Instruction of SBG Big Data Craft Demonstration

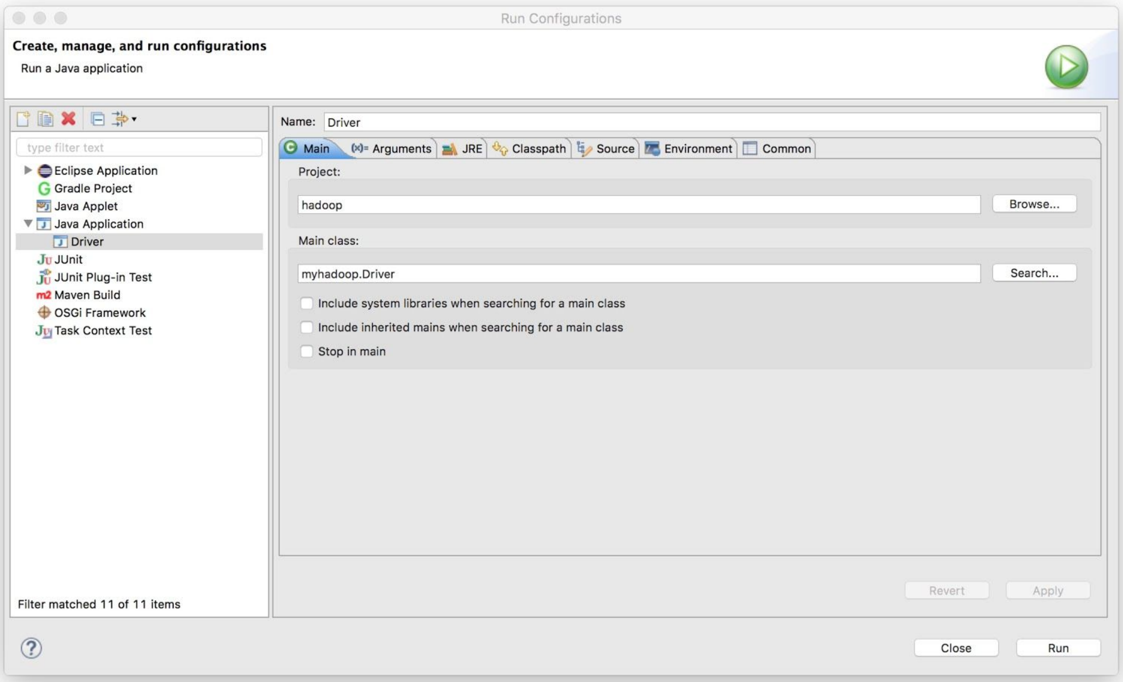
How to use the code :

Setup M2Eclipse with dependencies, or install hadoop locally for your computer.

If you use Maven repository in your Eclipse, by installing and creating a Maven project with setting dependencies, you could follow the next steps:

1. Put all of the java classes into the Maven project you created.

2. Click Run Configuration.
   Select “Java Application” and “Driver” in left column.
   In “Main” section, choose the project you want to run(‘hadoop’), and then search your main class(‘myhadoop.Driver’).
   Click ‘Apply’, then ‘Run’.



Run Configurations

Create, manage, and run configurations

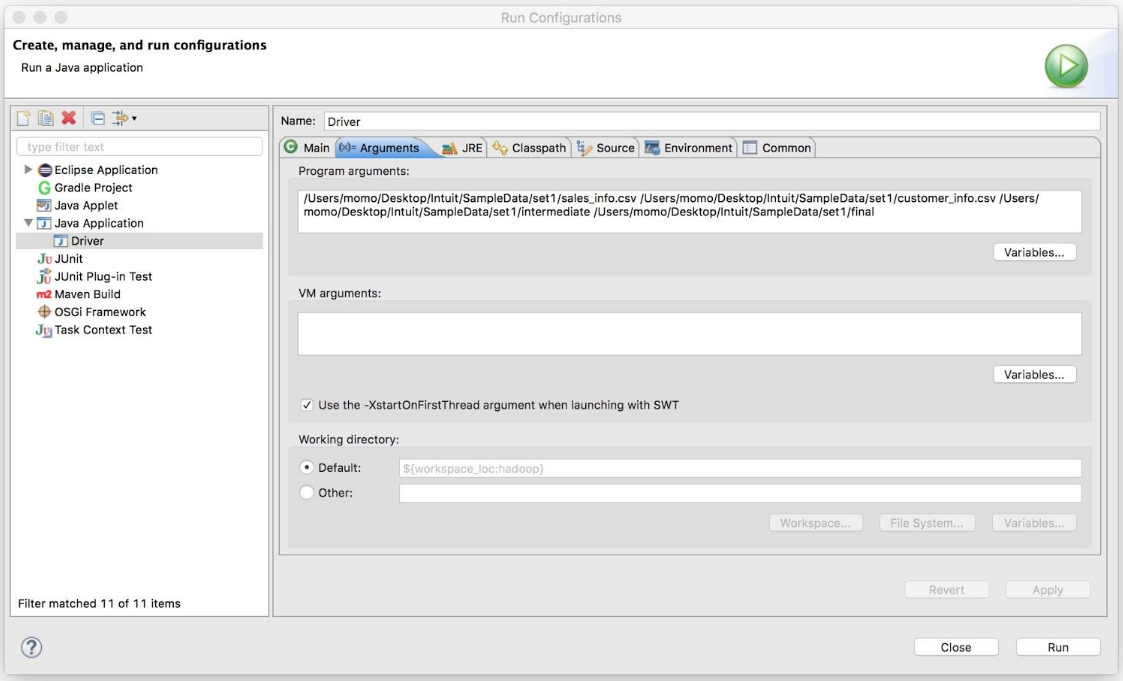
Run a Java application

Name: Driver

Project: hadoop

Main class: myhadoop.Driver

3. In “Arguments” section, add four program arguments, separate them by only one space.
   The four arguments should be absolute path of: sales information data, customer information data, intermediate result location you wish, final result location you wish.
   Click ‘Apply’, then ‘Run’.



Name: Driver

Program arguments:

```
/Users/momo/Desktop/Intuit/SampleData/set1/sales_info.csv /Users/momo/Desktop/Intuit/SampleData/set1/customer_info.csv /Users/
momo/Desktop/Intuit/SampleData/set1/intermediate /Users/momo/Desktop/Intuit/SampleData/set1/final
```

VM arguments:

✓ Use the -XstartOnFirstThread argument when launching with SWT

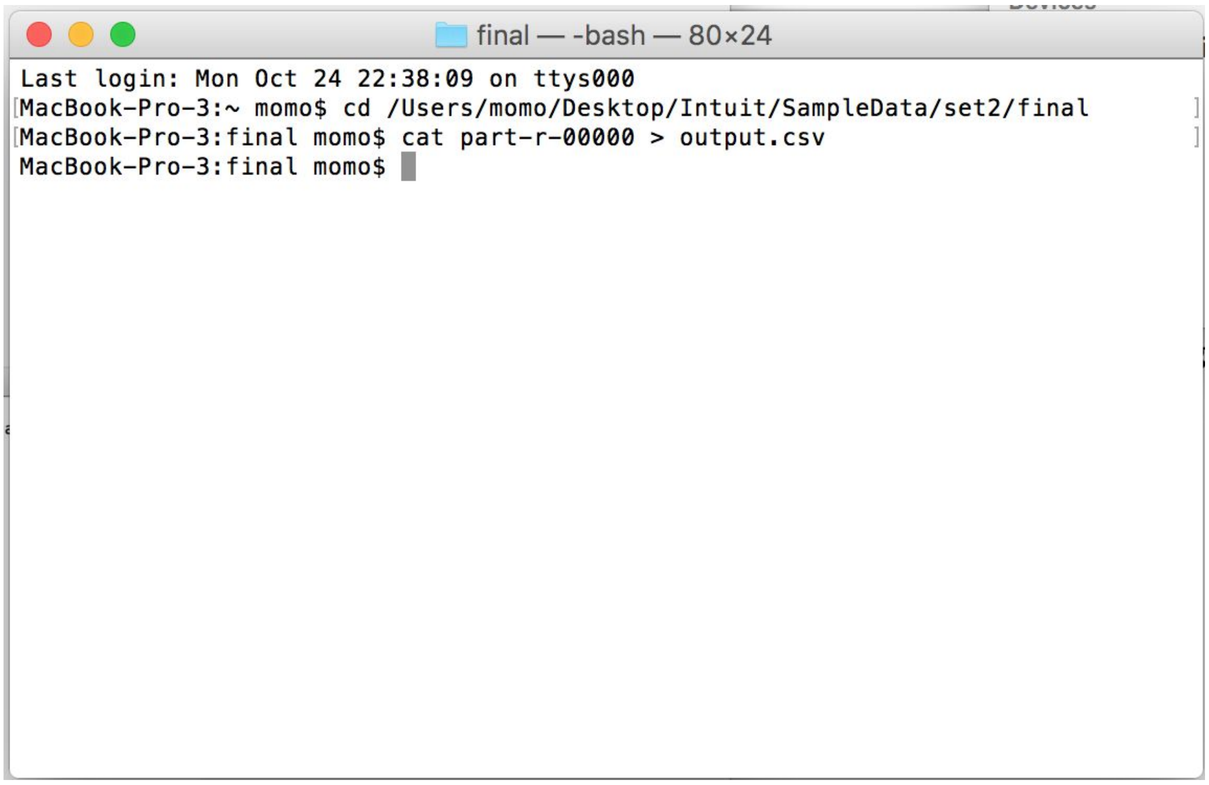
Working directory:

● Default: ${workspace_loc:hadoop}

○ Other:

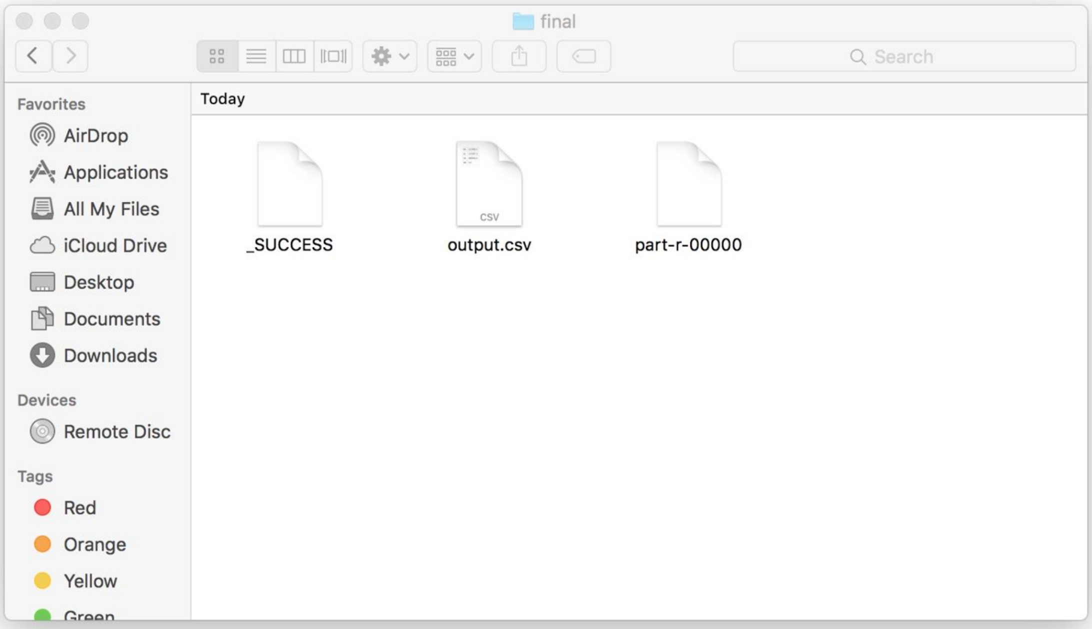
You could see the intermediate result and final result in the location you filled in program arguments.

Open your terminal, cd into final folder.
Then type ‘cat part-r-00000 > output.csv’.



```
Last login: Mon Oct 24 22:38:09 on ttys000
MacBook-Pro-3:~ momo$ cd /Users/momo/Desktop/Intuit/SampleData/set2/final
MacBook-Pro-3:final momo$ cat part-r-00000 > output.csv
MacBook-Pro-3:final momo$ 
```

Then a new file ‘output.csv’ would come out in your final folder. This is the final result.



Design (classes) :

1. CustomerIdKey : implements WritableComparable<CustomerIdKey>

- Use customer ID as a foreign key to manipulate map and reduce.
- Encapsulate customer ID information into this class.
- Set flag to distinguish which table current data comes from.
- This class is the output in mapper, input in reducer.

Constructor parameter:

int customerId
IntWritable recordType

Fields:

IntWritable cusotmerId
IntWritable recordType

Methods:

void write(DataOutput out)
void readFields(DataInput in)
int compareTo(CustomerIdKey o)

2. CustomerRecord : implements Writable

- Encapsulation of customer information, state as an input.

- Used as an output value of customer mapper after further encapsulation by JoinGenericWritable.
- Decapsulation in JoinReducer, then being recorded in intermediate result.

Constructor parameter:

String state

Fields:

Text state

Methods:

void write(DataOutput out)
void readFields(DataInput in)

3. SalesInfoDataRecord : implements Writable

- Encapsulation of sales information, sales amount as an input.
- Used as an output value of salesInfo mapper after further encapsulation by JoinGenericWritable.
- Decapsulation in JoinReducer, then being recorded in intermediate result.

Constructor parameter:

double sales

Fields:

DoubleWritable sales

Methods:

void write(DataOutput out)
void readFields(DataInput in)

4. JoinGenericWritable : extends GenericWritable

- A wrapper for Writable instances.
- Use as mapper output value, and reducer input value.

Constructor parameter:

Writable instance

Fields:

static Class<? extends Writable>[] CLASSES

Methods:

Class<? extends Writable>[] getTypes()

5.CustomerMapper: extends Mapper<LongWritable,Text,CustomerIdKey, JoinGenericWritable>

- Mapper for customer_info table.
- Input pair: LongWritable-no use, Text-raw data of this id
- Output pair: CustomerIdKey-encapsulated customer info, JoinGenericWritable-double encapsulated state info
- Next step: JoinReducer.java

Methods:

@override
map(LongWritable key, Text value, Mapper<LongWritable, Text, CustomerIdKey, JoinGenericWritable>.Context context)

6.SalesInfoDataMapper :

extends Mapper<LongWritable, Text, CustomerIdKey, JoinGenericWritable>

- Mapper for sales_info table.
- Input pair: LongWritable-no use, Text-raw data of this id
- Output pair: CustomerIdKey-encapsulated customer info, JoinGenericWritable-double encapsulated sales info
- Next step: JoinReducer.java

Methods:

@override
map(LongWritable key, Text value, Mapper<LongWritable, Text, CustomerIdKey, JoinGenericWritable>.Context context)

7. JoinReducer : extends Reducer<CustomerIdKey, JoinGenericWritable, NullWritable, Text>

- First Reducer. Receive from CustomerMapper and SalesInfoDataMapper, integrate by customerId.
- Input pair: CustomerIdKey-from two mappers, JoinGenericWritable-other information(including states and sales).
- Output pair: NullWritable-no keys needed, Text-states and sales information each customerId.
- Be prepared for next Job: integrate by states.
- Next step: StateMapper.java

Methods:

@Override
reduce(CustomerIdKey key, Iterable<JoinGenericWritable> values, Reducer<CustomerIdKey, JoinGenericWritable, NullWritable, Text>.Context context)

8. StateMapper : extends Mapper<Object, Text, Text, DoubleWritable>

- Mapper for last Job, group data by states.
- Input pair: Object-no use, Text-output Text from JoinReducer.
- Output pair: Text-states, DoubleWritable-sales.
- Next step: StateReducer.java

Methods:

@override
map(Object key, Text value, Mapper<Object, Text, Text, DoubleWritable>.Context context)

9. StateReducer : extends Reducer<Text, DoubleWritable, Text, DoubleWritable>

- Second Reducer. Receive from StateMapper, integrate by state.
- Input pair: Text-states, DoubleWritable-sales.
- Output pair: Text-states, DoubleWritable-sales.
- Final output.

Methods:

@Override
reduce(Text key, Iterable<DoubleWritable> values, Reducer<Text, DoubleWritable, Text, DoubleWritable>.Context context)

10. JoinGroupComparator : extends WritableComparator

- Group comparator, with CustomerIdKey passing in, group by customerId.
- Put same customerId together.
- Used in Job class.

Constructor parameter:

CustomerIdKey.class

Methods:

@Override
public int compare(WritableComparable a, WritableComparable b)

11. JoinSortingComparator : extends WritableComparator

- Sort comparator, with CustomerIdKey passing in, sort by customerId.
- Used in Job class.

Constructor parameter:

CustomerIdKey.class

Methods:

@Override
public int compare(WritableComparable a, WritableComparable b)

12. Driver : extends Configured implements Tool

- Entrance of the program.
- Chaining jobs.
- Set two jobs: first job is used to joined sales by customer names, second job is to join sales by state

Methods:

int run(String[] allArgs)
static void main(String[] args)