

## Problem Statement:

Execute Map Reduce program for the weather forecasting data.

Here, we will write a Map-Reduce program for analyzing weather datasets to understand its data processing programming model. Weather sensors are collecting weather information across the globe in a large volume of log data. This weather data is semi-structured and record-oriented.

This data is stored in a line-oriented ASCII format, where each row represents a single record. Each row has lots of fields like longitude, latitude, daily max-min temperature, daily average temperature, etc. for easiness; we will focus on the main element, i.e. temperature. We will use the data from the National Centers for Environmental Information (NCEI). It has a massive amount of historical weather data that we can use for our data analysis.

### Step 1:

We can download the dataset for various cities in different years. Choose the year of your choice and select any one of the data text-file for analyzing. In my case, I have selected CRND0103-2020-AK\_Fairbanks\_11\_NE.txt dataset for analysis of hot and cold days in Fairbanks, Alaska.

We can get information about data from README.txt file available on the NCEI website.

### Step 2:

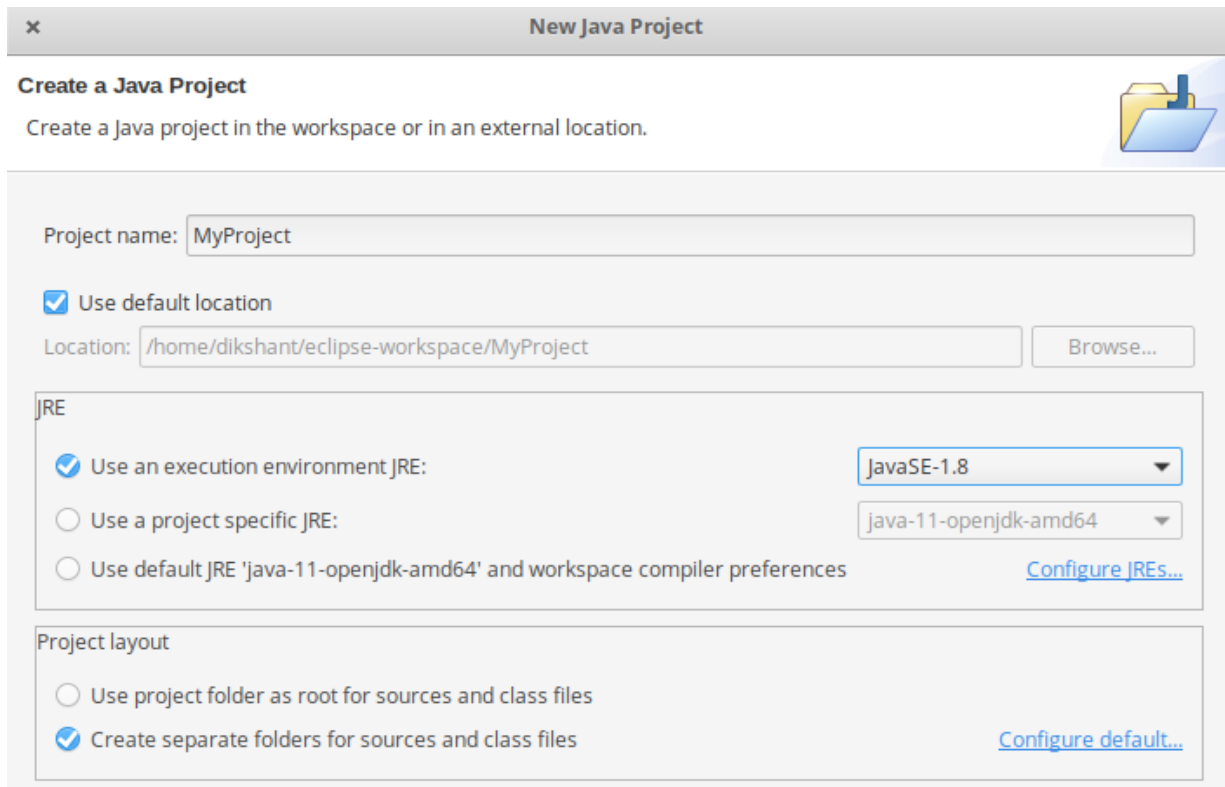
Below is the example of our dataset where column 6 and column 7 is showing Maximum and Minimum temperature, respectively.

Col. 6: Max. Temp.						Col. 7: Min. Temp.							
26494	20200101	2.424	-147.51	64.97	-18.8	-21.8	-20.3	-19.8	2.5	0.00 C	-17.9	-22.9	-19.5
81.1	72.9	77.9	-99.000	-99.000	-99.000	-99.000	-99.000	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
26494	20200102	2.424	-147.51	64.97	-19.1	-23.4	-21.3	-21.2	0.0	0.00 C	-19.4	-27.6	-22.5
78.5	73.1	76.2	-99.000	-99.000	-99.000	-99.000	-99.000	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
26494	20200103	2.424	-147.51	64.97	-19.0	-25.4	-22.2	-22.1	0.2	0.00 C	-18.4	-33.3	-28.4
79.6	65.2	75.4	-99.000	-99.000	-99.000	-99.000	-99.000	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
26494	20200104	2.424	-147.51	64.97	-18.4	-26.8	-22.6	-23.2	0.0	0.00 C	-22.8	-34.1	-28.5

### Step 3:

Make a project in Eclipse with below steps:

- First Open Eclipse -> then select File -> New -> Java Project -> Name it MyProject -> then select use an execution environment -> choose JavaSE-1.8 then next -> Finish.



**New Java Project**

Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location

Location:  [Browse...](#)

**JRE**

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE 'java-11-openjdk-amd64' and workspace compiler preferences [Configure JREs...](#)

**Project layout**

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

- In this Project Create Java class with name MyMaxMin -> then click Finish

**New Java Class**

**Java Class**

⚠ The use of the default package is discouraged.

Source folder:

Package:

☐ Enclosing type:

---

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

- Copy the below source code to this MyMaxMin java class

## JAVA Source Code

```
// importing Libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
```

```

import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;

public class MyMaxMin {

    // Mapper

    /*MaxTemperatureMapper class is static
    * and extends Mapper abstract class
    * having four Hadoop generics type
    * LongWritable, Text, Text, Text.
    */

    public static class MaxTemperatureMapper extends
        Mapper<LongWritable, Text, Text, Text> {

        /**
        * @method map
        * This method takes the input as a text data type.
        * Now leaving the first five tokens, it takes
        * 6th token is taken as temp_max and
        * 7th token is taken as temp_min. Now
        * temp_max > 30 and temp_min < 15 are
        * passed to the reducer.
        */

        // the data in our data set with
        // this value is inconsistent data
        public static final int MISSING = 9999;

        @Override
        public void map(LongWritable arg0, Text Value, Context context)
            throws IOException, InterruptedException {

            // Convert the single row(Record) to
            // String and store it in String
            // variable name line

            String line = Value.toString();

            // Check for the empty line
            if (!(line.length() == 0)) {

                // from character 6 to 14 we have
                // the date in our dataset
                String date = line.substring(6, 14);

                // similarly we have taken the maximum
                // temperature from 39 to 45 characters
                float temp_Max = Float.parseFloat(line.substring(39,

```

```

45).trim());

        // similarly we have taken the minimum
        // temperature from 47 to 53 characters

        float temp_Min = Float.parseFloat(line.substring(47,
53).trim());

        // if maximum temperature is
        // greater than 30, it is a hot day
        if (temp_Max > 30.0) {

            // Hot day
            context.write(new Text("The Day is Hot Day :" +
date),
                        new
Text(String.valueOf(temp_Max)));
        }

        // if the minimum temperature is
        // less than 15, it is a cold day
        if (temp_Min < 15) {

            // Cold day
            context.write(new Text("The Day is Cold Day :" +
date),
                        new Text(String.valueOf(temp_Min)));
        }
    }
}

// Reducer

/*MaxTemperatureReducer class is static
and extends Reducer abstract class
having four Hadoop generics type
Text, Text, Text, Text.
*/

public static class MaxTemperatureReducer extends
    Reducer<Text, Text, Text, Text> {

    /**
     * @method reduce
     * This method takes the input as key and
     * list of values pair from the mapper,
     * it does aggregation based on keys and
     * produces the final context.
     */

```

```

        public void reduce(Text Key, Iterator<Text> Values, Context
context)
            throws IOException, InterruptedException {

            // putting all the values in
            // temperature variable of type String
            String temperature = Values.next().toString();
            context.write(Key, new Text(temperature));
        }

    }

    /**
     * @method main
     * This method is used for setting
     * all the configuration properties.
     * It acts as a driver for map-reduce
     * code.
     */

    public static void main(String[] args) throws Exception {

        // reads the default configuration of the
        // cluster from the configuration XML files
        Configuration conf = new Configuration();

        // Initializing the job with the
        // default configuration of the cluster
        Job job = new Job(conf, "weather example");

        // Assigning the driver class name
        job.setJarByClass(MyMaxMin.class);

        // Key type coming out of mapper
        job.setMapOutputKeyClass(Text.class);

        // value type coming out of mapper
        job.setMapOutputValueClass(Text.class);

        // Defining the mapper class name
        job.setMapperClass(MaxTemperatureMapper.class);

        // Defining the reducer class name
        job.setReducerClass(MaxTemperatureReducer.class);

        // Defining input Format class which is
        // responsible to parse the dataset
        // into a key value pair
        job.setInputFormatClass(TextInputFormat.class);
    }

```

```

        // Defining output Format class which is
        // responsible to parse the dataset
        // into a key value pair
        job.setOutputFormatClass(TextOutputFormat.class);

        // setting the second argument
        // as a path in a path variable
        Path outputPath = new Path(args[1]);

        // Configuring the input path
        // from the filesystem into the job
        FileInputFormat.addInputPath(job, new Path(args[0]));

        // Configuring the output path from
        // the filesystem into the job
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        // deleting the context path automatically
        // from hdfs so that we don't have
        // to delete it explicitly
        outputPath.getFileSystem(conf).delete(outputPath);

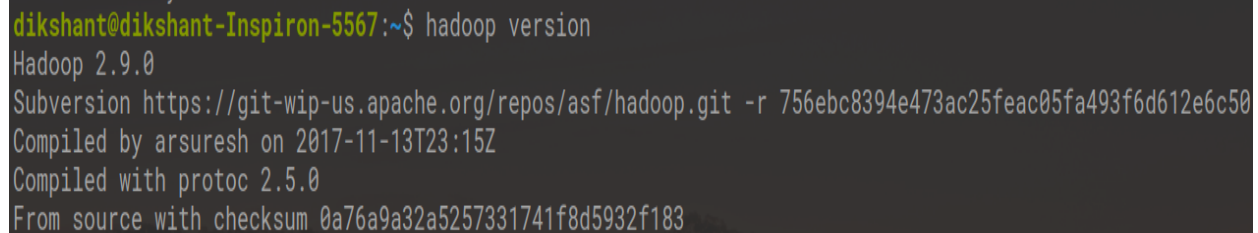
        // exiting the job only if the
        // flag value becomes false
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}

```

- Now we need to add external jar for the packages that we have import. Download the jar package Hadoop Common and Hadoop MapReduce Core according to your Hadoop version.

You can check Hadoop Version:

hadoop version



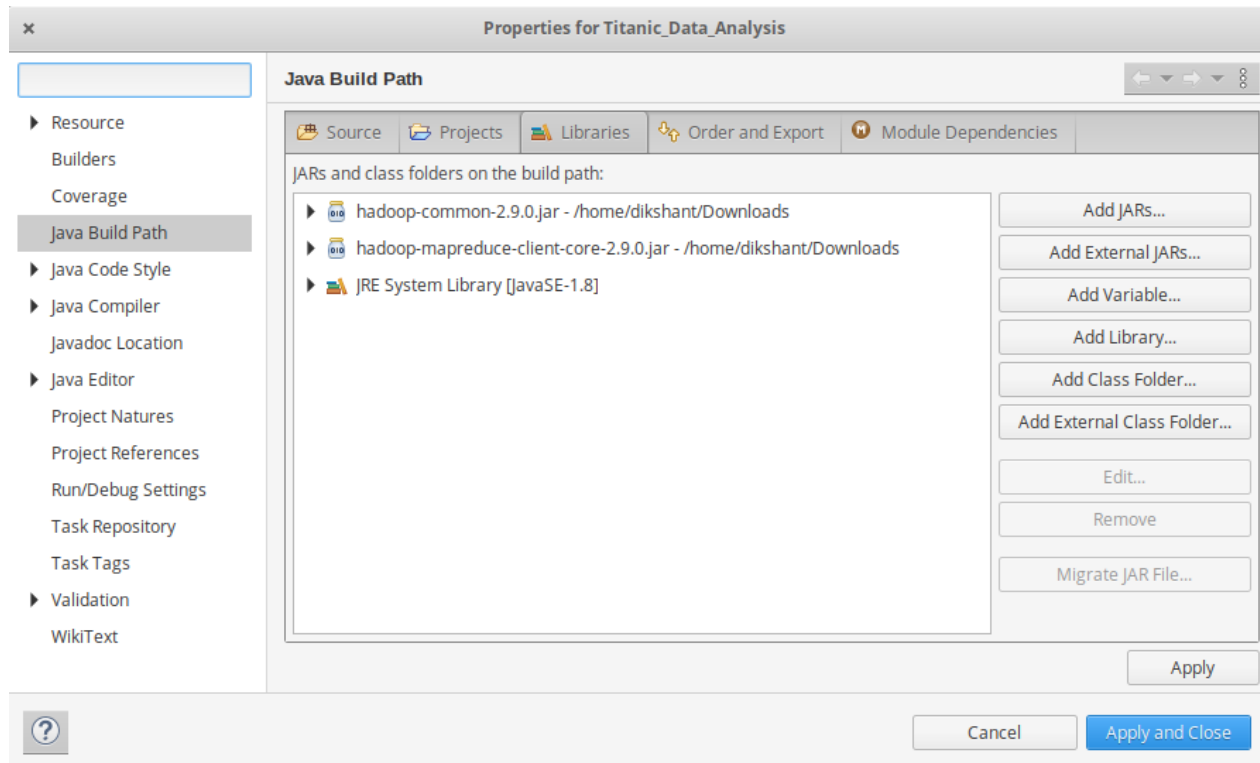
```

dikshant@dikshant-Inspiron-5567:~$ hadoop version
Hadoop 2.9.0
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r 756ebc8394e473ac25feac05fa493f6d612e6c50
Compiled by arsureh on 2017-11-13T23:15Z
Compiled with protoc 2.5.0
From source with checksum 0a76a9a32a5257331741f8d5932f183

```

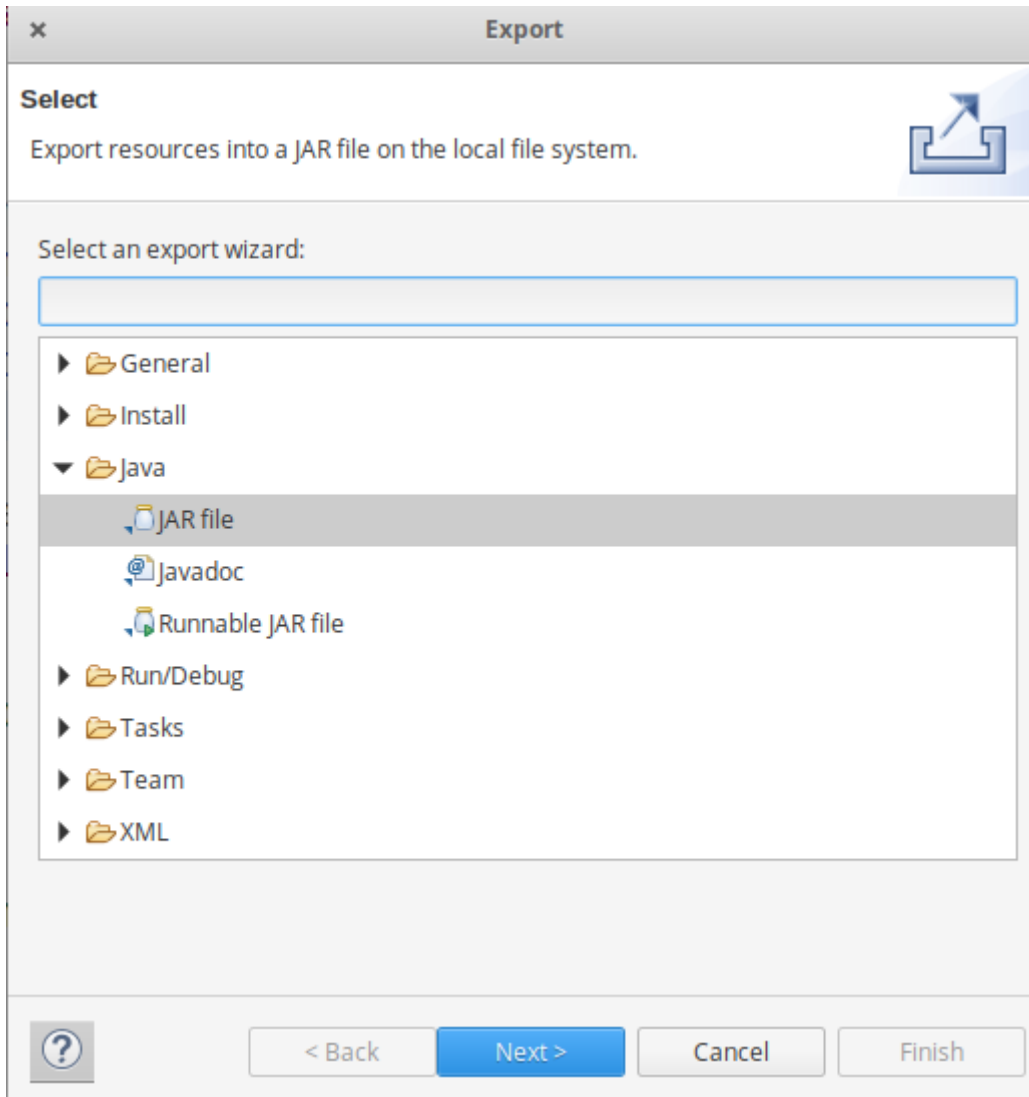
Now we add these external jars to our MyProject. Right Click on MyProject -> then select Build Path-> Click on Configure Build Path and select Add External jars.... and

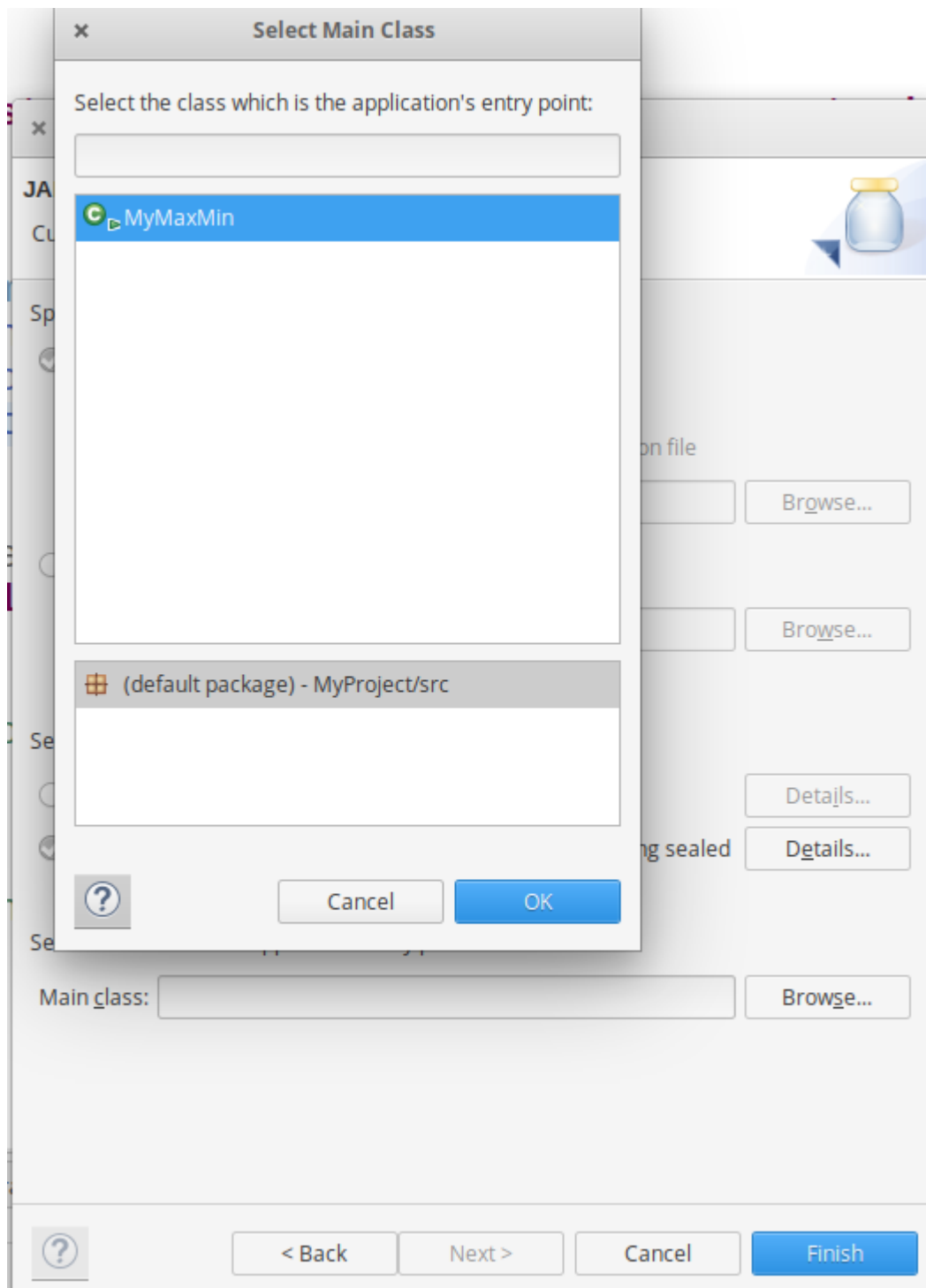
add jars from its download location then click -> Apply and Close.



- Now export the project as jar file. Right-click on MyProject choose Export.. and go to Java -> JAR file click -> Next and choose your export destination then click -> Next.  
Choose Main Class as MyMaxMin by clicking -> Browse and then click -> Finish -> Ok.







## Step 4:

Start our Hadoop Daemons

```
start-dfs.sh
```

```
start-yarn.sh
```

## Step 5:

Move your dataset to the Hadoop HDFS.

Syntax:

```
hdfs dfs -put /file_path /destination
```

In below command / shows the root directory of our HDFS.

```
hdfs dfs -put /home/dikshant/Downloads/CRND0103-2020-AK_Fairbanks_11_NE.txt /
```

Check the file sent to our HDFS.

```
hdfs dfs -ls /
```

```
dikshant@dikshant-Inspiron-5567:~$ hdfs dfs -put /home/dikshant/Downloads/CRND0103-2020-AK_Fairbanks_11_NE.txt /
dikshant@dikshant-Inspiron-5567:~$ hdfs dfs -ls /
Found 4 items
-rw-r--r--  1 dikshant supergroup    39711 2020-07-04 09:39 /CRND0103-2020-AK_Fairbanks_11_NE.txt
drwxrwxr-x+ - dikshant supergroup      0 2020-06-23 14:23 /Hadoop_File
drwxrwxrwx  - dikshant supergroup      0 2020-06-14 21:43 /tmp
drwxr-xr-x  - dikshant supergroup      0 2020-06-14 21:43 /user
dikshant@dikshant-Inspiron-5567:~$
```

## Step 6:

Now Run your Jar File with below command and produce the output in MyOutput File.

Syntax:

```
hadoop jar /jar_file_location /dataset_location_in_HDFS /output-file_name
```















Command:

```
hadoop jar /home/dikshant/Documents/Project.jar /CRND0103-2020-AK_Fairbanks_11_NE.txt /MyOutput
```

```
dikshant@dikshant-Inspiron-5567:~$ hadoop jar /home/dikshant/Documents/Project.jar /CRND0103-2020-AK_Fairbanks_11_NE.txt /MyOutput
20/07/04 09:44:40 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
20/07/04 09:44:40 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
20/07/04 09:44:41 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Im
```




## Step 7:

Now Move to localhost:50070/, under utilities select Browse the file system and download part-r-00000 in /MyOutput directory to see result.

<input type="checkbox"/>	 Permission	 Owner	 Group	 Size	 Last Modified	 Replication	 Block Size	 Name	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">dikshant</a>	<a href="#">supergroup</a>	38.78 KB	Jul 04 09:39	<a href="#">1</a>	122.07 MB	<a href="#">CRND0103-2020-AK_Fairbanks_11_NE.txt</a>	
<input type="checkbox"/>	<a href="#">drwxrwxr-x+</a>	<a href="#">dikshant</a>	<a href="#">supergroup</a>	0 B	Jun 23 14:23	<a href="#">0</a>	0 B	<a href="#">Hadoop_File</a>	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">dikshant</a>	<a href="#">supergroup</a>	0 B	Jul 04 09:44	<a href="#">0</a>	0 B	<a href="#">MyOutput</a>	
<input type="checkbox"/>	<a href="#">drwxrwxrwx</a>	<a href="#">dikshant</a>	<a href="#">supergroup</a>	0 B	Jun 14 21:43	<a href="#">0</a>	0 B	<a href="#">tmp</a>	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">dikshant</a>	<a href="#">supergroup</a>	0 B	Jun 14 21:43	<a href="#">0</a>	0 B	<a href="#">user</a>	

/MyOutput

Go!














Show

25

entries

Search:

<input type="checkbox"/>	 Permission	 Owner	 Group	 Size	 Last Modified	 Replication	 Block Size	 Name	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">dikshant</a>	<a href="#">supergroup</a>	0 B	Jul 04 09:44	<a href="#">1</a>	122.07 MB	<a href="#">_SUCCESS</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">dikshant</a>	<a href="#">supergroup</a>	3.85 KB	Jul 04 09:44	1	122.07 MB	<a href="#">part-r-00000</a>	

## Step 8:

See the result in the Downloaded File.

1	The Day is Cold Day :20200101	-21.8
2	The Day is Cold Day :20200102	-23.4
3	The Day is Cold Day :20200103	-25.4
4	The Day is Cold Day :20200104	-26.8
5	The Day is Cold Day :20200105	-28.8
6	The Day is Cold Day :20200106	-30.0
7	The Day is Cold Day :20200107	-31.4
8	The Day is Cold Day :20200108	-33.6
9	The Day is Cold Day :20200109	-26.6
10	The Day is Cold Day :20200110	-24.3

In the above image, you can see the top 10 results showing the cold days. The second column is a day in yyyy/mm/dd format. For Example, 20200101 means

year = 2020

month = 01

Date = 01