TASK-1 Find the number of unique listeners in the data set.

Below is the mapper code for the same task-

```
package task1;

import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;

public class UniqListenerMapper extends Mapper<LongWritable,Text,Text,IntWritable>{

	public void map(LongWritable key, Text values, Context context)
			throws IOException,InterruptedException{

		String[] line = values.toString().split("\\|");
		Text uid = new Text(line[0]);
		IntWritable value = new IntWritable(1);
		context.write(uid,value);
	}
}
```

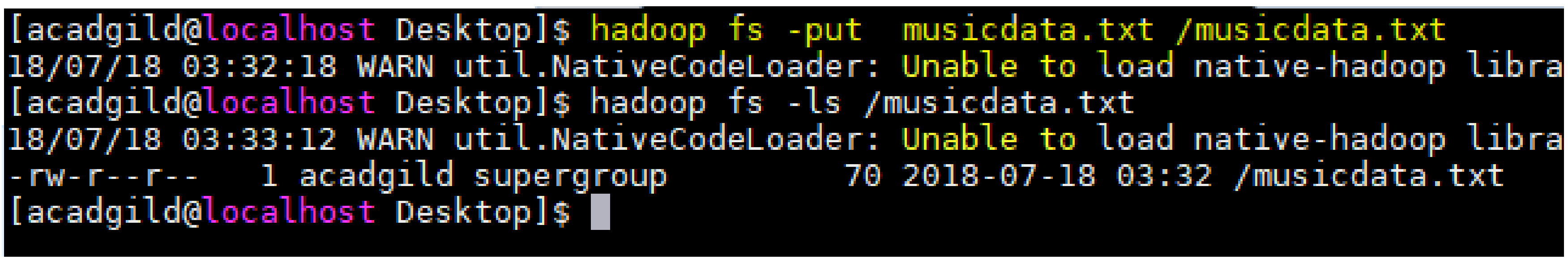
Below is the reducer code for the same task-

```
package task1;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class UniqueListenerReducer extends Reducer<Text,IntWritable,Text,IntWritable> {

	public void reduce(Text uid, Iterable<IntWritable> count, Context context)
				throws IOException, InterruptedException{
		int sum = 0;
		for(IntWritable values: count){
			sum+= values.get();
		}
		context.write(uid,new IntWritable(sum));
	}
}
```

Below is the driver code for same task-

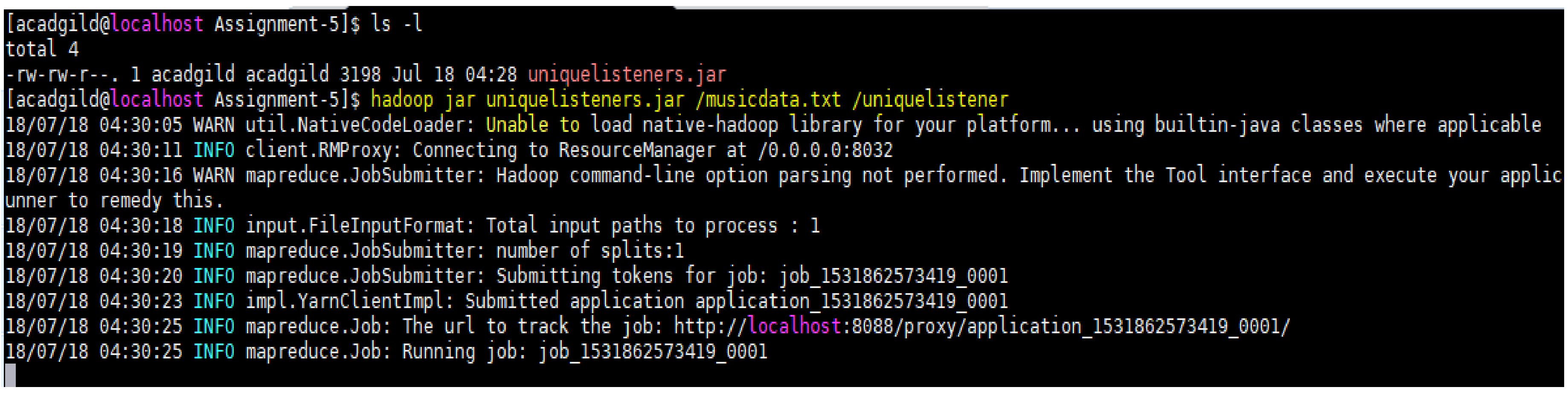
```
package task1;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class UniqueListener {

	public static void main(String[] args) throws Exception {
		// TODO Auto-generated method stub
		Configuration conf = new Configuration();
		Job job = new ~~Job~~(conf, "Assignment 5");
		job.setJarByClass(UniqueListener.class);

		job.setMapOutputKeyClass(Text.class);
		job.setMapOutputValueClass(IntWritable.class);

		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);

		job.setMapperClass(UniqListenerMapper.class);
		job.setReducerClass(UniqueListenerReducer.class);

		job.setNumReduceTasks(2);
		job.setInputFormatClass(TextInputFormat.class);
		job.setOutputFormatClass(TextOutputFormat.class);

		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job,new Path(args[1]));

		job.waitForCompletion(true);
	}
}
```

First we will put the input file at the HDFS location as shown in below screenshot-



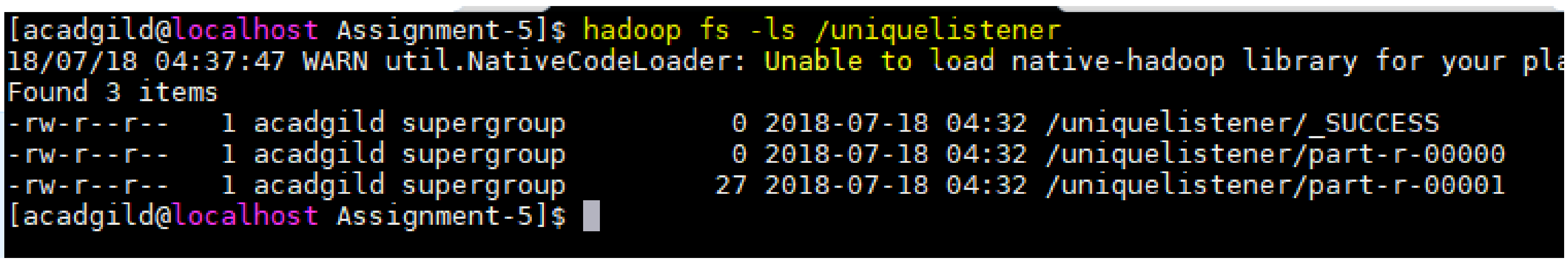
```
[acadgild@localhost Desktop]$ hadoop fs -put  musicdata.txt /musicdata.txt
18/07/18 03:32:18 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
[acadgild@localhost Desktop]$ hadoop fs -ls /musicdata.txt
18/07/18 03:33:12 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
-rw-r--r--   1 acadgild supergroup         70 2018-07-18 03:32 /musicdata.txt
[acadgild@localhost Desktop]$
```

Then we will run the mapreduce job with the jar created by Map Reduce Code as shown below-



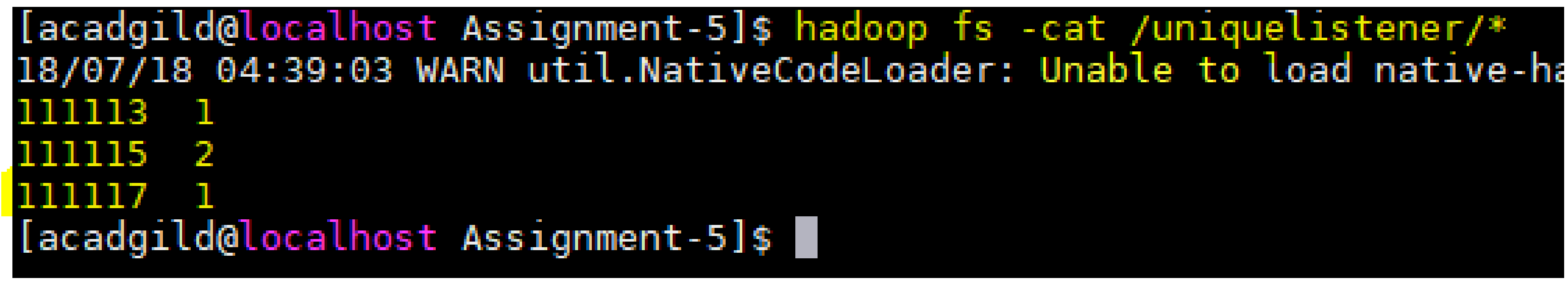
```
[acadgild@localhost Assignment-5]$ ls -l
total 4
-rw-rw-r--. 1 acadgild acadgild 3198 Jul 18 04:28 uniquelisteners.jar
[acadgild@localhost Assignment-5]$ hadoop jar uniquelisteners.jar /musicdata.txt /uniquelistener
18/07/18 04:30:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/07/18 04:30:11 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
18/07/18 04:30:16 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your applic
unner to remedy this.
18/07/18 04:30:18 INFO input.FileInputFormat: Total input paths to process : 1
18/07/18 04:30:19 INFO mapreduce.JobSubmitter: number of splits:1
18/07/18 04:30:20 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1531862573419_0001
18/07/18 04:30:23 INFO impl.YarnClientImpl: Submitted application application_1531862573419_0001
18/07/18 04:30:25 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1531862573419_0001/
18/07/18 04:30:25 INFO mapreduce.Job: Running job: job_1531862573419_0001
```

Below screenshot shows the output directory created at HDFS after the job execution-



```
[acadgild@localhost Assignment-5]$ hadoop fs -ls /uniquelistener
18/07/18 04:37:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your pla
Found 3 items
-rw-r--r--   1 acadgild supergroup          0 2018-07-18 04:32 /uniquelistener/_SUCCESS
-rw-r--r--   1 acadgild supergroup          0 2018-07-18 04:32 /uniquelistener/part-r-00000
-rw-r--r--   1 acadgild supergroup         27 2018-07-18 04:32 /uniquelistener/part-r-00001
[acadgild@localhost Assignment-5]$
```

Below screenshot shows the final output-



```
[acadgild@localhost Assignment-5]$ hadoop fs -cat /uniquelistener/*
18/07/18 04:39:03 WARN util.NativeCodeLoader: Unable to load native-ha
111113  1
111115  2
111117  1
[acadgild@localhost Assignment-5]$
```

<u>TASK-2- What are the number of times a song was heard fully.</u>

<u>Below is the mapper code for the same task-</u>

```java
package task2;

import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;

public class FullSongMapper extends Mapper<LongWritable,Text,Text,IntWritable> {

      public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException{

            String[] line = value.toString().split("\\|");
            Text uid = new Text(line[0]);
            if(line[4].equals("1")){
                  context.write(uid,new IntWritable(1));
            }
      }
}
```

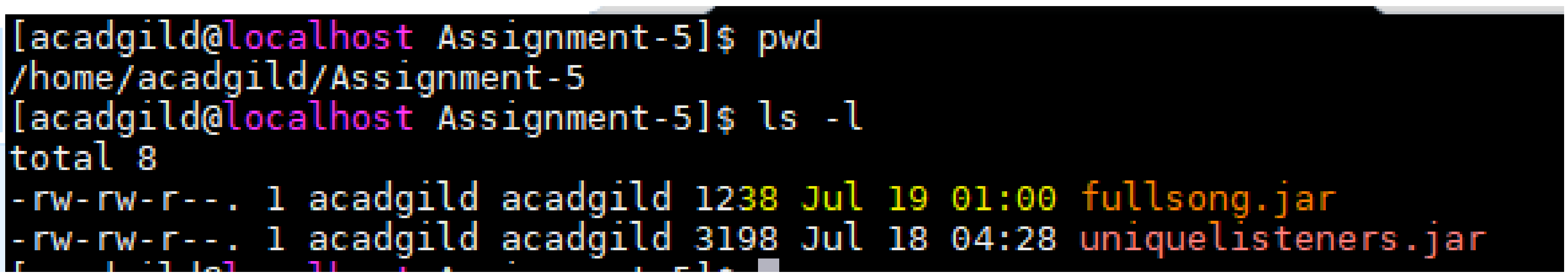
<u>Below is the reducer code for the same task-</u>

```java
package task2;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class FullSongReducer extends Reducer<Text,IntWritable,Text,IntWritable>{
	public void reduce(Text uid, Iterable<IntWritable> count, Context context) throws
IOException,InterruptedException{
		
		int sum = 0;
		for(IntWritable value: count){
			sum+= value.get();
		}
		context.write(uid,new IntWritable(sum));
	}
	
}
```

Below is the main driver code for the same task-

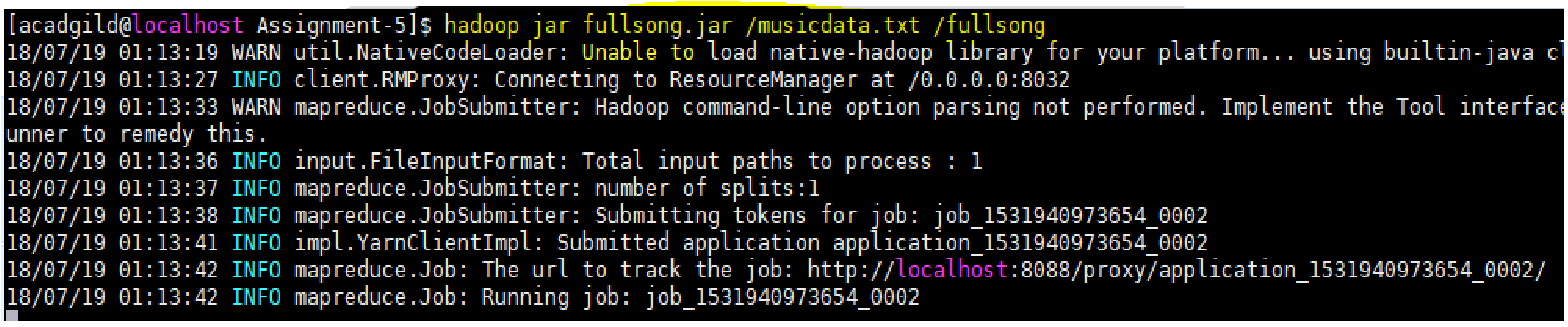
```
package task2;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class FullSong {

	public static void main(String[] args) throws Exception {
		
		Configuration conf = new Configuration();
		Job job = new Job(conf, "Assignment 5:task-2");
		job.setJarByClass(FullSong.class);
		
		job.setMapOutputKeyClass(Text.class);
		job.setMapOutputValueClass(IntWritable.class);
		
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);
		
		job.setMapperClass(FullSongMapper.class);
		job.setReducerClass(FullSongReducer.class);
		
		job.setNumReduceTasks(2);
		job.setInputFormatClass(TextInputFormat.class);
		job.setOutputFormatClass(TextOutputFormat.class);
		
		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job,new Path(args[1]));
		
		job.waitForCompletion(true);
	}
}
```

Below screenshot shows the jar created by the Map Reduce code-



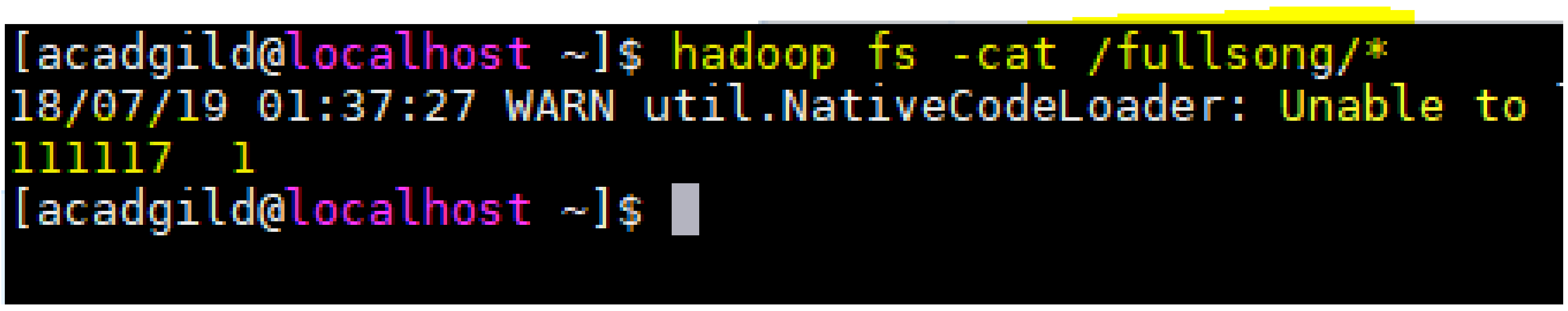
```
[acadgild@localhost Assignment-5]$ pwd
/home/acadgild/Assignment-5
[acadgild@localhost Assignment-5]$ ls -l
total 8
-rw-rw-r--. 1 acadgild acadgild 1238 Jul 19 01:00 fullsong.jar
-rw-rw-r--. 1 acadgild acadgild 3198 Jul 18 04:28 uniquelisteners.jar
```

Then we will run the mapreduce job with the jar created by Map Reduce Code as shown below-



```
[acadgild@localhost Assignment-5]$ hadoop jar fullsong.jar /musicdata.txt /fullsong
18/07/19 01:13:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
18/07/19 01:13:27 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
18/07/19 01:13:33 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface
unner to remedy this.
18/07/19 01:13:36 INFO input.FileInputFormat: Total input paths to process : 1
18/07/19 01:13:37 INFO mapreduce.JobSubmitter: number of splits:1
18/07/19 01:13:38 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1531940973654_0002
18/07/19 01:13:41 INFO impl.YarnClientImpl: Submitted application application_1531940973654_0002
18/07/19 01:13:42 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1531940973654_0002/
18/07/19 01:13:42 INFO mapreduce.Job: Running job: job_1531940973654_0002
```

Below screenshot shows the final output-



```
[acadgild@localhost ~]$ hadoop fs -cat /fullsong/*
18/07/19 01:37:27 WARN util.NativeCodeLoader: Unable to
111117  1
[acadgild@localhost ~]$
```

TASK-3- What are the number of times a song was shared.

Below code shows the mapper code for the same task-

```
package task3;

import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;

public class SongShareMapper extends Mapper<LongWritable,Text,Text,IntWritable> {

	public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException{
		String[] line = value.toString().split("\\|");
		if(line[2].equals("1")){
			context.write(new Text(line[0]), new IntWritable(1));
		}
	}

}
```

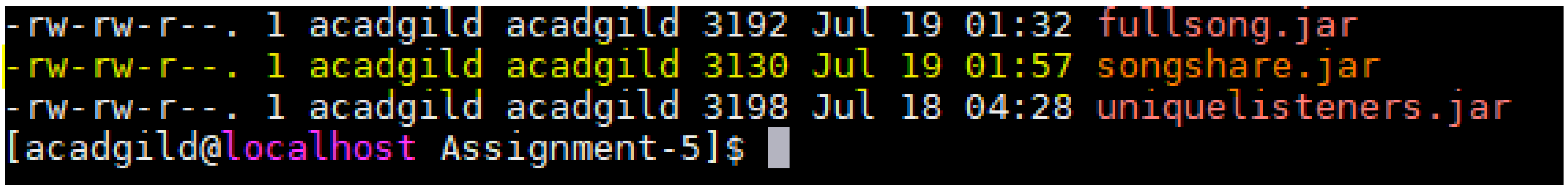
Below code shows the reducer code for the same task-

```
package task3;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;


public class SongShareReducer extends Reducer<Text,IntWritable,Text, IntWritable> {

	public void reducer(Text uid, Iterable<IntWritable> count, Context context) throws IOException,
InterruptedException{

		int sum = 0;
		for(IntWritable value:count){
			sum+= value.get();
		}
		context.write(uid, new IntWritable(sum));
	}
}
```

Below code shows the reducer code for the same task-

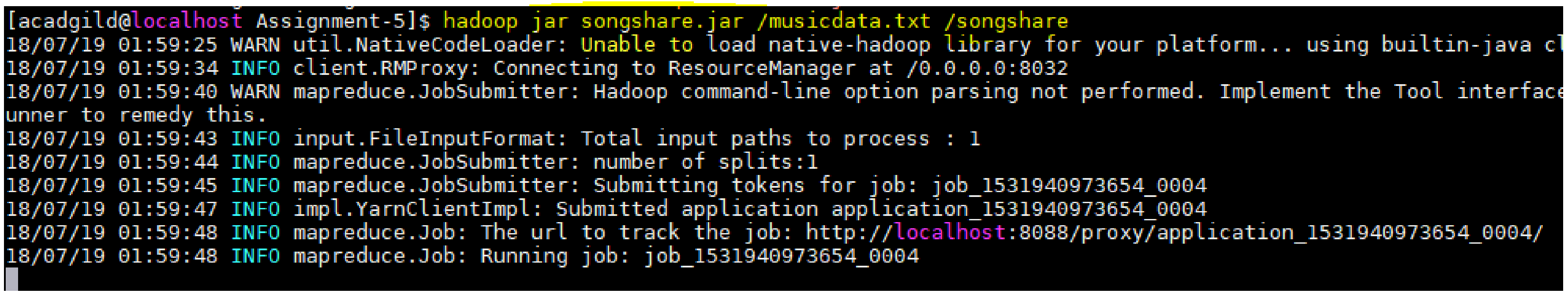
```
package task3;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import task2.FullSong;
import task2.FullSongMapper;
import task2.FullSongReducer;

public class SongShare {

	public static void main(String[] args) throws Exception {
		// TODO Auto-generated method stub
		Configuration conf = new Configuration();
		Job job = new Job(conf, "Assignment 5:task-3");
		job.setJarByClass(SongShare.class);

		job.setMapOutputKeyClass(Text.class);
		job.setMapOutputValueClass(IntWritable.class);

		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);

		job.setMapperClass(SongShareMapper.class);
		job.setReducerClass(SongShareReducer.class);

		job.setNumReduceTasks(2);
		job.setInputFormatClass(TextInputFormat.class);
		job.setOutputFormatClass(TextOutputFormat.class);

		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job,new Path(args[1]));

		job.waitForCompletion(true);
	}
}
```

Below screenshot shows the jar created by the Map Reduce code-



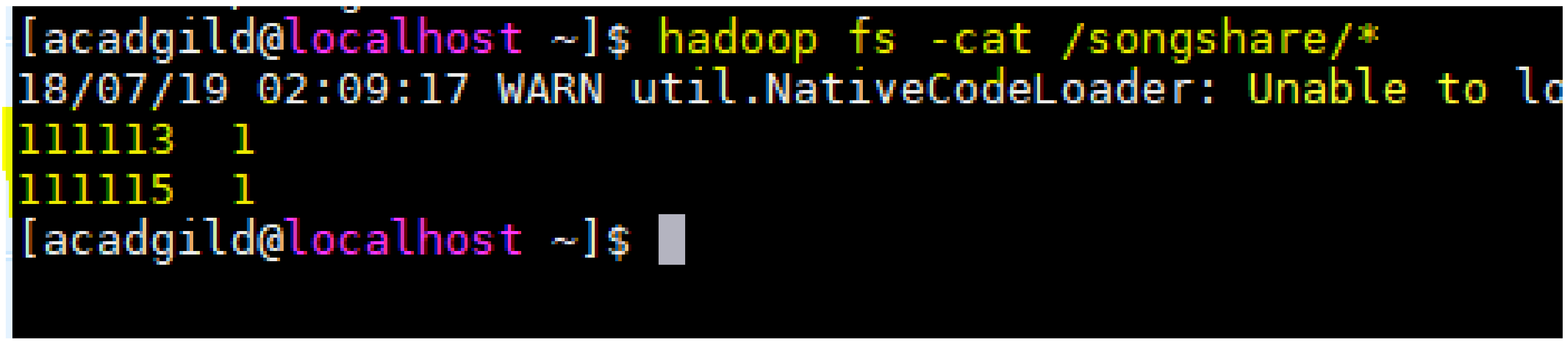
```
-rw-rw-r--. 1 acadgild acadgild 3192 Jul 19 01:32 fullsong.jar
-rw-rw-r--. 1 acadgild acadgild 3130 Jul 19 01:57 songshare.jar
-rw-rw-r--. 1 acadgild acadgild 3198 Jul 18 04:28 uniquelisteners.jar
[acadgild@localhost Assignment-5]$
```

Then we will run the mapreduce job with the jar created by Map Reduce Code as shown below-



```
[acadgild@localhost Assignment-5]$ hadoop jar songshare.jar /musicdata.txt /songshare
18/07/19 01:59:25 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
18/07/19 01:59:34 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
18/07/19 01:59:40 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface
unner to remedy this.
18/07/19 01:59:43 INFO input.FileInputFormat: Total input paths to process : 1
18/07/19 01:59:44 INFO mapreduce.JobSubmitter: number of splits:1
18/07/19 01:59:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1531940973654_0004
18/07/19 01:59:47 INFO impl.YarnClientImpl: Submitted application application_1531940973654_0004
18/07/19 01:59:48 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1531940973654_0004/
18/07/19 01:59:48 INFO mapreduce.Job: Running job: job_1531940973654_0004
```

Below screenshot shows the final output-



```
[acadgild@localhost ~]$ hadoop fs -cat /songshare/*
18/07/19 02:09:17 WARN util.NativeCodeLoader: Unable to lo
111113  1
111115  1
[acadgild@localhost ~]$
```