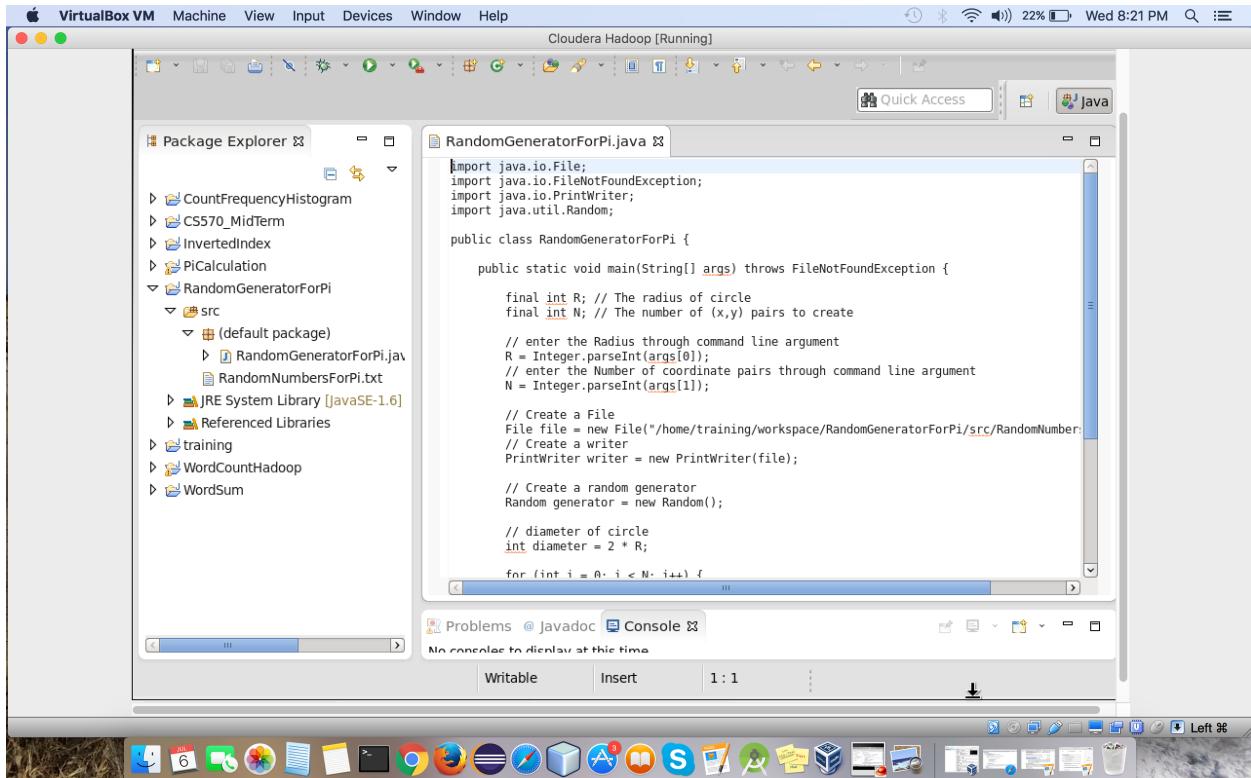


Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com
Student ID: 15977

Calculate Pi :

Create RandomGeneratorForPi.java class



The screenshot shows the Eclipse IDE interface. The title bar says "VirtualBox VM Machine View Input Devices Window Help" and "Cloudera Hadoop [Running]". The status bar at the bottom shows "22% 8:21 PM". The Package Explorer view on the left lists several projects: CountFrequencyHistogram, CS570_MidTerm, InvertedIndex, PiCalculation, and RandomGeneratorForPi. The RandomGeneratorForPi project is expanded, showing a src folder containing RandomGeneratorForPi.java and RandomNumbersForPi.txt. The RandomGeneratorForPi.java file is open in the editor, displaying the following Java code:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Random;

public class RandomGeneratorForPi {

    public static void main(String[] args) throws FileNotFoundException {
        final int R; // The radius of circle
        final int N; // The number of (x,y) pairs to create

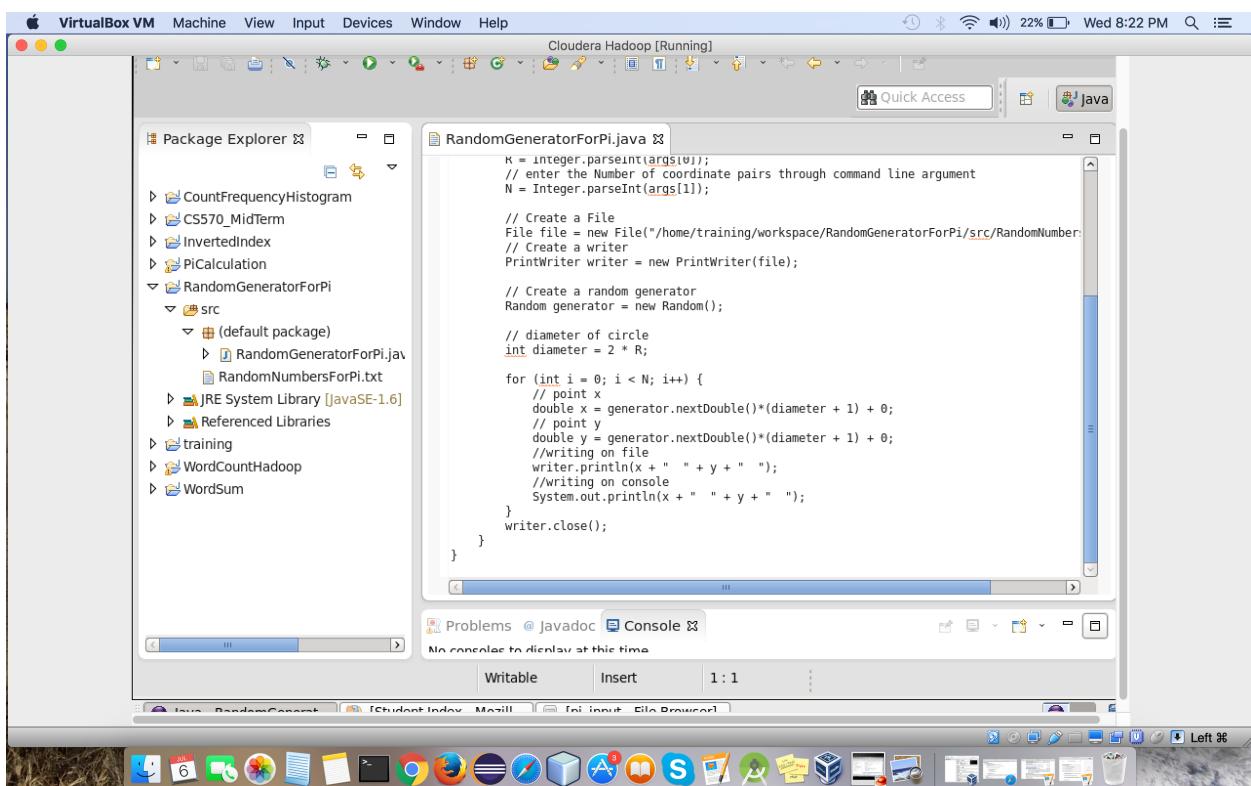
        // enter the Radius through command line argument
        R = Integer.parseInt(args[0]);
        // enter the Number of coordinate pairs through command line argument
        N = Integer.parseInt(args[1]);

        // Create a File
        File file = new File("./home/training/workspace/RandomGeneratorForPi/src/RandomNumbersForPi.txt");
        // Create a writer
        PrintWriter writer = new PrintWriter(file);

        // Create a random generator
        Random generator = new Random();

        // diameter of circle
        int diameter = 2 * R;

        for (int i = 0; i < N; i++) {
            ... // code omitted for brevity
        }
    }
}
```



The screenshot shows the Eclipse IDE interface again, similar to the previous one but with more code visible in the editor. The title bar and status bar are identical. The Package Explorer view shows the same project structure. The RandomGeneratorForPi.java file is open in the editor, displaying the following Java code:

```
K = integer.parseInt(args[0]);
// enter the Number of coordinate pairs through command line argument
N = Integer.parseInt(args[1]);

// Create a File
File file = new File("./home/training/workspace/RandomGeneratorForPi/src/RandomNumbersForPi.txt");
// Create a writer
PrintWriter writer = new PrintWriter(file);

// Create a random generator
Random generator = new Random();

// diameter of circle
int diameter = 2 * R;

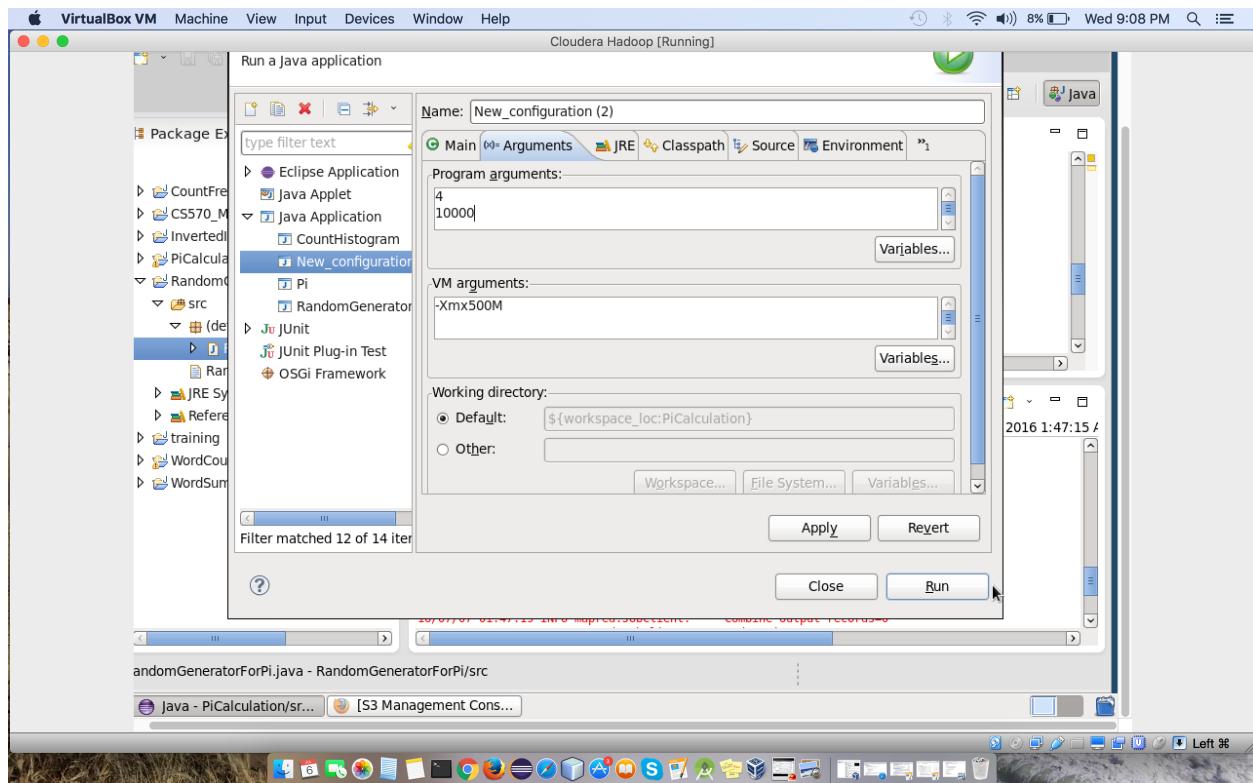
for (int i = 0; i < N; i++) {
    // point x
    double x = generator.nextDouble()*(diameter + 1) + 0;
    // point y
    double y = generator.nextDouble()*(diameter + 1) + 0;
    //writing on file
    writer.println(x + " " + y + " ");
    //writing on console
    System.out.println(x + " " + y + " ");
}
writer.close();
}
```

Name: Sayali Vishnu Kute

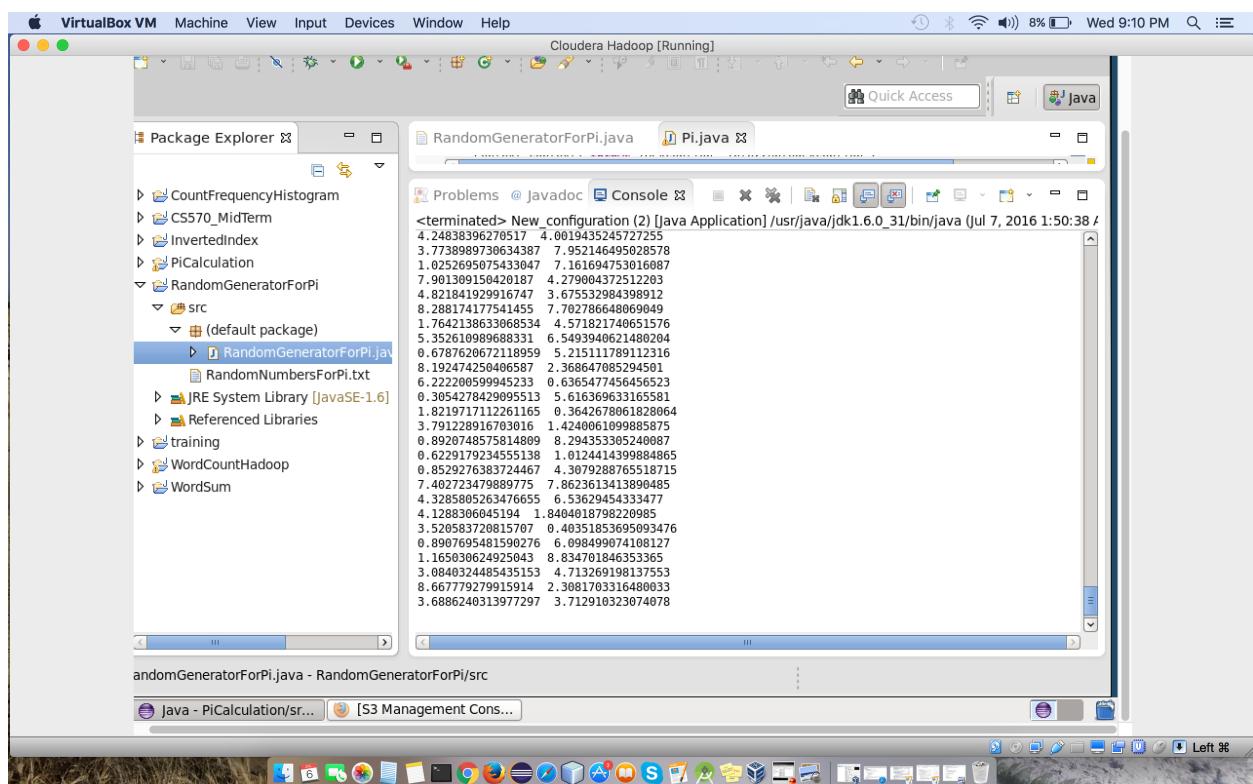
Email ID: sayalikute2507@gmail.com

Student ID: 15977

Enter Arguments as Radius=4 and Number of pairs to generate=10000:



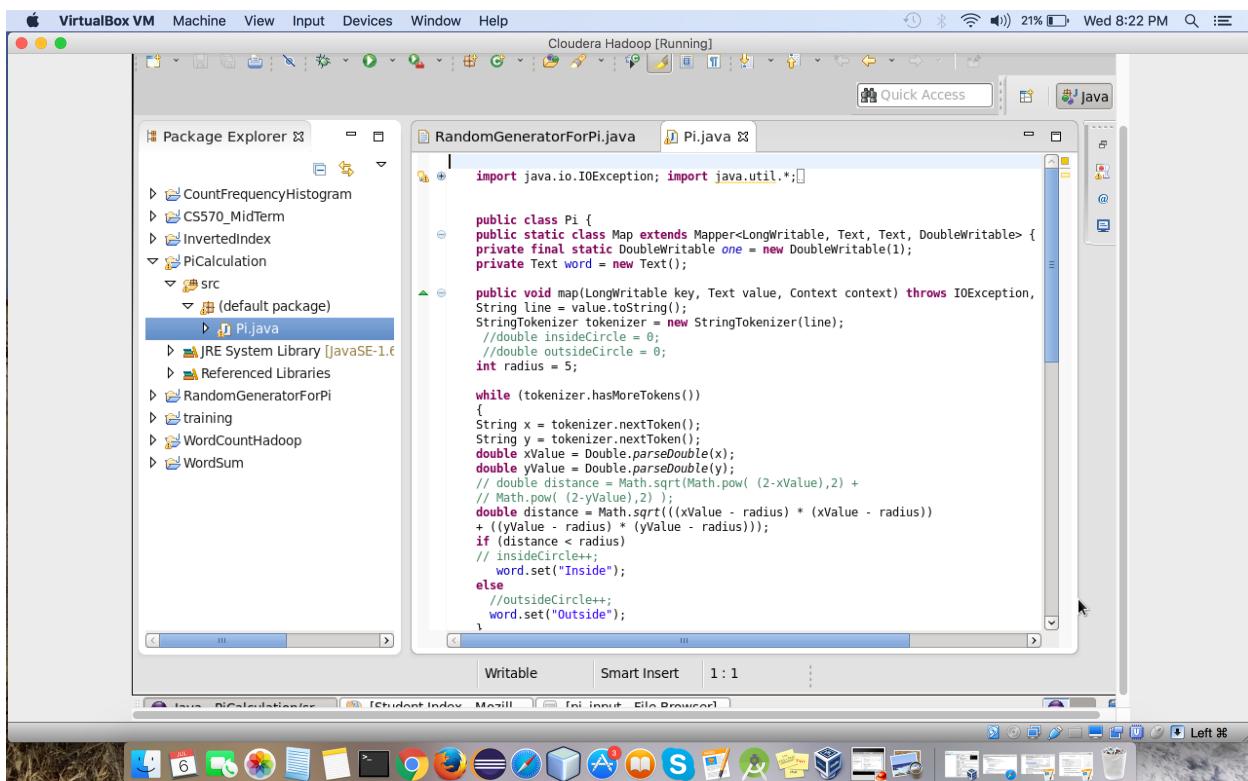
Run the RandomGeneratorForPi.java class in command line and check the output:



Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com
Student ID: 15977

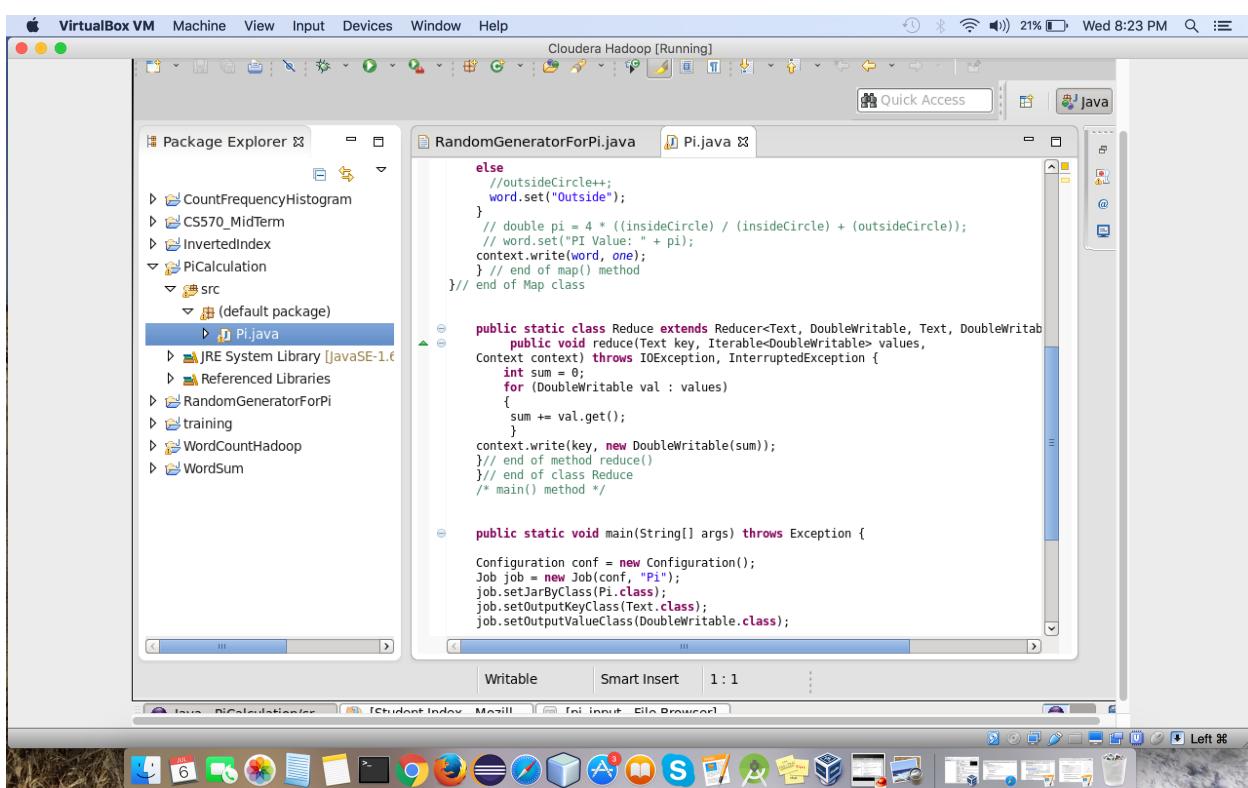
The above output generated is save in RandomGeneratorForPi.txt file.

Create a Pi.java class:



Screenshot of the Eclipse IDE interface showing the RandomGeneratorForPi.java file in the editor. The code implements the map() method of the Mapper interface. It tokenizes input lines, calculates distances from a center point (0,0) with radius 5, and counts points inside or outside the circle. The code uses StringTokenizer, Double.parseDouble, and Math.sqrt methods. The editor shows syntax highlighting and code completion features.

```
import java.io.IOException; import java.util.*;public class Pi { public static class Map extends Mapper<LongWritable, Text, Text, DoubleWritable> { private final static DoubleWritable one = new DoubleWritable(1); private Text word = new Text(); public void map(LongWritable key, Text value, Context context) throws IOException, String line = value.toString(); StringTokenizer tokenizer = new StringTokenizer(line); double insideCircle = 0; double outsideCircle = 0; int radius = 5; while (tokenizer.hasMoreTokens()) { String x = tokenizer.nextToken(); String y = tokenizer.nextToken(); double xValue = Double.parseDouble(x); double yValue = Double.parseDouble(y); double distance = Math.sqrt(Math.pow((xValue - 0), 2) + Math.pow((yValue - 0), 2)); double distance = Math.sqrt(((xValue - radius) * (xValue - radius)) + ((yValue - radius) * (yValue - radius))); if (distance < radius) // insideCircle++; word.set("Inside"); else //outsideCircle++; word.set("Outside"); } } } // end of Map class
```



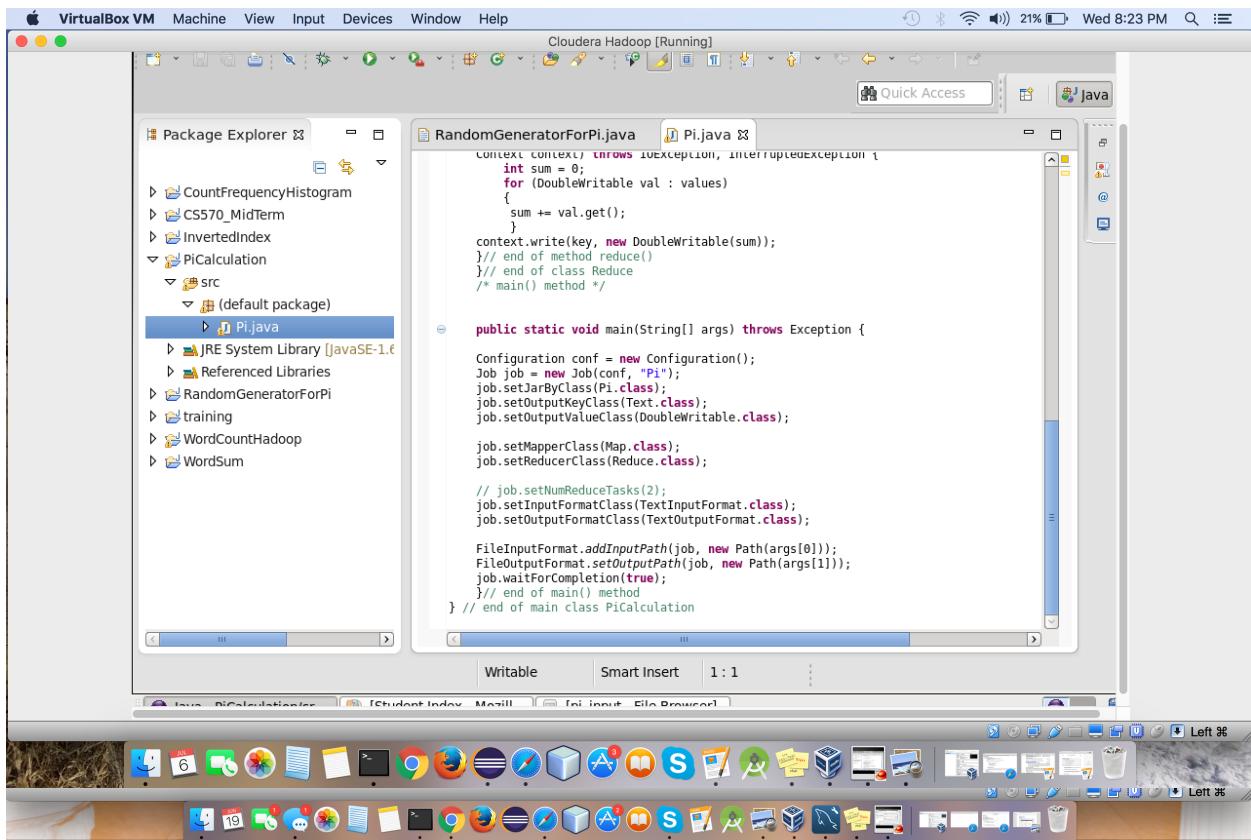
Screenshot of the Eclipse IDE interface showing the RandomGeneratorForPi.java file in the editor. The code includes the reduce() method which sums up the values and the main() method which sets up the Hadoop job configuration. The editor shows the full Java code for the Pi class, including the Map and Reduce inner classes.

```
else //outsideCircle++; word.set("Outside"); } // double pi = 4 * ((insideCircle) / (insideCircle) + (outsideCircle)); // word.set("PI Value: " + pi); context.write(word, one); } // end of map() method } // end of Map class
```

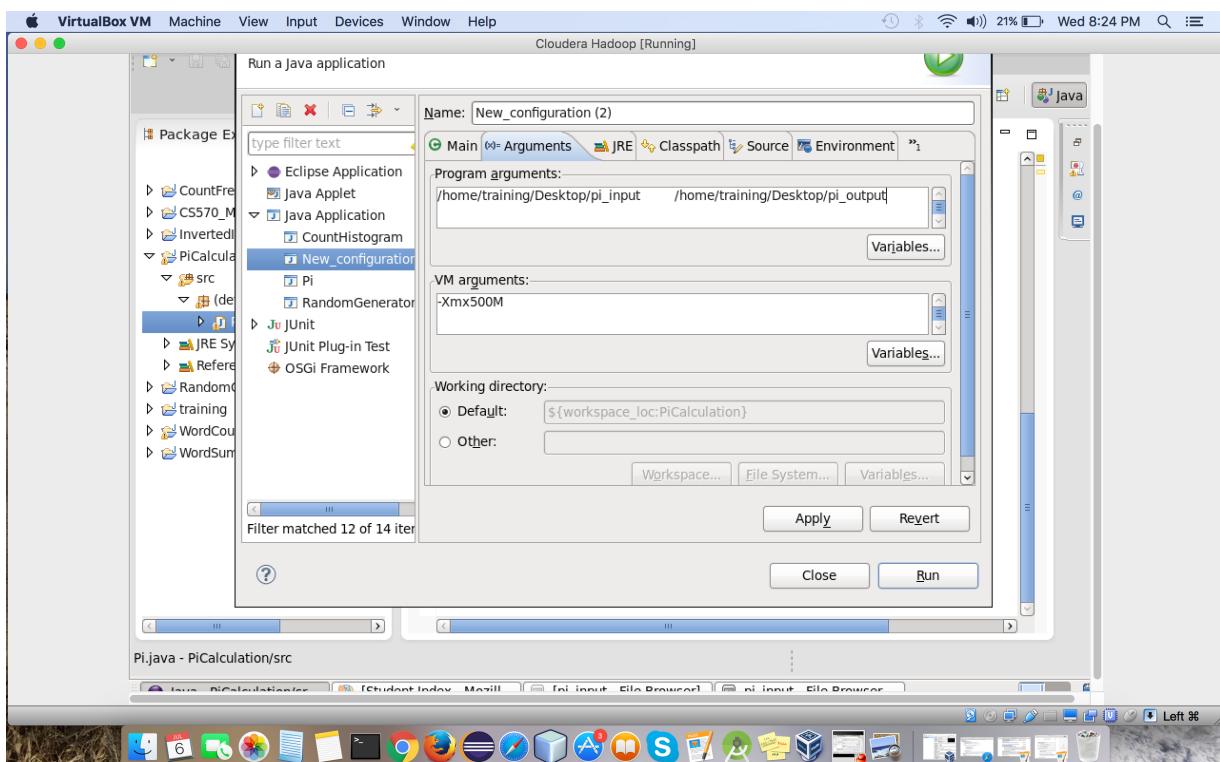
```
public static class Reduce extends Reducer<Text, DoubleWritable, Text, DoubleWritable> { public void reduce(Text key, Iterable<DoubleWritable> values, Context context) throws IOException, InterruptedException { int sum = 0; for (DoubleWritable val : values) { sum += val.get(); } context.write(key, new DoubleWritable(sum)); } // end of method reduce() } // end of class Reduce /* main() method */
```

```
public static void main(String[] args) throws Exception { Configuration conf = new Configuration(); Job job = new Job(conf, "Pi"); job.setJarByClass(Pi.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(DoubleWritable.class); }
```

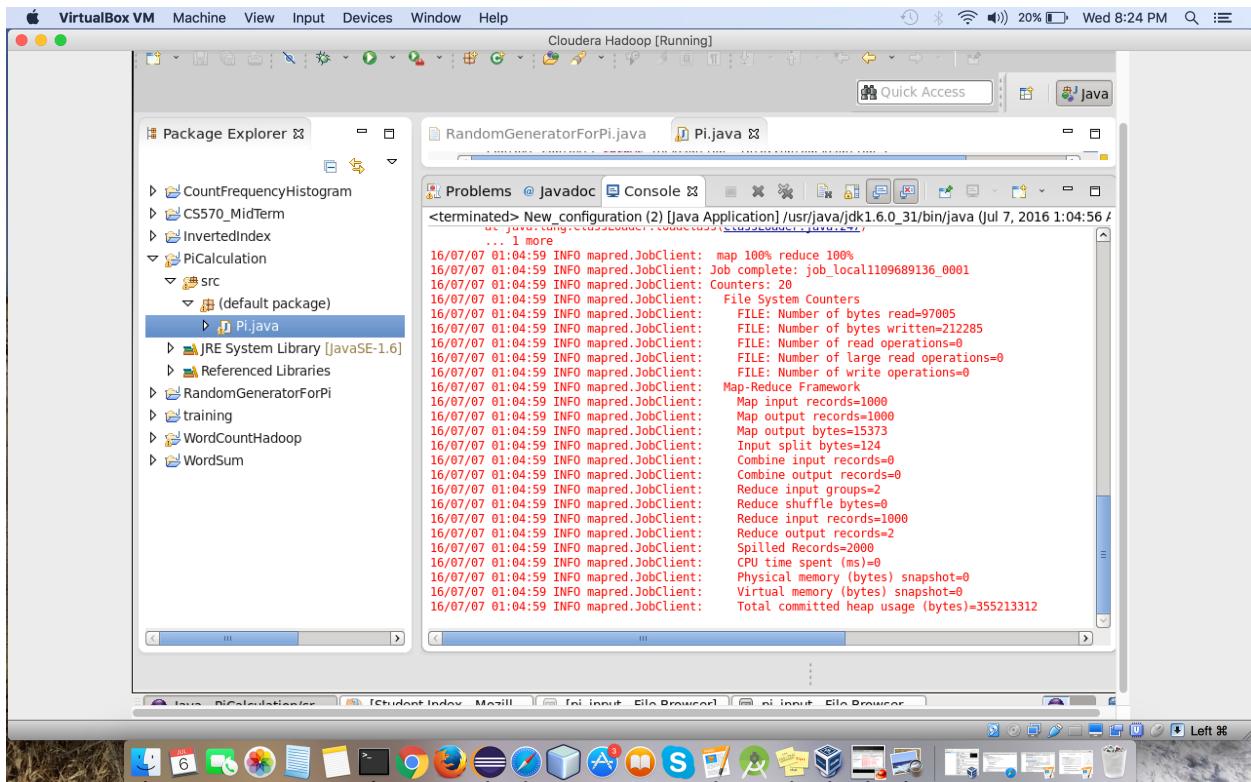
Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com
Student ID: 15977



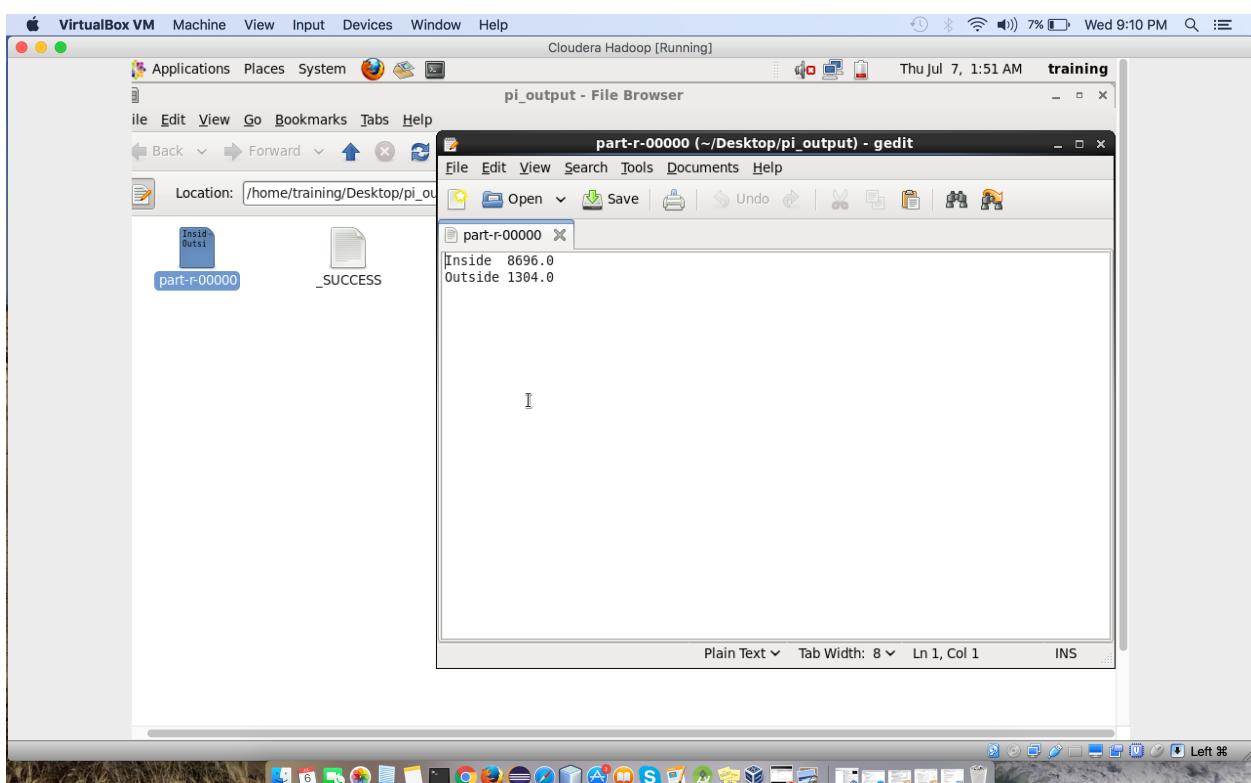
In the argument, provide the input file path where RandomGeneratorForPi.txt is saved and the path where output file will be automatically generated:



Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com
Student ID: 15977
Run the Pi.java class:



**The following output is generated in Pi_output file:
Number of darts inside:8696.0
Number of darts outside: 1304.0**

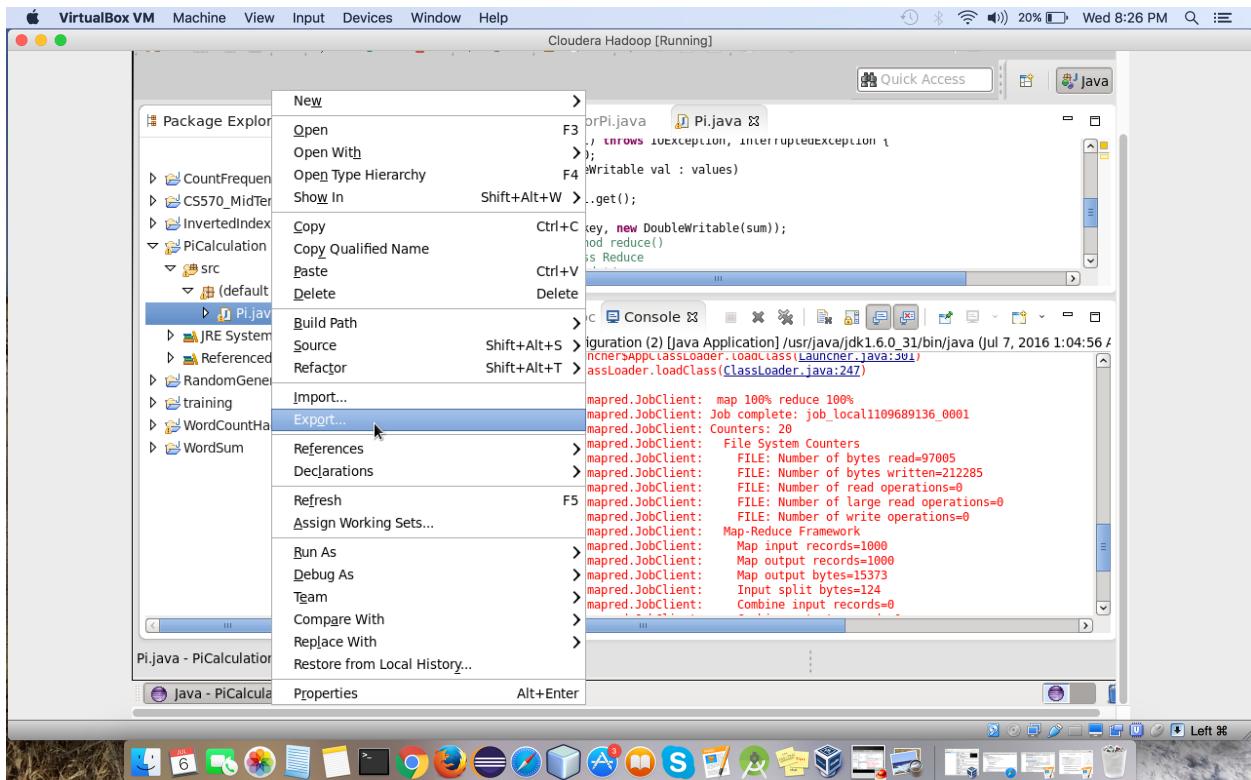


Name: Sayali Vishnu Kute

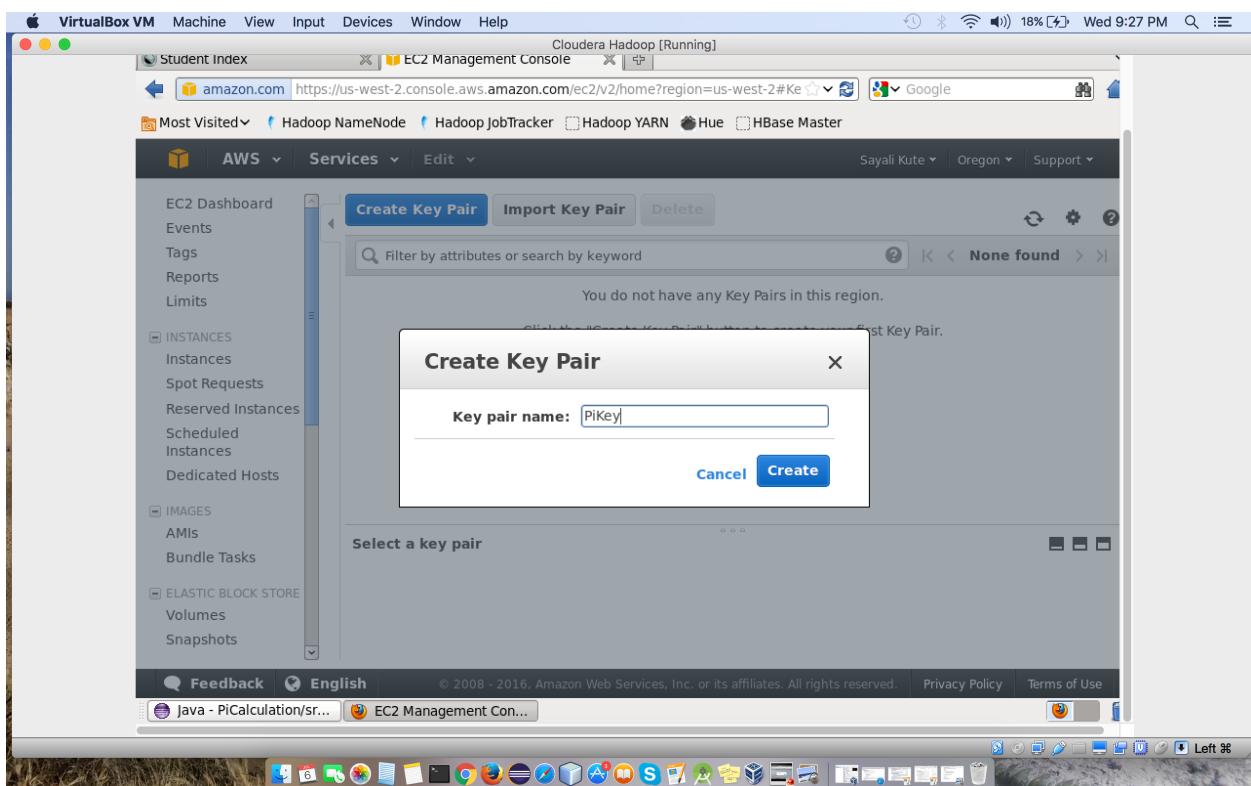
Email ID: sayalikute2507@gmail.com

Student ID: 15977

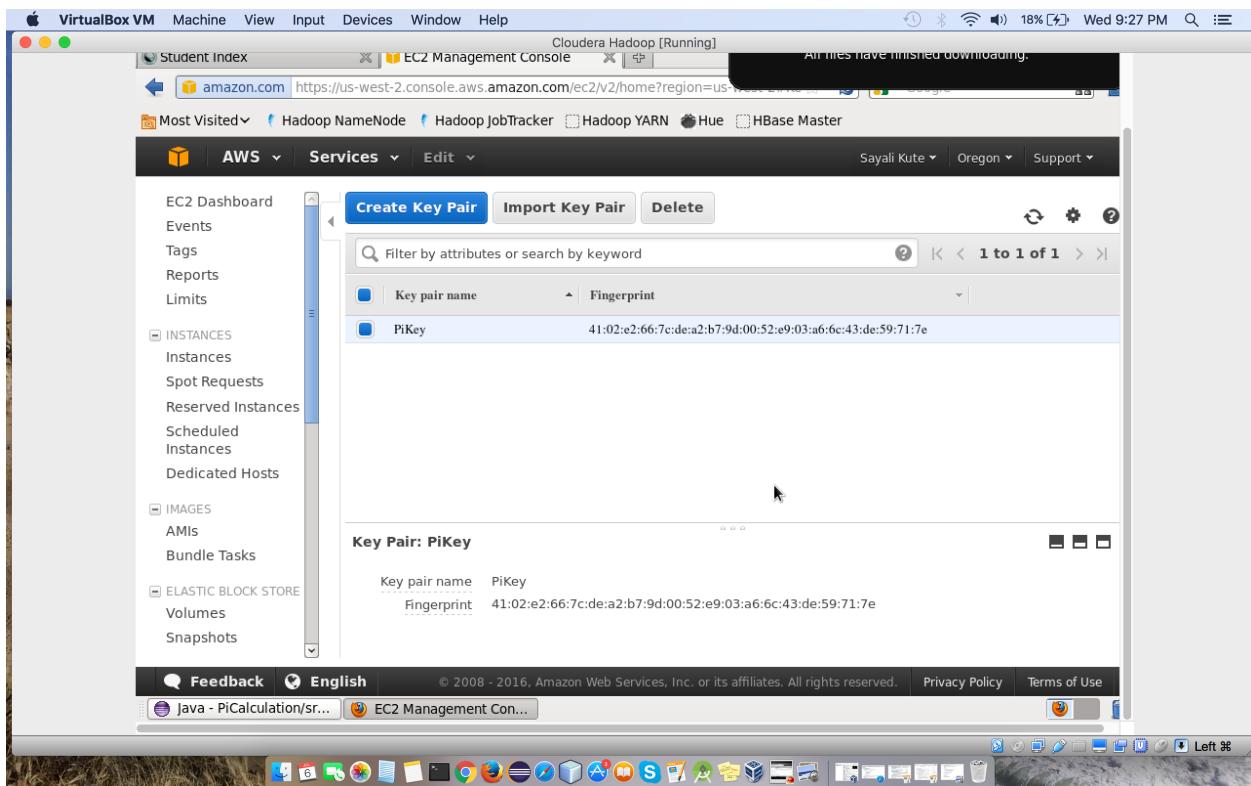
Export the Pi.java class and save it with .jar extension:



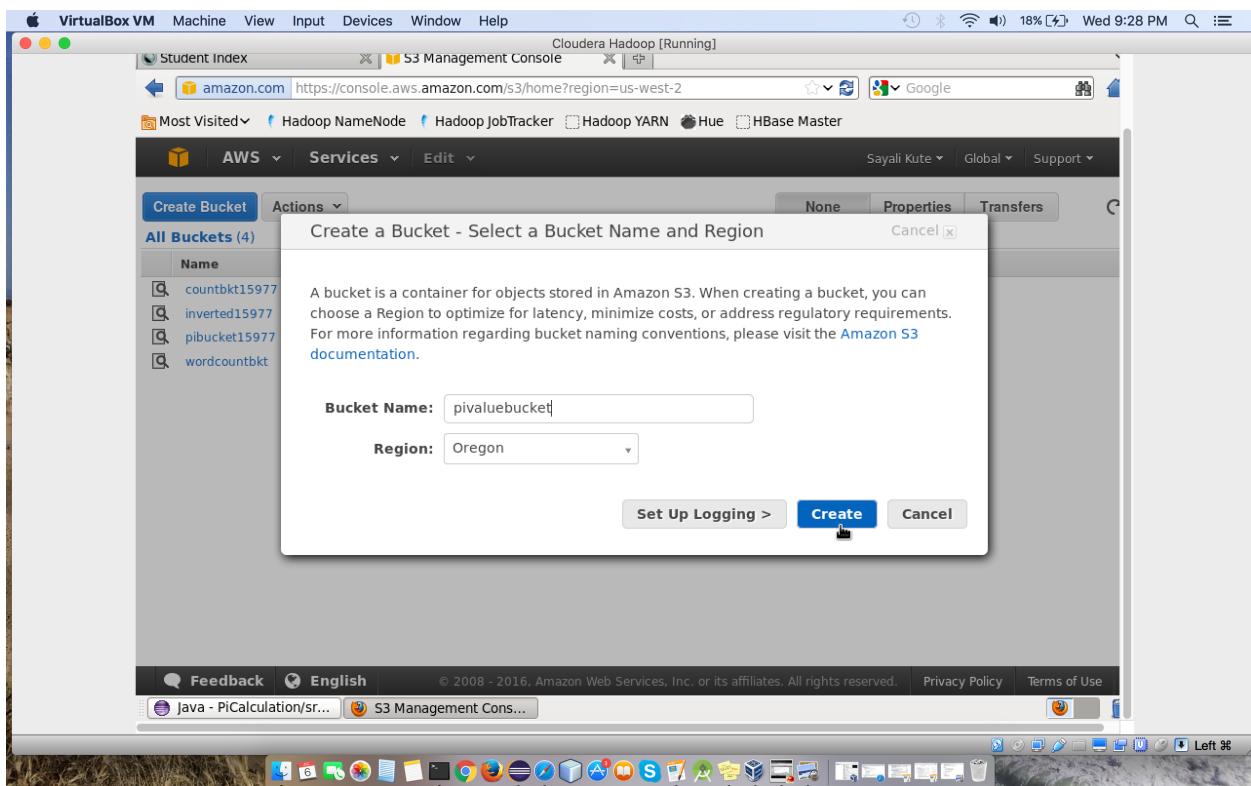
In AWS, create key pair through EC2 service:



Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com
Student ID: 15977



Create Bucket through S3 service:



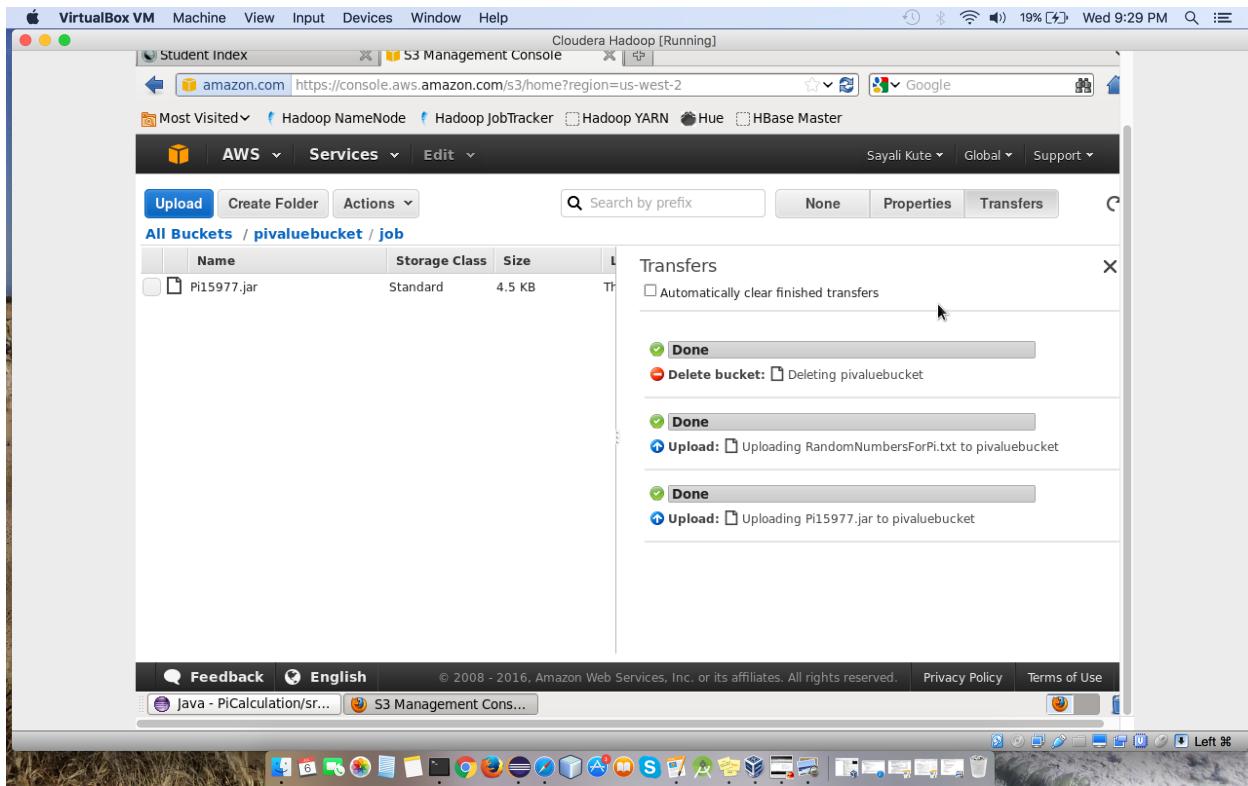
Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com
Student ID: 15977

The screenshot shows the AWS S3 Management Console interface. At the top, there are tabs for 'Student Index' and 'S3 Management Console'. Below the tabs, the URL is https://console.aws.amazon.com/s3/home?region=us-west-2. The main navigation bar includes 'AWS', 'Services', 'Edit', 'Upload', 'Create Folder', and 'Actions'. A search bar says 'Search by prefix' with 'None' selected. There are also 'Properties' and 'Transfers' buttons. The title bar indicates 'All Buckets / pivaluebucket'. The table below lists four folders: 'data', 'job', 'logs', and 'result', all with storage class 'Standard', size '--', and last modified '--'. The bottom of the screen shows the Mac OS X dock with various application icons.

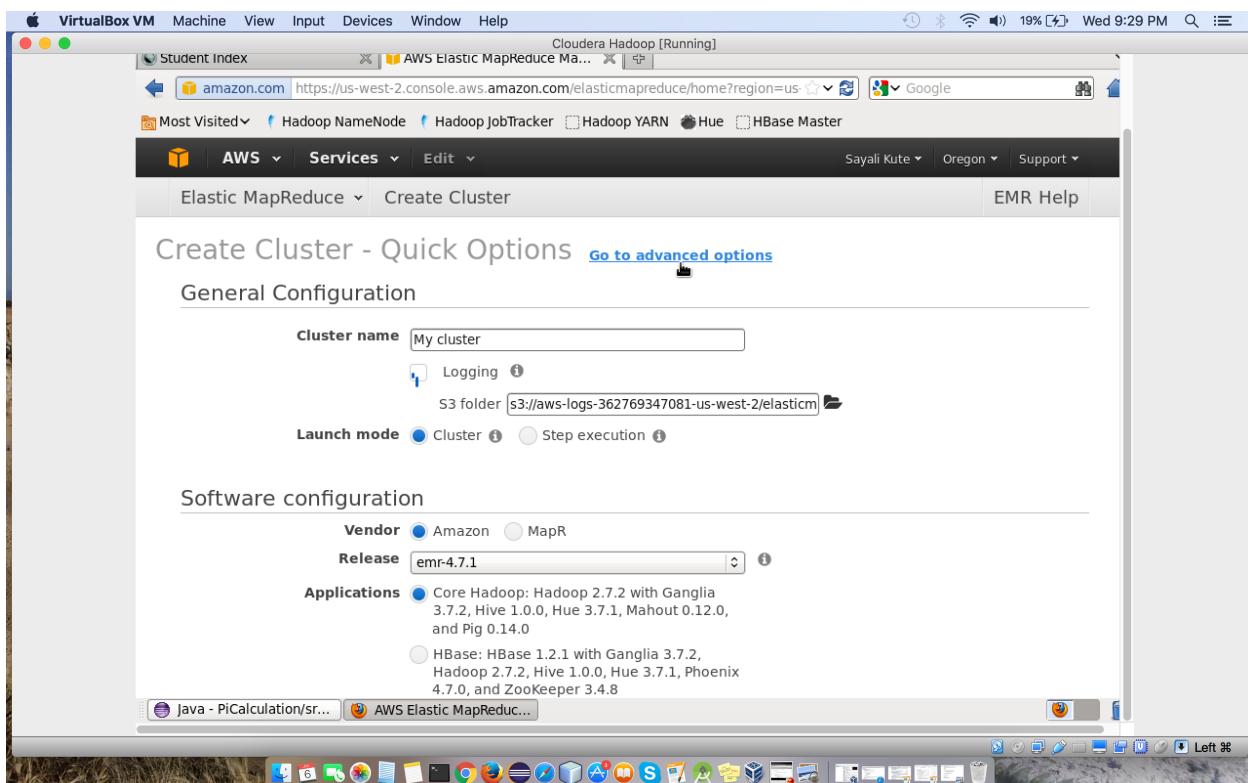
Upload the RandomGeneratorForPi.txt file in ‘data’ folder and exported Pi15977.jar file in the ‘job’ folder.

This screenshot shows the same AWS S3 Management Console interface as the previous one, but with a modal dialog open in the foreground. The dialog is titled 'Transfers' and displays two entries: 'Done' (with a green checkmark) and 'Delete bucket: Deleting pivaluebucket' (with a red error icon). Below that, another 'Done' entry is shown with the message 'Uploading RandomNumbersForPi.txt to pivaluebucket'. The background shows the same S3 bucket structure as the first screenshot.

Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com
Student ID: 15977



Create a cluster through EMR service and select all the required advance choices:

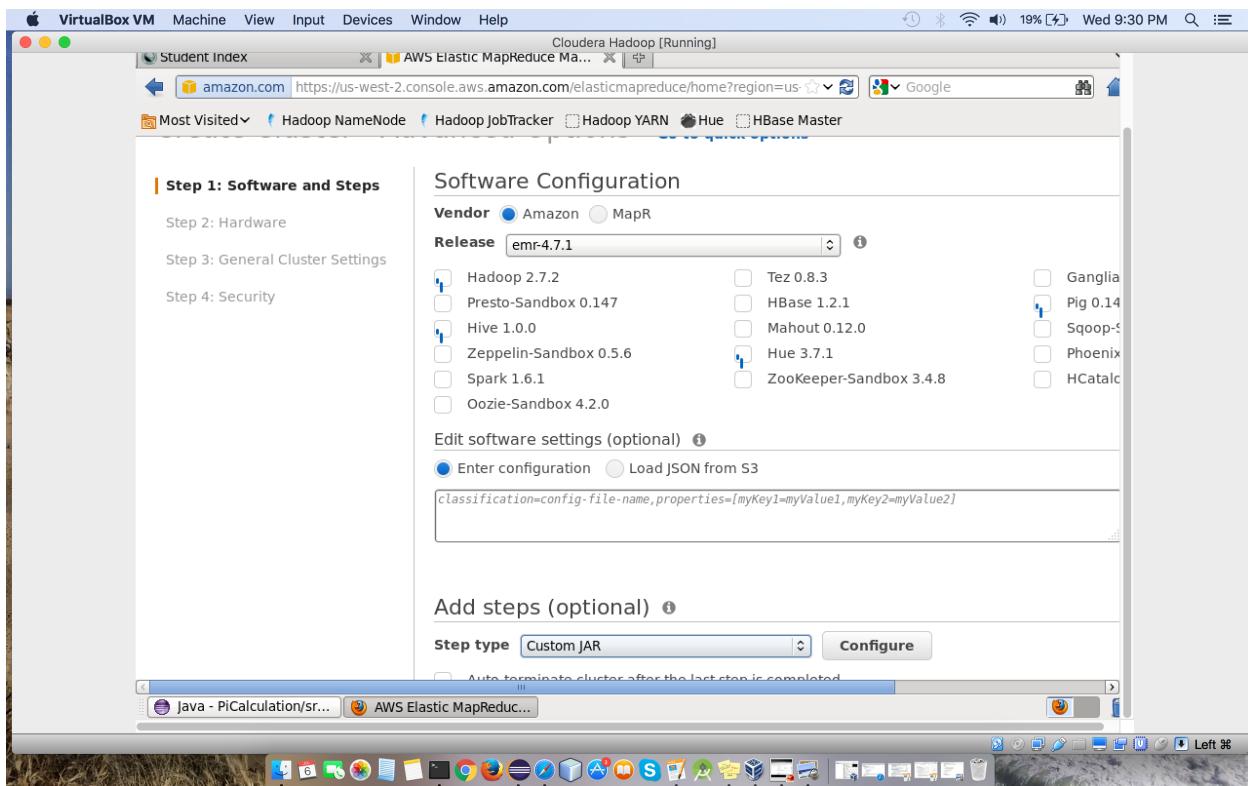


Name: Sayali Vishnu Kute

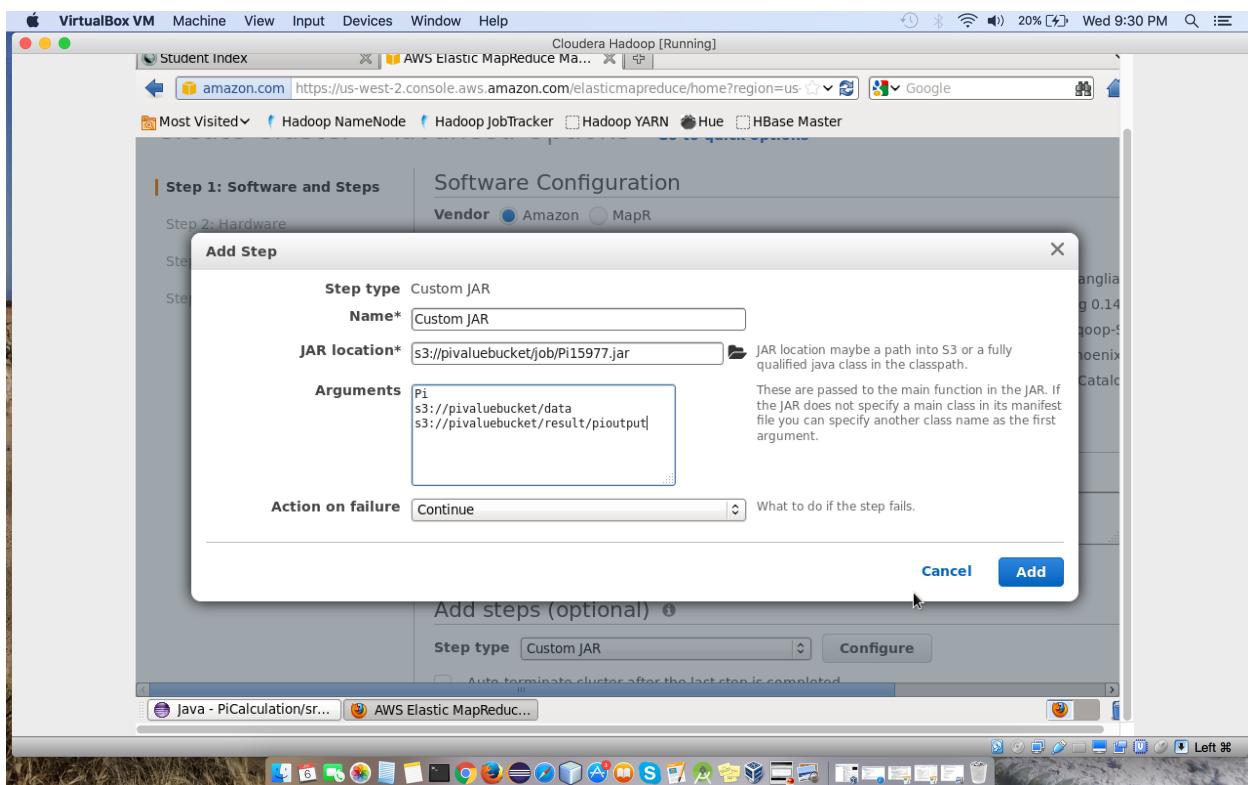
Email ID: sayalikute2507@gmail.com

Student ID: 15977

Select ‘step type’ as ‘Custom JAR’ and configure other settings as follows:



Provide JAR file location path and arguments:



Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com
Student ID: 15977

Make instance count of 'core' as zero:

The screenshot shows the 'Hardware Configuration' step of the AWS Elastic MapReduce cluster creation wizard. The 'Core' instance group is selected, and its 'Instance count' is set to 0. Other groups like 'Master' and 'Task' also have their counts set to 1. The 'EC2 instance type' for all groups is 'm3.xlarge'. The 'Storage per instance' is 80 GiB, and the 'Request spot price' checkbox is unchecked.

Type	Name	EC2 instance type	Instance count	Storage per instance	Request spot price
Master	Master instance group	m3.xlarge	1	80 GiB Add EBS volumes	<input type="checkbox"/>
Core	Core instance group - 2	m3.xlarge	0	80 GiB Add EBS volumes	<input type="checkbox"/>
Task	Task instance group - 3	m3.xlarge	0	80 GiB Add EBS volumes	<input type="checkbox"/>

Provide the log folder path present in the S3 Service:

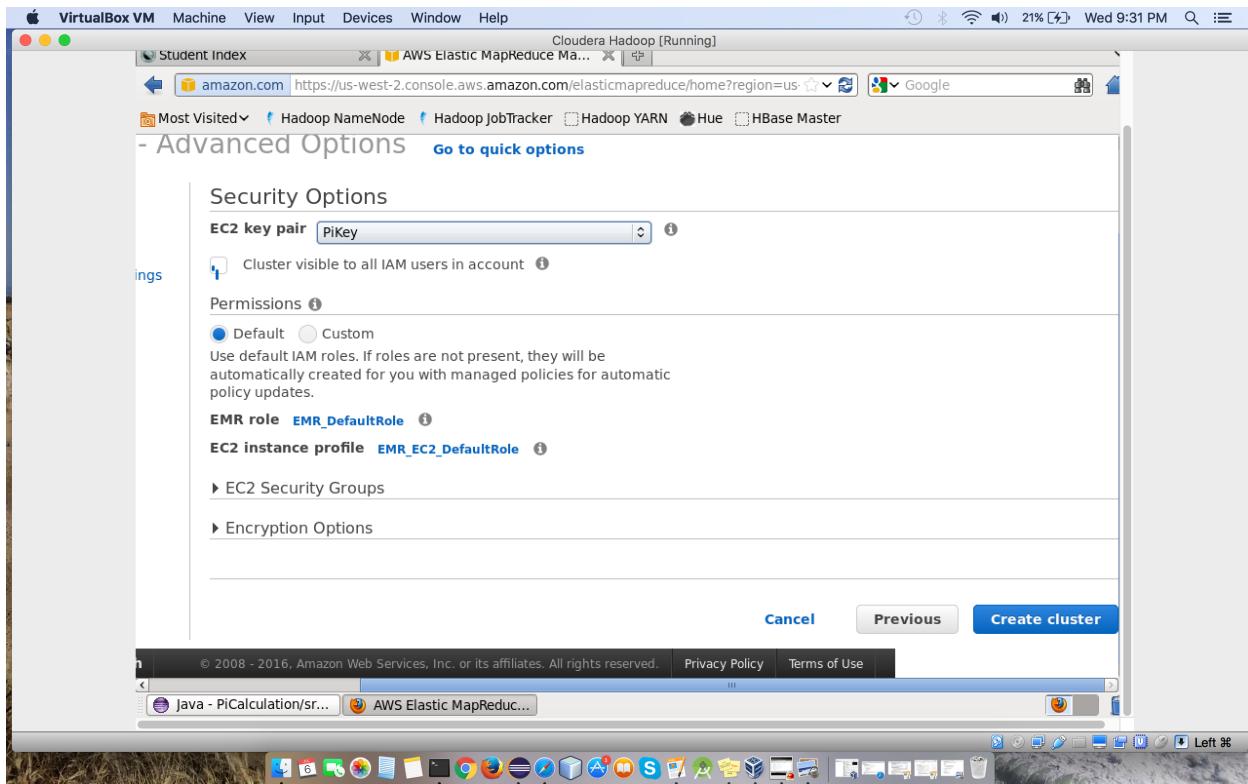
The screenshot shows the 'Advanced Options' step of the AWS Elastic MapReduce cluster creation wizard. Under the 'General Options' section, the 'Cluster name' is set to 'PiCluster'. In the 'Logging' section, the 'S3 folder' is specified as 's3://pivaluebucket/logs/'. There are also sections for 'Debugging' and 'Termination protection'. Below these, there is a 'Tags' section where users can add key-value pairs, but no tags are currently defined.

Name: Sayali Vishnu Kute

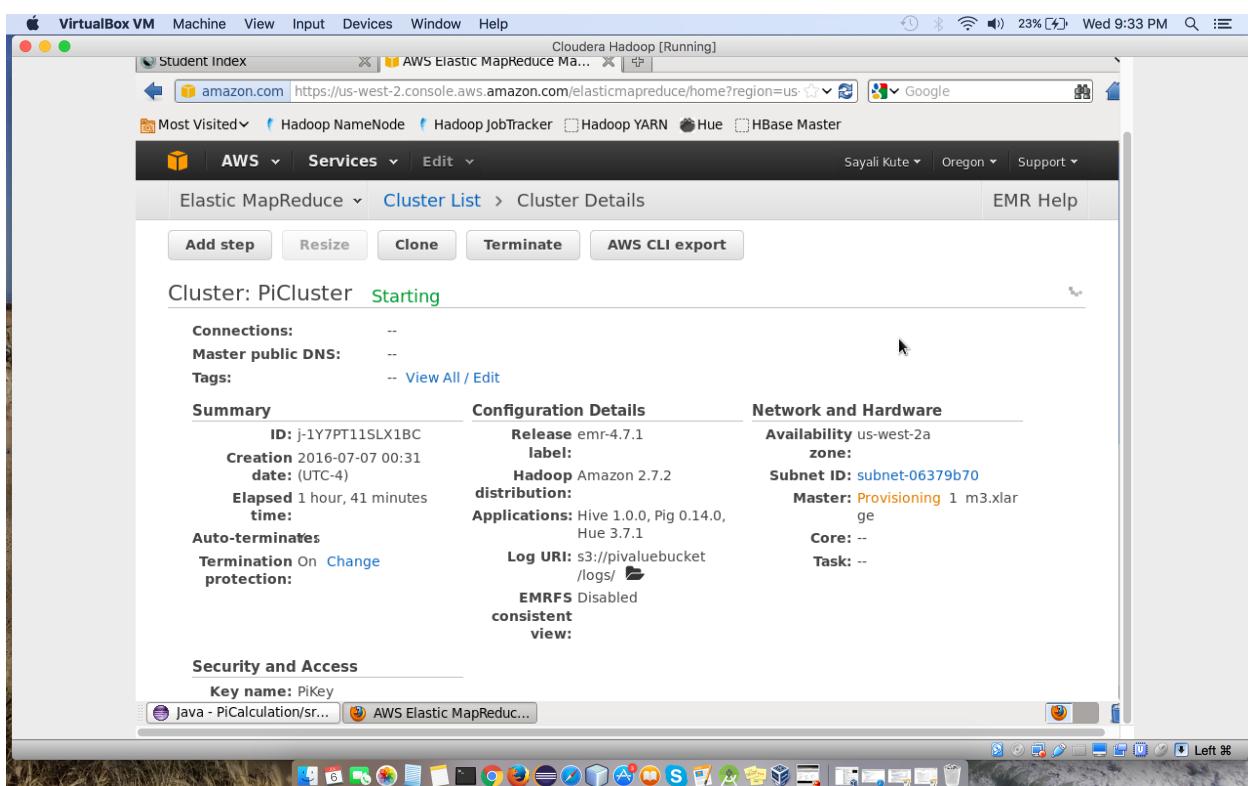
Email ID: sayalikute2507@gmail.com

Student ID: 15977

Select the appropriate EC2 key pair:



Cluster has started working and runs successfully :



Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com
Student ID: 15977

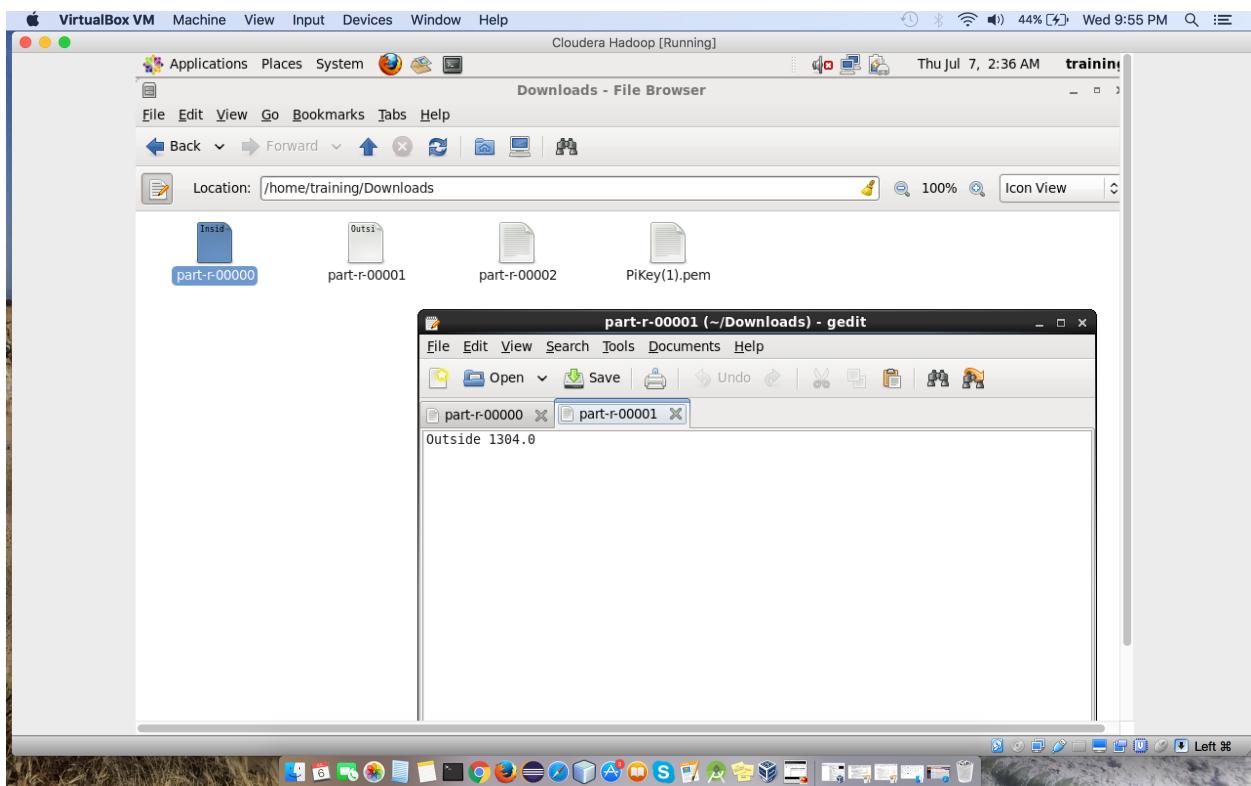
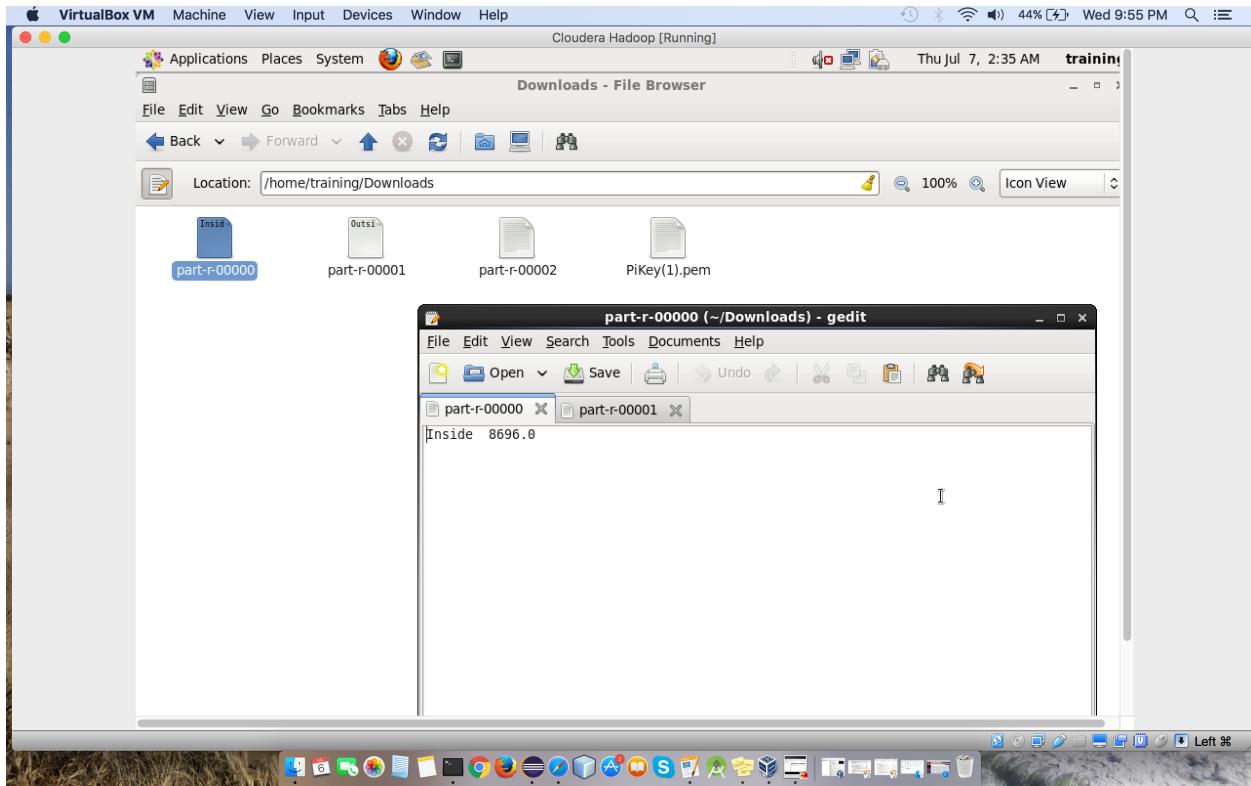
The screenshot shows the AWS Elastic MapReduce Cluster Details page. The cluster is named "PiCluster" and is listed as "Terminated". The "Summary" section includes details like ID: j-1Y7PT11SLX1BC, Creation: 2016-07-07 00:31 (UTC-4), and End date: 2016-07-07 00:40 (UTC-4). The "Configuration Details" section shows Release: emr-4.7.1, Hadoop: Amazon 2.7.2, Applications: Hive 1.0.0, Pig 0.14.0, Hue 3.7.1, and Log URI: s3://pivaluebucket/logs/. The "Network and Hardware" section indicates Availability: us-west-2a, zone: subnet-06379b70, Master: Bootstrapping, and Core: --. The "Security and Access" section shows Key name: PiKey.

The output is generated in the ‘results’ folder of S3 Service:

The screenshot shows the AWS S3 Management Console. The user is viewing the contents of the "pioutput" bucket under the "result" folder. The table lists five objects: "_SUCCESS" (Standard storage class, 0 bytes, last modified: Thu Jul 07 00:39:42 GMT-400 201), and four part files ("part-r-00000", "part-r-00001", "part-r-00002", "part-r-00003") each with a size of 0 bytes and last modified at Thu Jul 07 00:39:42 GMT-400 201.

Name	Storage Class	Size	Last Modified
_SUCCESS	Standard	0 bytes	Thu Jul 07 00:39:42 GMT-400 201
part-r-00000	Standard	14 bytes	Thu Jul 07 00:39:39 GMT-400 201
part-r-00001	Standard	15 bytes	Thu Jul 07 00:39:41 GMT-400 201
part-r-00002	Standard	0 bytes	Thu Jul 07 00:39:42 GMT-400 201

Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com
Student ID: 15977



Name: Sayali Vishnu Kute
Email ID: sayalikute2507@gmail.com

Student ID: 15977

Calculation of PI value:

Output File (pioutput): This file contains value listed as:

Inside : 8696.0

Outside : 1304.0

Formula to calculate pi : $4*(S/N) = \pi$

where, S = Number of darts hit inside the circle

N = Total number of darts thrown

$$\text{Pi} = 4 * (\text{Inside} / (\text{Inside} + \text{Outside}))$$

$$\text{Pi} = 4 * (8696.0 / (8696.0 + 1304.0))$$

$$\text{Pi} = 4 * (8696.0 / 10000)$$

$$\text{Pi} = 4 * 0.8696$$

$$\text{Pi} = 3.4784 \text{ (approximately)}$$