```

```
  if(x==0)// You can use something like this to process headers
  {
    x++;
    // Handle headers here

    //
    continue; // This will skip to the next iteration of the for loop without.
  }
  // Get row Name
  String tname = row[0];
  //Get the hour and minutes into there own strings.
  String[] startTime = row[1].split(":");
  String[] stoptime = row[2].split(":");
  // Get the minutes of the day for both
  int startminutes =
Integer.parseInt(startTime[0])*60+Integer.parseInt(startTime[1]);
  int stopminutes =
Integer.parseInt(stoptime[0])*60+Integer.parseInt(startTime[1]);

  // Reformate into hh:mm and place into a String[]
  String[] tempRow = {tname,((stopminutes-startminutes)/60)+":"+
                        ((stopminutes-startminutes)%60)};
  // Add temp row into the Variable that we will use to write the new CSV
  newCsv.add(tempRow);
}


// Create Java.io.Writer
Writer writer = Files.newBufferedWriter(Paths.get("timetotal.csv"));
// Create com.opencsv.CSVWriter
CSVWriter csvWriter = new CSVWriter(writer);

// Write the newCsv variable to the file
csvWriter.writeAll(newCsv);
csvWriter.flush();
// Close to avoid resource leaks
csvWriter.close();
```

Now one more step before you execute. ***Right click*** on the ***build.gradle*** file and click ***Refresh gradle Project***



Now you can execute your ReadWriteDemo.java. After the program executes you should refresh the project and see the new csv that has been created. This should be a simple example to test Gradle and make sure that it is working.

It also provides a little insight into how useful it can be. You now can get your hands on a CSV parser that has been created to follow all standard guidelines without writing any of the code yourself.

There are multiple very helpful things that you can put into your program. Some of the ones that I wont be going over but all are helpful.

- ***Apachie POI*** - This allows you to read and wright Microsoft excel files and doc files. It is very verbose and includes support for formatting.
- ***Jackson*** - This is a package that provides more seamless support for JSON. And is incredibly helpful for web servers.
- ***Hibernate*** - This package is designed as an interface between SQL and java to allow you to better create objects from database information and much more. The idea is to make SQL more integrated into the OOP world.
- ***Java Mail*** - An emailing package built for java to send and view emails using a pre-setup mail server.