MapReduce 程序实现步骤：

Mapper 部分：

Mapper 的作用是将输入的文本分割成单词，并为每个单词输出一个键值对 (单词, 1)。

```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

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

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
}
```

Reducer 部分：

Reducer 接收到 Mapper 输出的键值对 (单词, [1, 1, 1, ...])，将相同单词的计数求和，输出结果为 (单词, 总计数)。

```java
import java.io.IOException;

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

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

    private IntWritable result = new IntWritable();

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

主程序（Driver）部分：

主程序负责设置并运行 MapReduce 作业。

```java
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.output.FileOutputFormat;

public class WordCount {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");

        job.setJarByClass(WordCount.class);
        job.setMapperClass(WordCountMapper.class);
        job.setCombinerClass(WordCountReducer.class);
        job.setReducerClass(WordCountReducer.class);

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

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

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

如何运行程序：

1. **准备输入文件：**将包含文本的文件放置在 Hadoop 文件系统中的适当位置。
2. **编译程序：**将以上代码编译为 JAR 文件。
3. **运行作业：**使用 `hadoop jar` 命令提交作业。

   ```bash
   hadoop jar WordCount.jar WordCount input_path output_path
   ```

   其中，`WordCount.jar` 是你编译的 JAR 文件，`input_path` 是输入文件或文件夹的路径，`output_path` 是输出结果的路径。
4. **查看结果：**作业完成后，可以查看输出路径中的结果文件，以查看每个单词的词频总和。