**Run Book:**

**Jar Build Steps:**

One can build the jar from scala-ide eclipse by following the below steps and we can use the fat jar generated to run the code.

(Or)

Alternatively one can use the thin jar available in target path of the git project and do spark-submit on it by providing dependencies using –packages option

**Steps To Build The jar From Scala-Ide eclipse:**

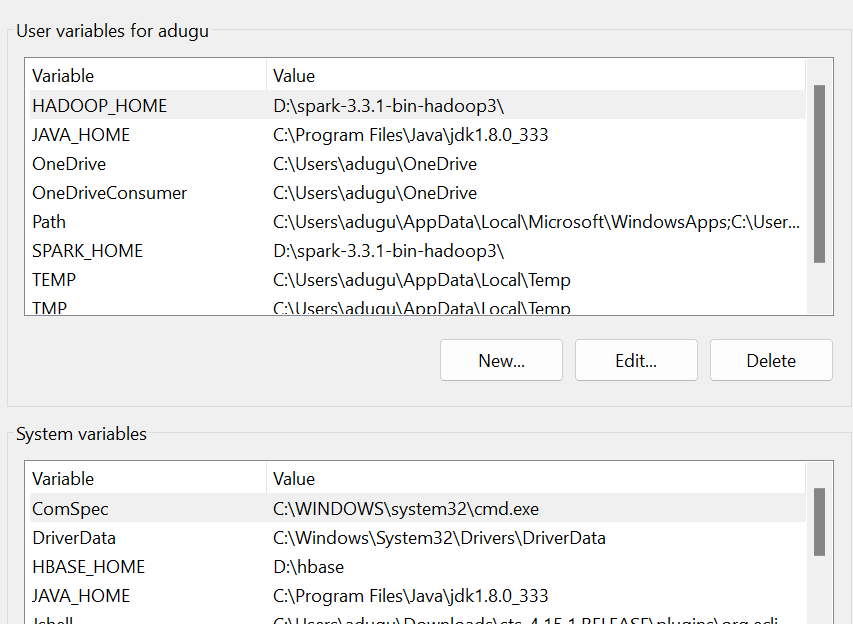
**1.Install Java8 :**

To install Apache Spark on windows, you would need Java 8 or the latest version hence download the Java version from Oracle and install it on your system. If you wanted OpenJDK you can download it from [here](https://www.oracle.com/java/technologies/downloads/#java8).

After download, double click on the downloaded .exe ([jdk-8u371-windows-x64.exe](https://www.oracle.com/java/technologies/downloads/#license-lightbox)) file in order to install it on your windows system. Choose any custom directory or keep the default location.

After installation set JAVA\_HOME environment variable.

Eg: JAVA\_HOME= C:\Program Files\Java\jdk1.8.0\_333



2.Do git clone from the path “https://github.com/ASravanKumar456/CodePractice.git”

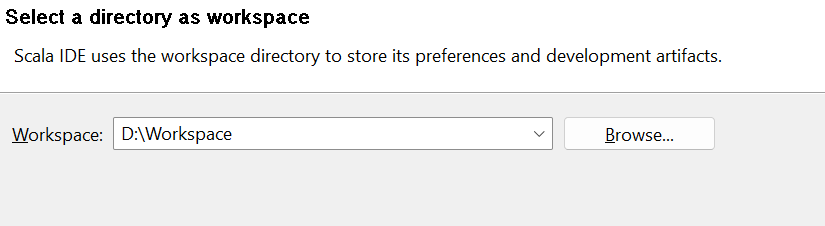
3.Install Scala-IDE for eclipse:

i. Download and install Scala-IDE for eclipse from [here](http://downloads.typesafe.com/scalaide-pack/4.7.0-vfinal-oxygen-212-20170929/scala-SDK-4.7.0-vfinal-2.12-win32.win32.x86_64.zip)

ii. Extract the zip file and run eclipse.exe.

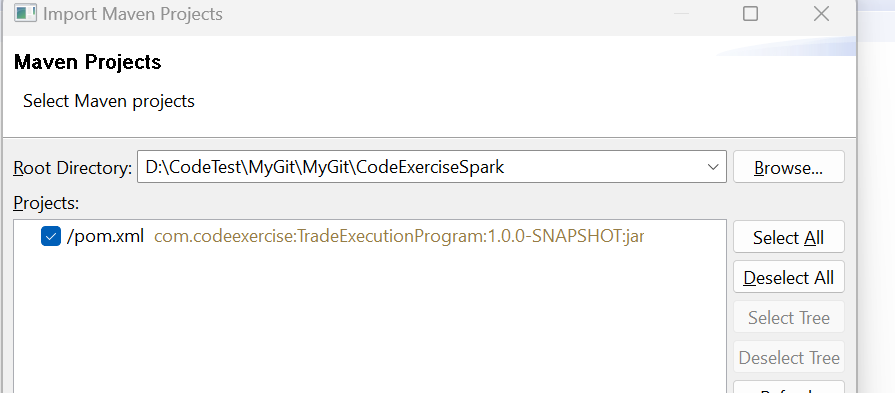
iii. Select Workspace

Eg: D:\Workspace



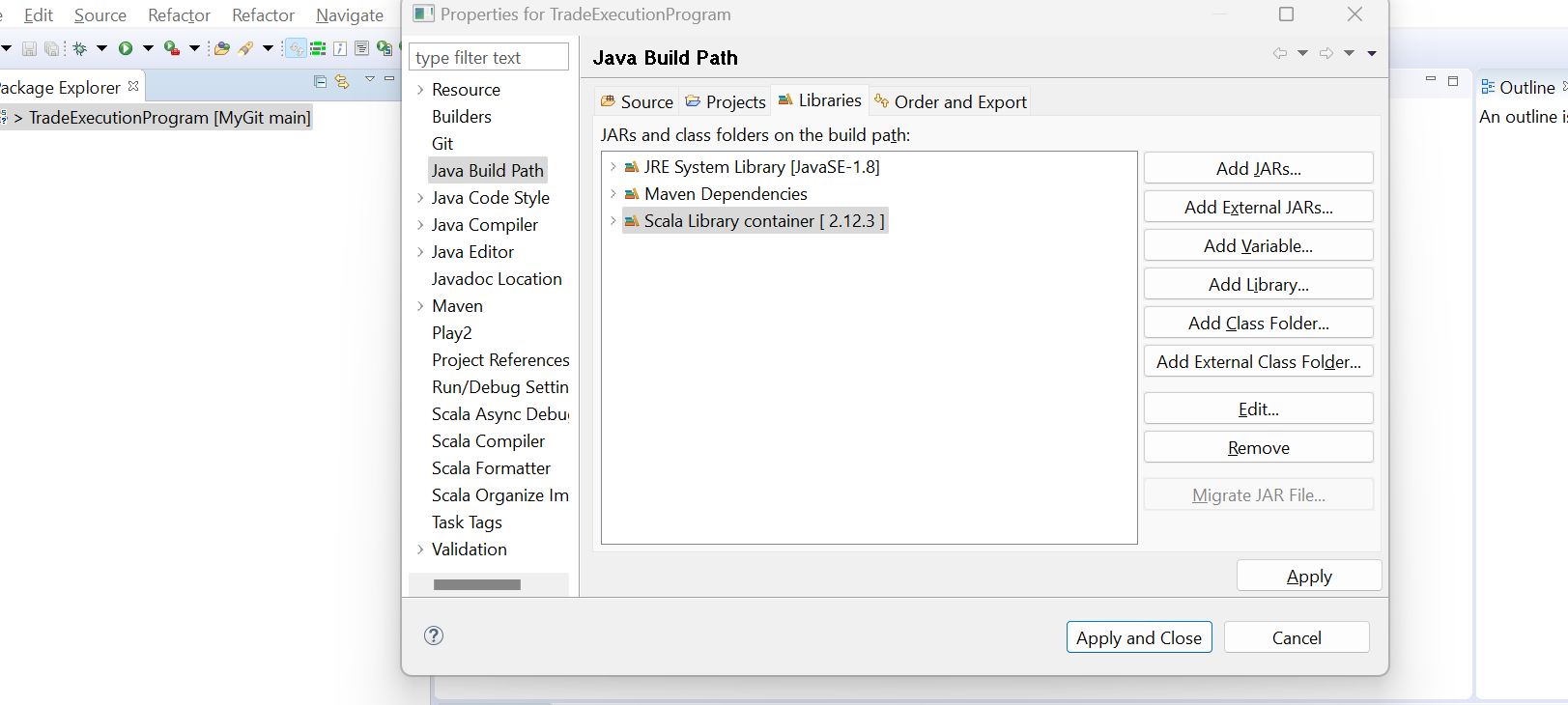
**4. Import the project from by clicking File -> Import -> Maven -> Existing Maven Project**

Then browse the project name and select the project path.

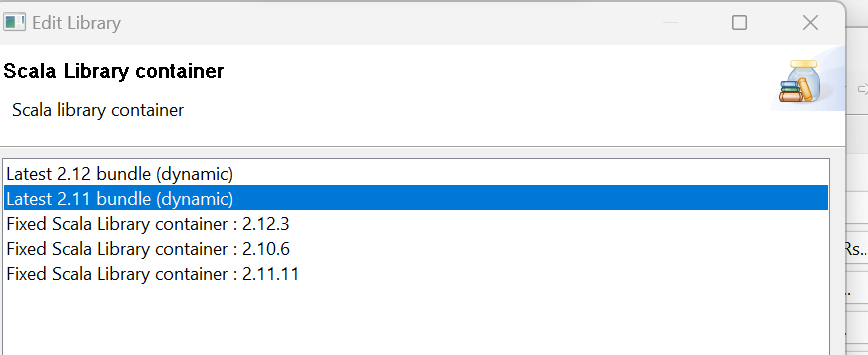


**5.Configure scala build path :**

Right Click on Project ->BuildPath -> Configure Build Path -> Libraries ->Scala Library container

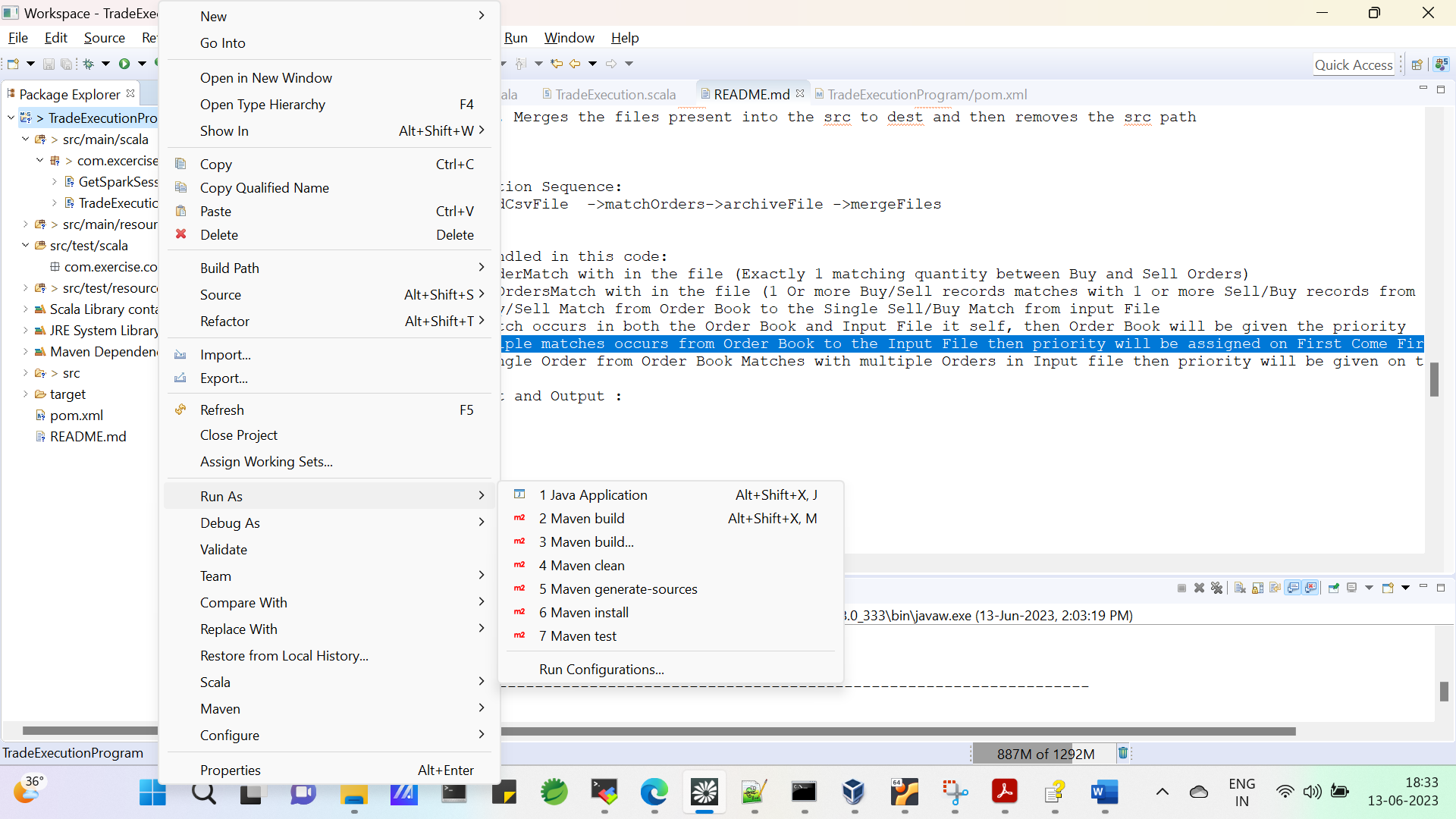


Change the scala version to Latest 2.11 dynamic then click Finish

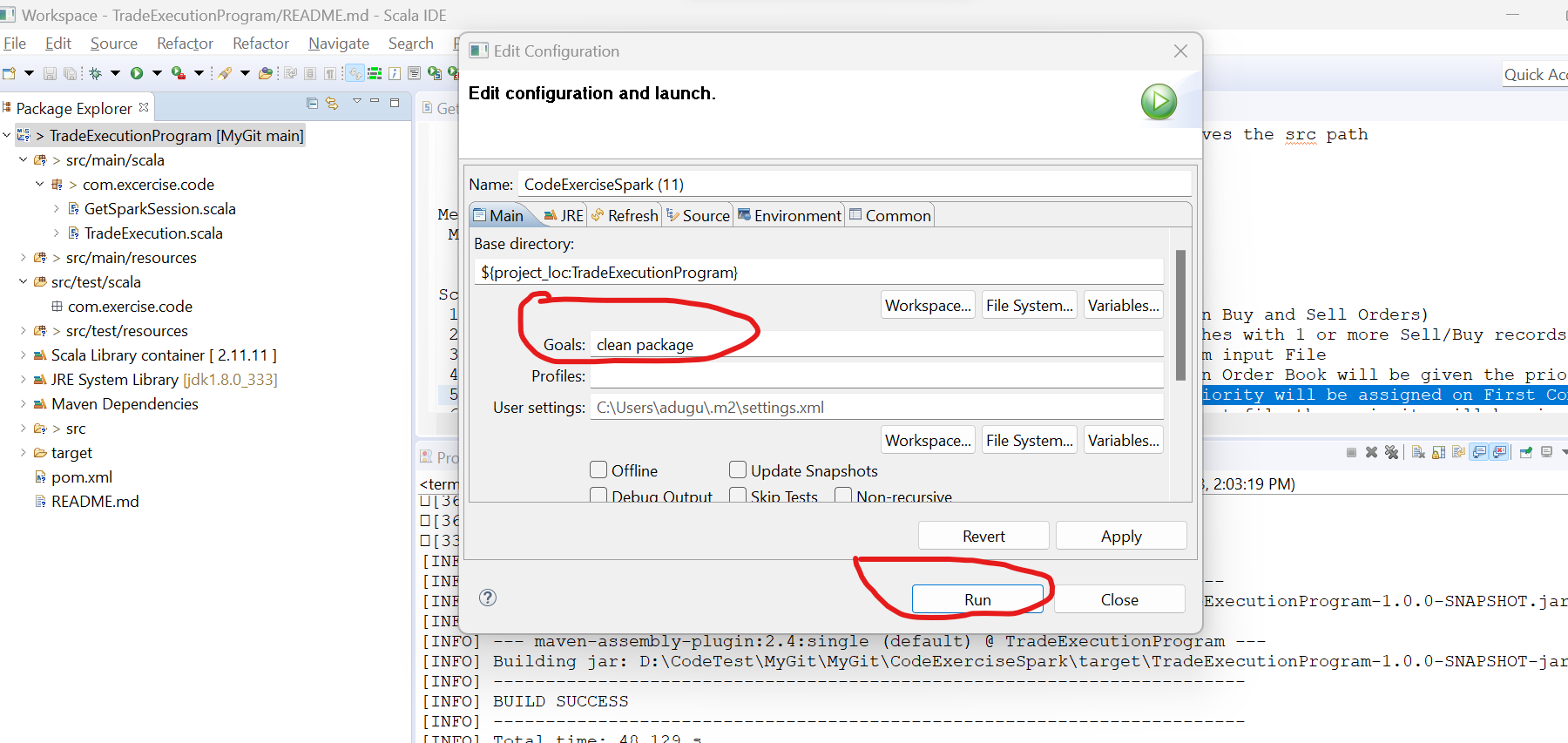


**6.Build Jar :**

Build the jar by doing right clicking on project and navigating to Run As -> 3 Maven Bulid

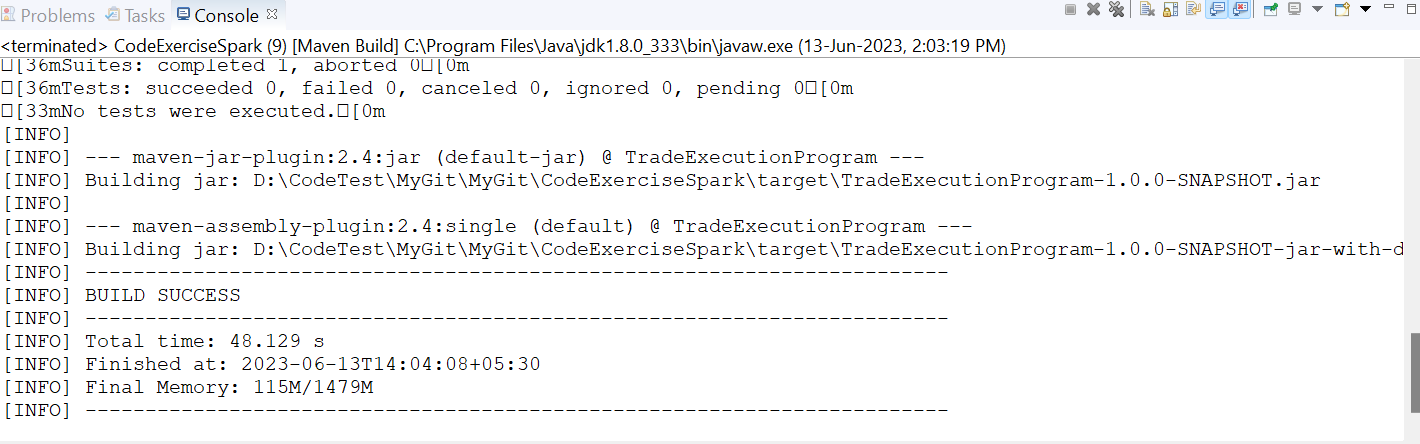


Provide **clean package** in goals section



Then click Run.

After successful jar build you should see the “BUILD SUCCESS” message below



**Steps to Run Program from Eclipse:**

1.Create the following paths which are going to be passed as input arguments to the jar

1.ordersFilePath - File path where the order files are present

Eg: D:\CodeTest\orders

2.orderBookFilePath - File path where the Booking will be Stored

Eg: D:\CodeTest\book

3.closedOrdersFilePath - Output file path where the closed/matched orders will be stored

Eg:D:\CodeTest\tradematch

4.orderFileArchivalPath - Path where the input/Order files will be archived after processing

Eg : D:\CodeTest\orders\ordersarchive

5.orderBookArchivalPath - Path where the orderBook files will be archived after processing

Eg : D:\CodeTest\orders\bookarchive

2.Place the input files from [SampleInputOutputFiles](https://github.com/ASravanKumar456/CodePractice/tree/main/SampleInputOutputFiles)present in the project to their respective paths.

exampleOrders.csv file should be placed the ordersFilePath

orderBook.csv file should be placed OrderBookFilePath

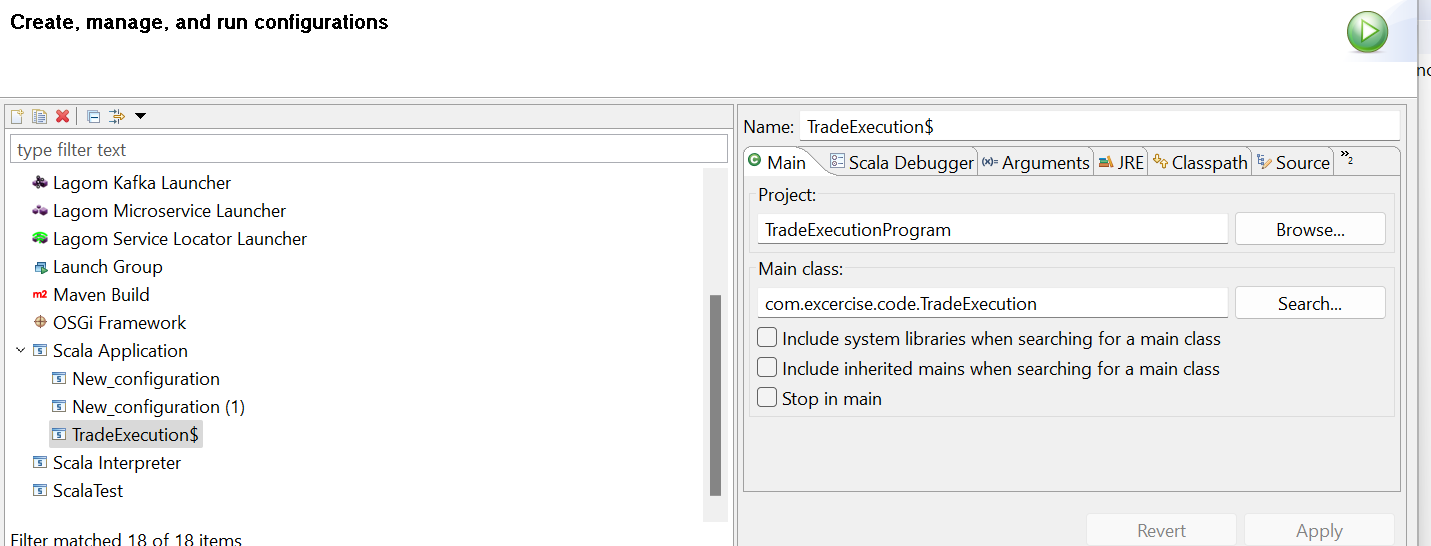
3.Setup run configurations:

1.Click on project and navigate to src/main/scala/com/exercise/code/TradeExecution.scala

2.Right click on TradeExecution.scala and do Run As -> Run Configurations -> Scala Application

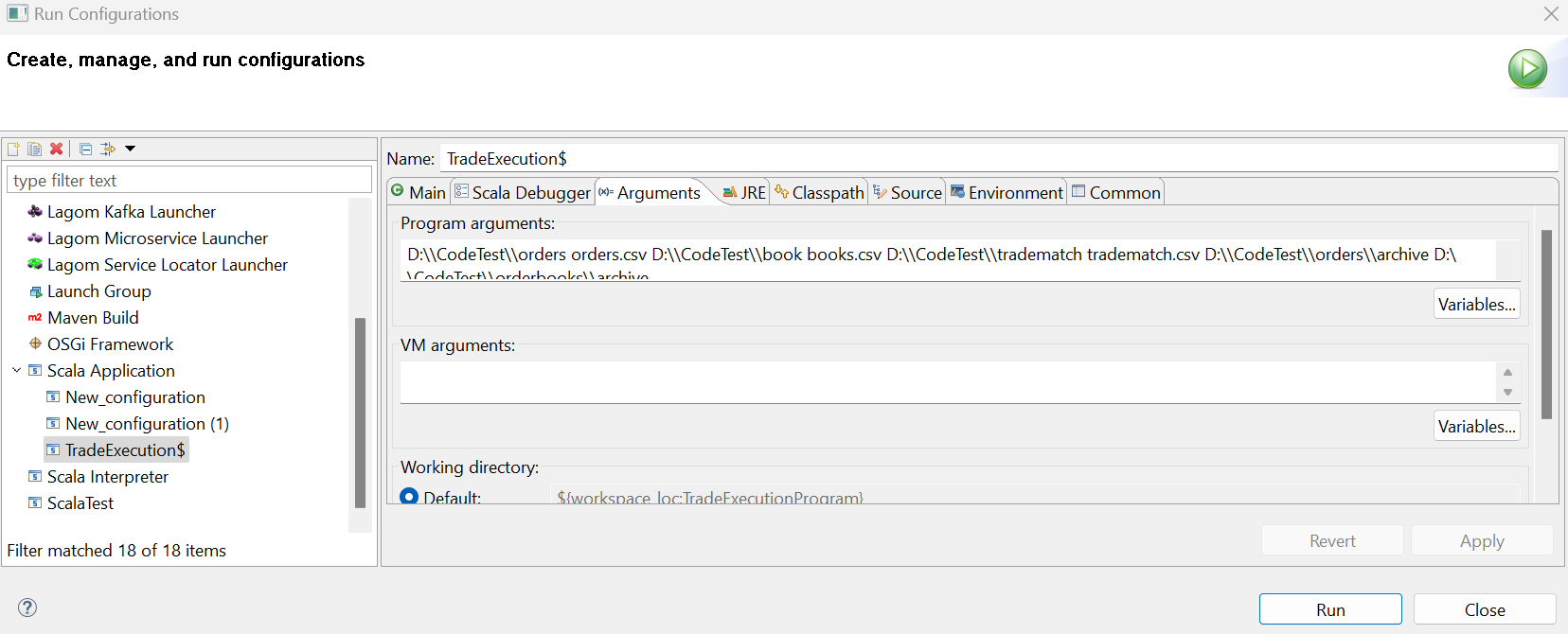
Provide The Name as **TradeExecution$**

Provide **com.exercise.code.TradeExecution** in Main Class



Click on Arguments and pass the appropriate arguments then click Run

Example: D:\\CodeTest\\orders exampleOrders.csv D:\\CodeTest\\book books.csv D:\\CodeTest\\tradematch tradematch.csv D:\\CodeTest\\orders\\archive D:\\CodeTest\\orderbooks\\archive



After successful run check the output path and archive paths.

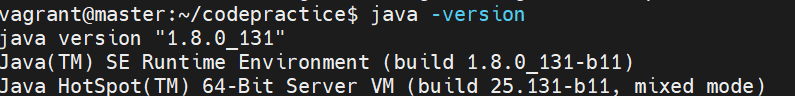
**To Run the code on Unix Environment with Hadoop Cluster:**

Pre-Requisites:

1. The Hadoop version should be greater then 2.10



1. Check Java version . It should be 1.8



1. Spark version should be greater than or equal to 2.4.3



Run Procedure:

1.Place the jar in desired path

Eg : /home/vagrant/codepractice/jars/TradeExecutionProgram-1.0.0-SNAPSHOT-jar-with-dependencies.jar\_bkp

2. Create the following folders in Hadoop

1.ordersFilePath - File path where the order files are present

Eg: hdfs dfs -mkdir /home/vagrant/codepractice/input

2.orderBookFilePath - File path where the Booking will be Stored

Eg: hdfs dfs -mkdir /home/vagrant/codepractice/orderbook

3.closedOrdersFilePath - Output file path where the closed/matched orders will be stored

Eg: hdfs dfs -mkdir /home/vagrant/codepractice/output

4.orderFileArchivalPath - Path where the input/Order files will be archived after processing

Eg : hdfs dfs -mkdir /home/vagrant/codepractice/archive/orderFiles

5.orderBookArchivalPath - Path where the orderBook files will be archived after processing

Eg: hdfs dfs -mkdir /home/vagrant/codepractice/archive/orderBooks

3.Upload the input files (orderFile , OrderBookFile(If Exists)) to their respective paths.

4.Upload the jar to desired path

5.Now run the jar with the spark-submit command.

Example:

**Sample Command To Run the uber jar:**

spark-submit –name “TradeExecEngine” –master yarn –deploy-mode cluster –driver-memory 1g –executory–memory 1g –executor-cores 4 –class com.excercise.code.TradeExecution /home/vagrant/codepractice/jars/TradeExecutionProgram-1.0.0-SNAPSHOT-jar-with-dependencies.jar /home/vagrant/codepractice/input exampleOrders.csv /home/vagrant/codepractice/orderbook orderbook.csv /home/vagrant/codepractice/output tradematch.csv /home/vagrant/codepractice/archive/orderFiles /home/vagrant/codepractice/archive/orderBooks

**Sample Command To Run the thin jar:**

spark-submit --name "TradeExecEngine" --master yarn --deploy-mode cluster --driver-memory 1g --executor-memory 1g --executor-cores 4 --packages org.scala-lang:scala-library:2.11.11,org.apache.hadoop:hadoop-common:2.10.0,org.apache.spark:spark-core\_2.11:2.4.3,org.apache.spark:spark-sql\_2.11:2.4.3 --class com.excercise.code.TradeExecution /home/vagrant/codepractice/jars/TradeExecutionProgram-1.0.0-SNAPSHOT.jar /home/vagrant/codepractice/input exampleOrders.csv /home/vagrant/codepractice/orderbook orderbook.csv /home/vagrant/codepractice/output tradematch.csv /home/vagrant/codepractice/archive/orderFiles /home/vagrant/codepractice/archive/orderBooks

5.Verify the output path after successful execution

Example : hdfs dfs -ls /home/vagrant/codepractice/output

6.Verify the archive paths to check if the files have been archived