## 1. Clase Mapper

La clase Mapper procesará las líneas de entrada y emitirá pares clave-valor donde la clave será el aeropuerto y el valor será 1.

```
package FlightCount;

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

public class FlightMapper extends MapReduceBase implements
Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value,
OutputCollector<Text, IntWritable> output, Reporter reporter) throws
IOException {
        String valueString = value.toString();
        String[] flightData = valueString.split(",");
        output.collect(new Text(flightData[2]), one); // Suponiendo
que la columna 2 es el aeropuerto
    }
}
```

## 2. Clase Reducer

La clase Reducer agrupará los valores por clave (aeropuerto) y sumará los valores.

```
package FlightCount;

import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class FlightReducer extends MapReduceBase implements
Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values,
OutputCollector<Text, IntWritable> output, Reporter reporter) throws
IOException {
        int count = 0;
        while (values.hasNext()) {
            count += values.next().get();
        }
        output.collect(key, new IntWritable(count));
    }
}
```

## 3. Clase Driver

La clase Driver configurará y ejecutará el trabajo MapReduce.

```
package FlightCount;

import org.apache.hadoop.fs.Path;
import.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class FlightCountDriver {
    public static void main(String[] args) {
        JobClient my_client = new JobClient();
        JobConf job_conf = new JobConf(FlightCountDriver.class);

        job_conf.setJobName("FlightCountPerAirport");
        job_conf.setOutputKeyClass(Text.class);
        job_conf.setOutputValueClass(IntWritable.class);

        job_conf.setMapperClass(FlightMapper.class);
        job_conf.setReducerClass(FlightReducer.class);

        job_conf.setInputFormat(TextInputFormat.class);
        job_conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

        my_client.setConf(job_conf);
        try {
            JobClient.runJob(job_conf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Ejemplo de archivo CSV

Aquí tienes un ejemplo de cómo podría verse un archivo CSV con datos sobre itinerarios y vuelos:

```
FlightID,Date,Airport,Destination,Duration
1,2025-01-01,JFK,LAX,6
2,2025-01-02,LAX,JFK,6
3,2025-01-03,JFK,SFO,5
4,2025-01-04,SFO,JFK,5
5,2025-01-05,LAX,SFO,1
6,2025-01-06,SFO,LAX,1
7,2025-01-07,JFK,ORD,2
8,2025-01-08,ORD,JFK,2
9,2025-01-09,LAX,ORD,4
10,2025-01-10,ORD,LAX,4
```

Guarda este contenido en un archivo llamado `flights_data.csv` y úsalo como entrada para tu trabajo de MapReduce.

## Crear el archivo JAR y ejecutar el trabajo

Sigue los mismos pasos que antes para compilar las clases, crear el archivo JAR y ejecutar el trabajo MapReduce:

1. Compilar las clases Java:

```
2. javac -classpath $(hadoop classpath) -d . FlightCount/*.java
```

3. Crear el archivo JAR:

```
4. jar -cvf FlightCount.jar -C . FlightCount
```

5. Ejecutar el trabajo MapReduce:

```
hadoop jar FlightCount.jar FlightCount.FlightCountDriver
/input/flights_data.csv /output
```