

**CSE488: Big Data Analytics**

**[SPRING 2023]**

**All Online Lab**

**MapReduce program**

**Submitted by:**

Student ID: 2019-3-60-046

Student Name: Mohsenul Kabir Mithun

**LAB 1**

\*\*--------------------------------------------\*\*

**1. Find the Aircraft, country flying mission and target country for those missions in which time on target was less than 1000.**

Bombing\_Operations.createOrReplaceTempView('bombing\_operation')

query = """

SELECT AirCraft, ContryFlyingMission, TargetCountry

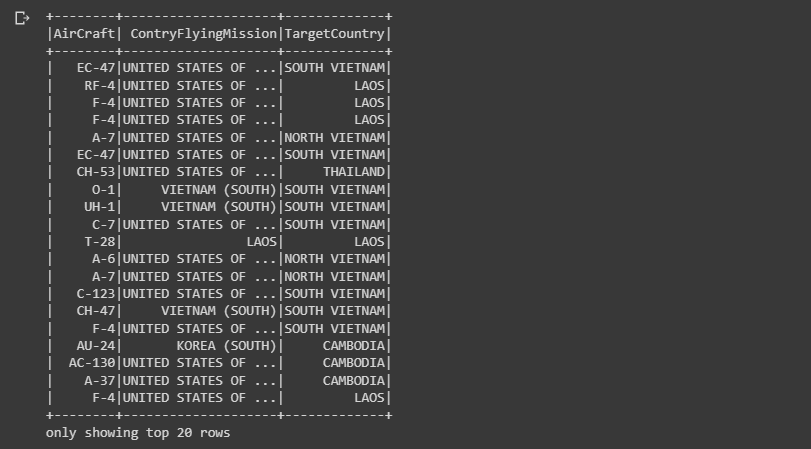
FROM bombing\_operation

WHERE TimeOnTarget < 1000

"""

result = spark.sql(query)

result.show()



**2. Find the target country and the number of times they have been attacked. Show the**

**result in descending order.**

Bombing\_Operations.createOrReplaceTempView('bombing\_operation')

query = """

SELECT TargetCountry, COUNT(TargetCountry) AS Attack

FROM bombing\_operation

GROUP BY TargetCountry

ORDER BY COUNT(TargetCountry) DESC

"""

result = spark.sql(query)

result.show()

A picture containing text

Description automatically generated

**3. Find the target country which had been attacked most.**

Bombing\_Operations.createOrReplaceTempView('bombing\_operation')

query1 = """

SELECT TargetCountry, COUNT(TargetCountry) AS NoOfAttack

FROM bombing\_operation

GROUP BY TargetCountry

ORDER BY COUNT(TargetCountry) DESC

"""

result = spark.sql(query1)

result.createOrReplaceTempView('TempResult')

query2 = """

SELECT \*

FROM TempResult

WHERE NoOfAttack = (SELECT MAX(NoOfAttack)

FROM TempResult)

"""

result = spark.sql(query2)

result.show()

Shape

Description automatically generated with medium confidence

**4. Find the country flying mission which attacked most.**

Bombing\_Operations.createOrReplaceTempView('bombing\_operation')

query1 = """

SELECT ContryFlyingMission, COUNT(ContryFlyingMission) AS NoOfFlyingMission

FROM bombing\_operation

GROUP BY ContryFlyingMission

ORDER BY COUNT(ContryFlyingMission) DESC

"""

result = spark.sql(query1)

result.createOrReplaceTempView('TempResult')

query2 = """

SELECT \*

FROM TempResult

WHERE NoOfFlyingMission = (SELECT MAX(NoOfFlyingMission)

FROM TempResult)

"""

result = spark.sql(query2)

result.show()

Text

Description automatically generated

**5. Find the take-off location and how many times they had used.**

Bombing\_Operations.createOrReplaceTempView('bombing\_operation')

query = """

SELECT TakeoffLocation, COUNT(TakeoffLocation) AS NoOfTakeoffLocation

FROM bombing\_operation

GROUP BY TakeoffLocation

"""

result = spark.sql(query)

result.show()

Text

Description automatically generated with low confidence

**6. Find the number of fighter jets from Aircraft\_Glossary.**

Aircraft\_Glossary.createOrReplaceTempView('aircraft\_glossary')

query = """

SELECT COUNT(AirCraftType) as TotalNoOfFighterJet

FROM aircraft\_glossary

WHERE AirCraftType == 'Fighter Jet'

"""

result = spark.sql(query)

result.show()

**Shape

Description automatically generated with medium confidence**

**7. Find the number of different types aircraft for each air craft type.**

Aircraft\_Glossary.createOrReplaceTempView('aircraft\_glossary')

query = """

SELECT AirCraftType, COUNT(AirCraftType) AS NoOfAirCraftType

FROM aircraft\_glossary

GROUP BY AirCraftType

"""

result = spark.sql(query)

result.show()

A picture containing text

Description automatically generated

**8. Find the aircraft name which had used most in attacks.**

Aircraft\_Glossary.createOrReplaceTempView('aircraft\_glossary')

Bombing\_Operations.createOrReplaceTempView('bombing\_operation')

query1= """

SELECT AirCraftName, TargetCountry

FROM aircraft\_glossary A, bombing\_operation B

WHERE A.AirCraft = B.AirCraft

"""

result = spark.sql(query1)

result.createOrReplaceTempView("TempResult1")

query2 = """

SELECT AirCraftName, COUNT(AirCraftName) as NoOfAirCraftName

FROM TempResult1

GROUP BY AirCraftName

"""

result = spark.sql(query2)

result.createOrReplaceTempView("TempResult2")

query3 = """

SELECT \*

FROM TempResult2

WHERE NoOfAirCraftName == (SELECT MAX(NoOfAirCraftName)

FROM TempResult2 )

"""

result = spark.sql(query3)

result.show()

Graphical user interface, text

Description automatically generated

**LAB 3**

\*\*--------------------------------------------\*\*

**Creating a Hadoop project in Eclipse: (MapReduce used for SumCalculator)**

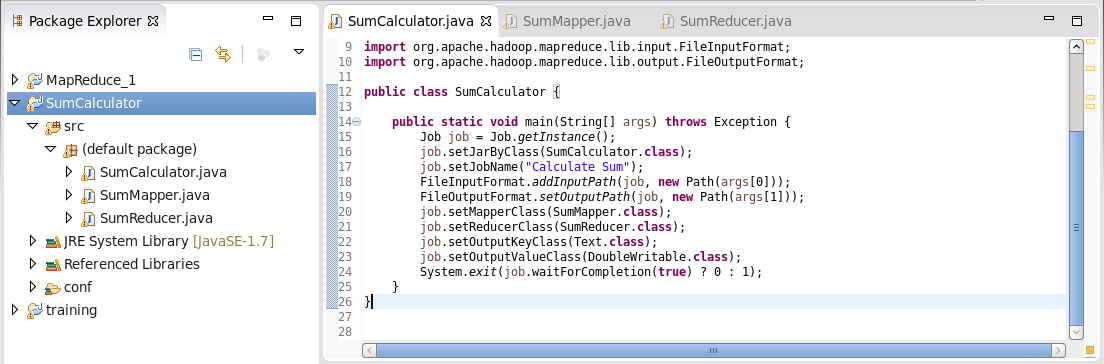
**🡪 MapReduce Program for SumCalculator:**

1. Open Eclipse IDE

2. Copy 'training' project and paste it on same place by name it "SumCalculator "

3. Expand SumCalculator 🡪 src 🡪 (default package)

4. Rename 4 of 3 java files are " SumCalculator.java", " SumReducer.java", " SumMapper.java" and replace the code that I provide bellow,



Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

5. Right click on ' SumCalculator ' project 🡪  export 🡪 java 🡪 select jar 🡪 Next 🡪 give a jar file name as "Sum.jar" 🡪 Next 🡪 Next 🡪 Finish

6. Avoid if any error message occurs.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

7. Go to http://quickstart.cloudera:8888/accounts/login/?next=/

8. Login by using Username: cloudera & Password: cloudera

9. Go to file browser that you see upper right side of the browser

10. Create new directory by click on New 🡪 Directory 🡪 provide name as ‘sum’ 🡪 go that directory create new file as file1.csv

11. Go to that file1.csv 🡪 click on 'edit file' option and add some text as coma separated. Then save it.

Graphical user interface, text, application

Description automatically generated

12. open CMD and write bellow command

hadoop jar sum.jar SumCalculator  /user/cloudera/sum/file1.txt  sumout

More generic way to explain:

🡪 hadoop jar <jar file> <main class> <input path> <output path>

Text, letter

Description automatically generated

Text

Description automatically generated

13. To see output go to ‘Web Browser’ where you create a folder and add csv file. Here you also see another folder named "sumout" In this folder you will find the output.

Table

Description automatically generated

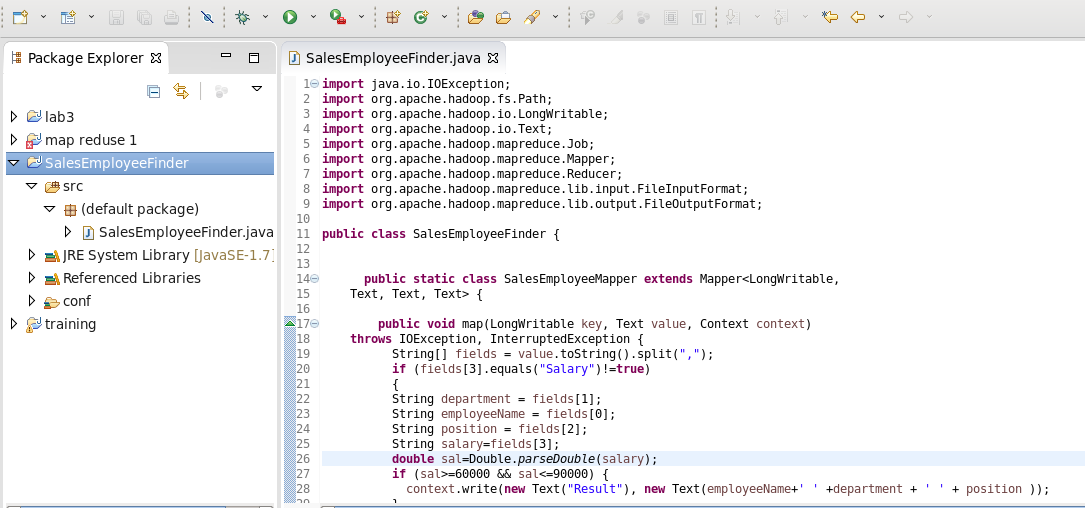
Graphical user interface, text, application

Description automatically generated

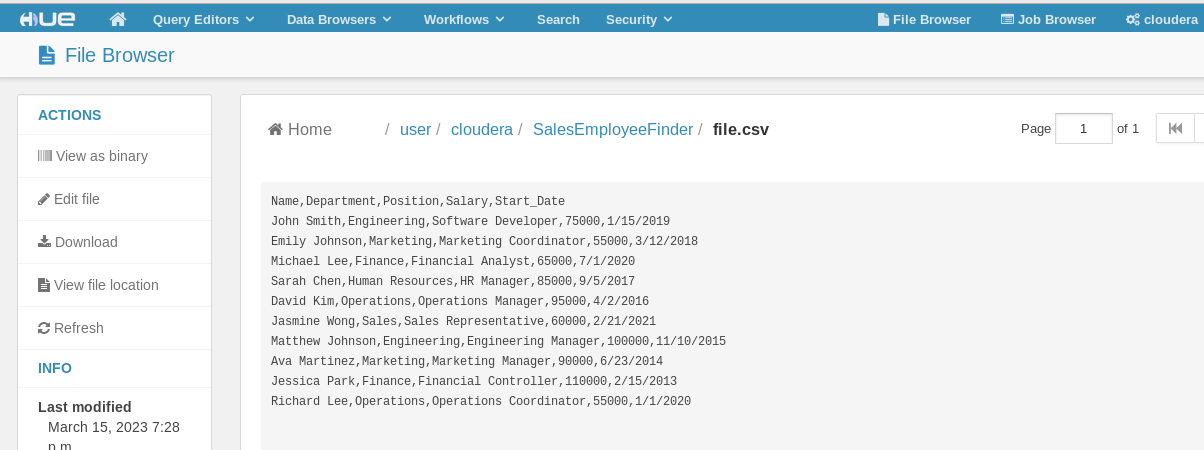
**LAB 4**

\*\*--------------------------------------------\*\*

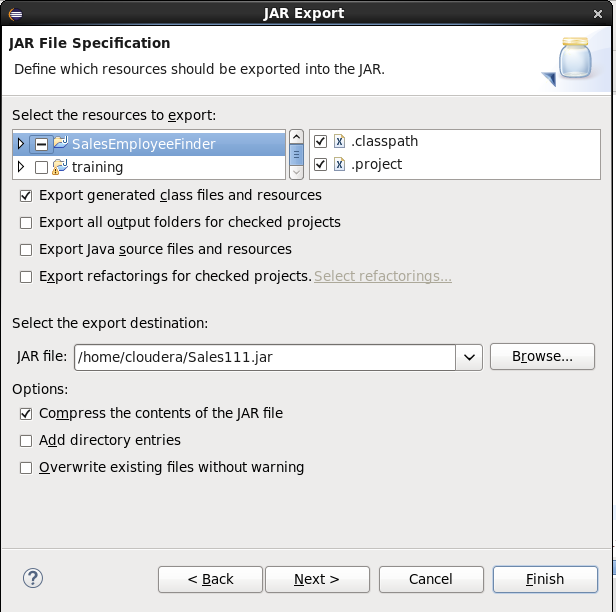
**Creating a Hadoop project in Eclipse: (MapReduce used for finding Employee)**



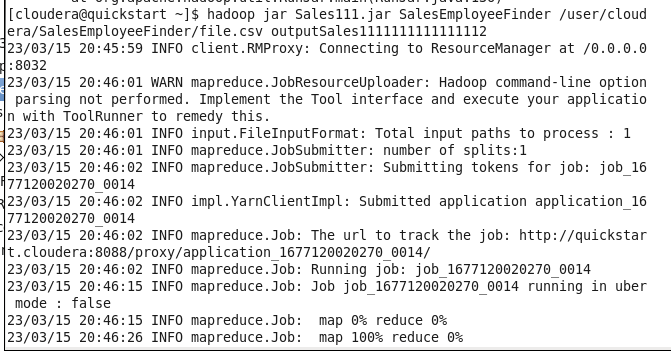
Input:

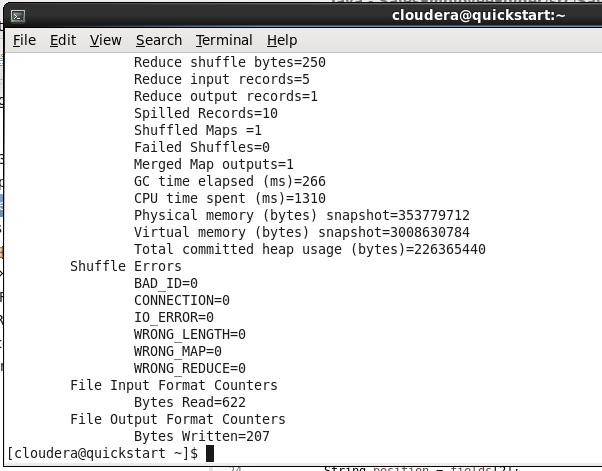


Jar Creation:

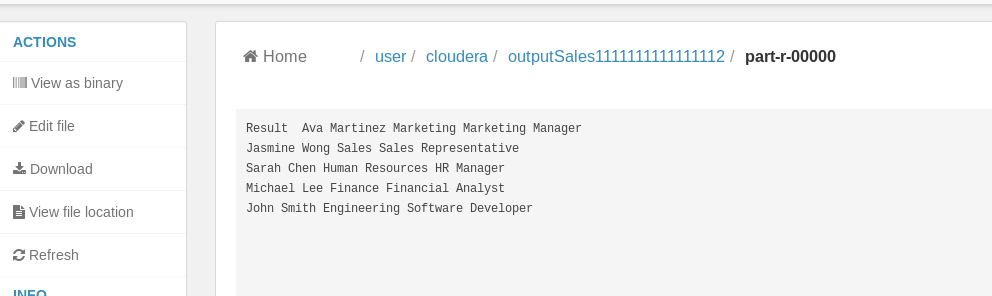


CMD run the program:





Output:



**LAB 5**

\*\*--------------------------------------------\*\*

**Creating a Hadoop project in Eclipse: (MapReduce used for ItemsetCount)**

Source Code SS: (Mapper and Reducer Class)

Graphical user interface, text

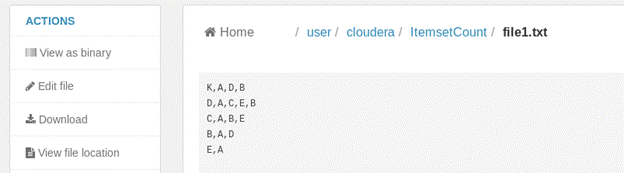
Description automatically generated

Creating Jar file:

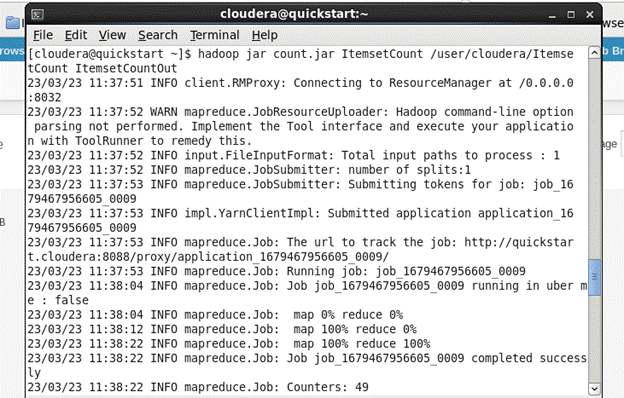
Graphical user interface, text, application

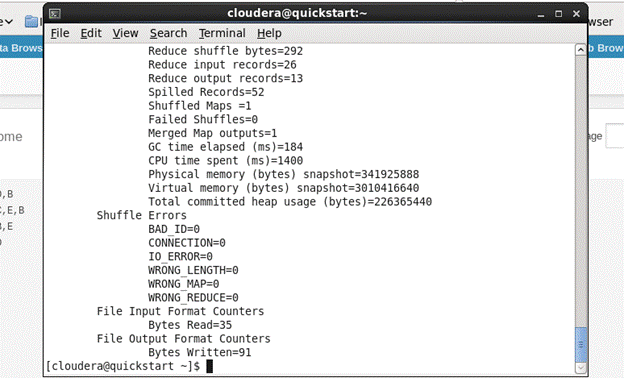
Description automatically generated

Creating Input file:



CMD run the program:





Output:

Graphical user interface

Description automatically generated with medium confidence

**LAB 6**

\*\*--------------------------------------------\*\*

**Creating a Hadoop project in Eclipse:**

**1.  Find the occurrences of each page (numbered as 1, 2, 3, and so on).**

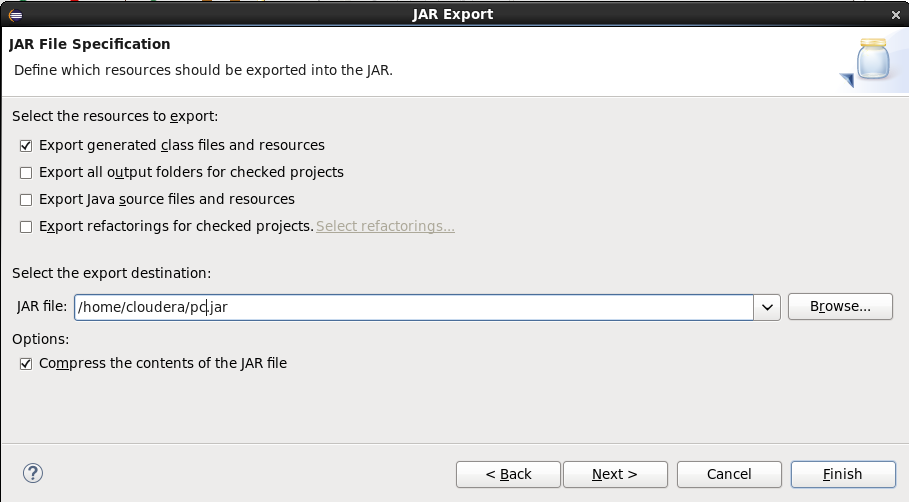
Source Code SS: (Mapper and Reducer Class)

Graphical user interface, text, application, email

Description automatically generated A picture containing text

Description automatically generated

Creating Jar file:



Creating Input file:

Graphical user interface, text, application

Description automatically generated

CMD run the program:

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Output:

Graphical user interface, text, application, email

Description automatically generated

**2.** **Find the most visited page. You may use the output file generated from Question 1 as an input file to solve this problem.**

Source Code SS: (Mapper and Reducer Class)

Graphical user interface, text, application, email

Description automatically generatedText

Description automatically generated with medium confidence

Creating Jar file:

Graphical user interface, text, application, email

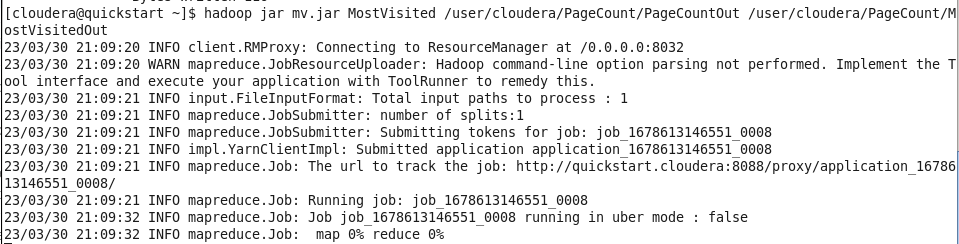
Description automatically generated

Creating Input file:

Graphical user interface, text, application

Description automatically generated

CMD run the program:



Text

Description automatically generated

Output:

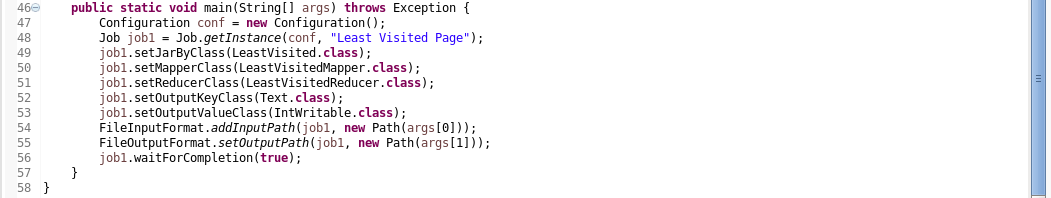
Graphical user interface, text, application, website

Description automatically generated

**3.** **Find the least visited page. You may use the output file generated from Question 1 as an input file to solve this problem.**

Source Code SS: (Mapper and Reducer Class)





Creating Jar file:

Graphical user interface, text, application, email

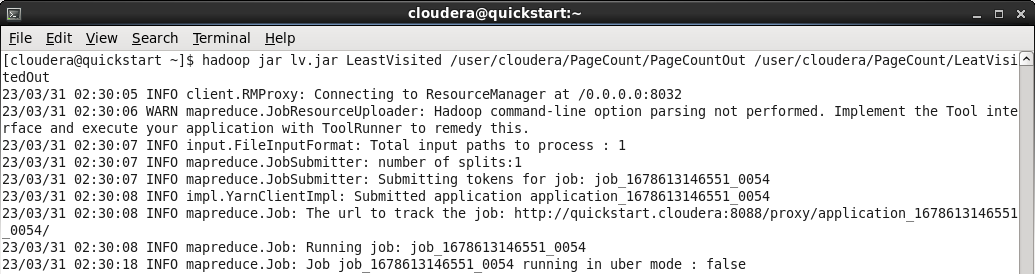
Description automatically generated

Creating Input file:

Graphical user interface, text, application

Description automatically generated

CMD run the program:



Graphical user interface, text

Description automatically generated

Output:

Graphical user interface, application, website

Description automatically generated

**4.** **Find the frequency of pairs. In this case, remove the duplicate numbers from each transaction as a data pre-processing step. Use the concept of Java set for removing duplicates.**

Source Code SS: (Mapper and Reducer Class)

Graphical user interface, text, application, email

Description automatically generated Graphical user interface

Description automatically generated with medium confidence

Creating Jar file:

Graphical user interface, text, application, email

Description automatically generated

Creating Input file:

Graphical user interface, text, application

Description automatically generated

CMD run the program:

Text

Description automatically generated

Text

Description automatically generated

Output:

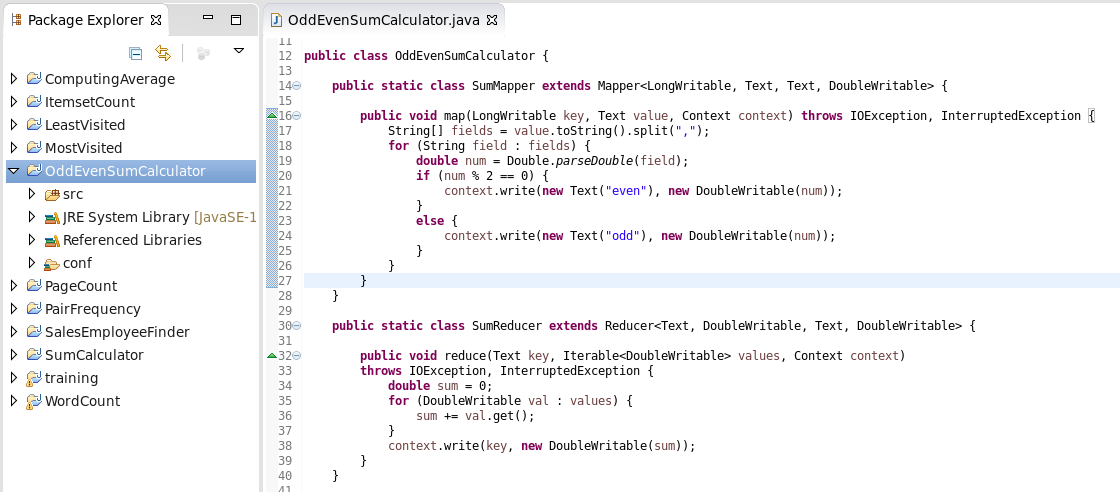
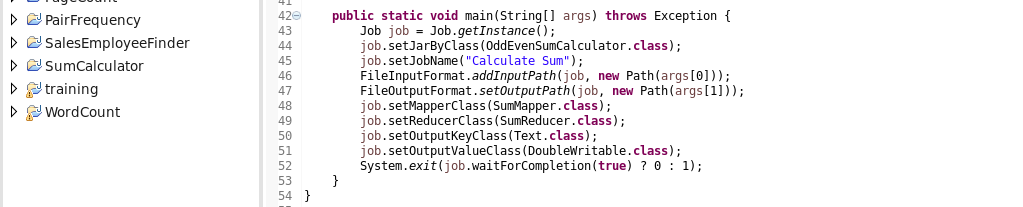


**LAB 3 Bonus Part**

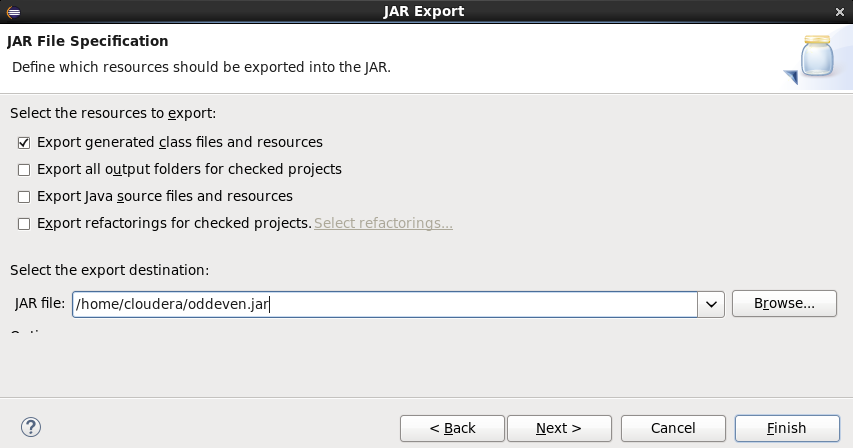
\*\*--------------------------------------------\*\*

**1. Bonus: If you are able to execute this program, then write a program of your own that generates sum of Odd and sum of even numbers from the same csv files.**

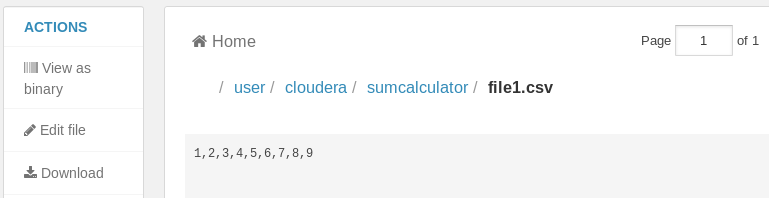
Source Code SS: (Mapper and Reducer Class)

Creating Input file:



Creating Input file:



CMD run the program:

**Text

Description automatically generated**

**Text

Description automatically generated**

Output:

Graphical user interface, text

Description automatically generated