

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.conf.Configured;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.mapreduce.Job;
9 import org.apache.hadoop.mapreduce.lib.input.
10    FileInputFormat;
11 import org.apache.hadoop.mapreduce.lib.input.
12    TextInputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.
14    FileOutputFormat;
15 import org.apache.hadoop.mapreduce.lib.output.
16    TextOutputFormat;
17 import org.apache.hadoop.util.Tool;
18 import org.apache.hadoop.util.ToolRunner;
19
20 /**
21  * Implements the - Driver class
22  */
23 public class WordCount extends Configured implements
24   Tool {
25
26   public int run(String[] args) throws Exception {
27     int numberReducers = Integer.parseInt(args[0]);
28     Path inputDir = new Path(args[1]);
29     Path output = new Path(args[2]);
30     String wordLenLimit = args[3];
31
32     Configuration conf = getConf();
33     conf.setInt("wordLenLimit", Integer.parseInt(
34       wordLenLimit));
35
36     Job job = Job.getInstance(conf);
37
38     job.setJobName("WordCount");
39
40   }
41 }
```

```
35         FileInputFormat.addInputPath(job, inputDir);
36         FileOutputFormat.setOutputPath(job, output);
37
38         job.setInputFormatClass(TextInputFormat.class);
39         job.setOutputValueClass(TextOutputFormat.
40             class);
41         job.setJarByClass(WordCount.class);
42
43         // Init mapper
44         job.setMapperClass(WordCountMapper.class);
45         job.setMapOutputKeyClass(Text.class);
46         job.setMapOutputValueClass(IntWritable.class);
47
48         // Init reducer
49         job.setReducerClass(WordCountReducer.class);
50         job.setOutputKeyClass(Text.class);
51         job.setOutputValueClass(IntWritable.class);
52
53         job.setNumReduceTasks(numberOfReducers);
54
55         if (job.waitForCompletion(true))
56             return 0;
57
58         return 1;
59     }
60
61     public static void main(String[] args) throws
62         Exception {
63         int res = ToolRunner.run(new Configuration
64             (), new WordCount(), args);
65         System.exit(res);
66     }
67 }
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.conf.Configured;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapreduce.Job;
8 import org.apache.hadoop.mapreduce.lib.input.
  FileInputFormat;
9 import org.apache.hadoop.mapreduce.lib.input.
  TextInputFormat;
10 import org.apache.hadoop.mapreduce.lib.output.
   FileOutputFormat;
11 import org.apache.hadoop.mapreduce.lib.output.
   TextOutputFormat;
12 import org.apache.hadoop.util.Tool;
13 import org.apache.hadoop.util.ToolRunner;
14
15 public class DailyTopApp extends Configured
16   implements Tool {
17
18   @Override
19   public int run(String[] args) throws Exception {
20     int reducers = Integer.parseInt(args[0]);
21     Path inputPath = new Path(args[1]);
22     Path outputPath = new Path(args[2]);
23
24     Job job = Job.getInstance(getConf());
25     job.setJobName("DailyTopApp");
26     job.setJarByClass(DailyTopApp.class);
27
28     FileInputFormat.addInputPath(job, inputPath);
29     FileOutputFormat.setOutputPath(job,
30       outputPath);
31
32     job.setInputFormatClass(TextInputFormat.class
33   );
34     job.setOutputFormatClass(TextOutputFormat.
35   class);
36
37     job.setMapperClass(DailyTopAppMapper.class);
```

```
34         job.setMapOutputKeyClass(Text.class);
35         job.setMapOutputValueClass(Text.class);
36
37         job.setReducerClass(DailyTopAppReducer.class
38 );
38         job.setOutputKeyClass(Text.class);
39         job.setOutputValueClass(Text.class);
40
41         job.setNumReduceTasks(reducers);
42
43         return job.waitForCompletion(true) ? 0 : 1;
44     }
45
46     public static void main(String[] args) throws
Exception {
47         int res = ToolRunner.run(new Configuration
(), new DailyTopApp(), args);
48         System.exit(res);
49     }
50 }
51
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.conf.Configured;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.mapreduce.Job;
9 import org.apache.hadoop.mapreduce.lib.input.
10    FileInputFormat;
11 import org.apache.hadoop.mapreduce.lib.input.
12    TextInputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.
14    FileOutputFormat;
15 import org.apache.hadoop.mapreduce.lib.output.
16    TextOutputFormat;
17 import org.apache.hadoop.util.Tool;
18 import org.apache.hadoop.util.ToolRunner;
19
20 /**
21  * Driver class for: total usage (minutes) per app.
22  * Args:
23  *   [0] = number of reducers
24  *   [1] = input path (CSV file or folder in HDFS)
25  *   [2] = output path (folder in HDFS, must NOT
26  *         exist)
27 */
28 public class TopAppsByUsage extends Configured
29 implements Tool {
30
31     @Override
32     public int run(String[] args) throws Exception {
33
34         int numberReducers = Integer.parseInt(args[0]);
35         Path inputPath = new Path(args[1]);
36         Path outputPath = new Path(args[2]);
37
38         Configuration conf = getConf();
39         Job job = Job.getInstance(conf);
40
41         job.setJarByClass(TopAppsByUsage.class);
42         job.setMapperClass(Map.class);
43         job.setReducerClass(Reduce.class);
44
45         job.setOutputKeyClass(Text.class);
46         job.setOutputValueClass(IntWritable.class);
47
48         job.setInputFormatClass(TextInputFormat.class);
49         job.setOutputFormatClass(TextOutputFormat.class);
50
51         TextInputFormat.setInputPaths(job, inputPath);
52         TextOutputFormat.setOutputPath(job, outputPath);
53
54         return job.waitForCompletion(true) ? 0 : 1;
55     }
56 }
```

```
35         job.setJobName("TopAppsByUsage");
36         job.setJarByClass(TopAppsByUsage.class);
37
38         FileInputFormat.addInputPath(job, inputPath);
39         FileOutputFormat.setOutputPath(job,
40             outputPath);
41         job.setInputFormatClass(TextInputFormat.class);
42         job.setOutputFormatClass(TextOutputFormat.
43             class);
44         job.setMapperClass(TopAppsByUsageMapper.class);
45         job.setMapOutputKeyClass(Text.class);
46         job.setMapOutputValueClass(IntWritable.class);
47
48
49         job.setReducerClass(TopAppsByUsageReducer.
50             class);
51         job.setOutputKeyClass(Text.class);           // app name
52         job.setOutputValueClass(IntWritable.class); // total usage minutes
53
54
55         boolean success = job.waitForCompletion(true);
56
57     }
58
59     public static void main(String[] args) throws
Exception {
60         int res = ToolRunner.run(new Configuration
(), new TopAppsByUsage(), args);
61         System.exit(res);
62     }
63 }
64
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.io.IntWritable;
4 import org.apache.hadoop.io.LongWritable;
5 import org.apache.hadoop.io.Text;
6 import org.apache.hadoop.mapreduce.Mapper;
7
8 import java.io.IOException;
9
10 public class WordCountMapper extends Mapper<
11     LongWritable, Text, Text, IntWritable> {
12     @Override
13     protected void map(LongWritable key, Text value,
14         Mapper<LongWritable, Text, Text, IntWritable>.Context
15         context) throws IOException, InterruptedException {
16         String[] values = value.toString().split("\n");
17         int wordLenLimit = context.getConfiguration()
18             .getInt("wordLenLimit", 0);
19         for (String w : values) {
20             // emit (w, 1)
21             if (w.length() >= wordLenLimit) {
22                 context.write(new Text(w.toLowerCase()),
23                             new IntWritable(1));
24             }
25         }
26     }
27 }
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.io.IntWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Reducer;
6
7 import java.io.IOException;
8
9 public class WordCountReducer extends Reducer<Text,
10     IntWritable, Text, IntWritable> {
11     @Override
12     protected void reduce(Text key, Iterable<
13         IntWritable> values, Reducer<Text, IntWritable, Text
14         , IntWritable>.Context context) throws IOException,
15     InterruptedException {
16         int sum = 0;
17         for (IntWritable value : values) {
18             sum += value.get();
19         }
20         context.write(key, new IntWritable(sum));
21     }
22 }
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.io.LongWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Mapper;
6
7 import java.io.IOException;
8
9 /**
10  * Output: key = date, value = "AppName\tTimesOpened"
11 */
12 public class DailyTopAppMapper
13         extends Mapper<LongWritable, Text, Text, Text> {
14
15     private final Text outKey = new Text();
16     private final Text outValue = new Text();
17
18     @Override
19     protected void map(LongWritable key, Text value,
20                        Context context)
21                     throws IOException, InterruptedException
22     {
23         String line = value.toString().trim();
24         if (line.isEmpty() || line.startsWith("Date,")) {
25             return;
26         }
27
28         String[] parts = line.split(",");
29         if (parts.length < 5) {
30             return;
31         }
32
33         String date = parts[0].trim();
34         String app = parts[1].trim();
35         String timesStr = parts[4].trim();
36
37         outKey.set(date);
38         outValue.set(app + "\t" + timesStr);
```

```
38         context.write(outKey, outValue);  
39     }  
40 }  
41
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.conf.Configured;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.mapreduce.Job;
9 import org.apache.hadoop.mapreduce.lib.input.
10    FileInputFormat;
11 import org.apache.hadoop.mapreduce.lib.input.
12    TextInputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.
14    FileOutputFormat;
15 import org.apache.hadoop.mapreduce.lib.output.
16    TextOutputFormat;
17 import org.apache.hadoop.util.Tool;
18 import org.apache.hadoop.util.ToolRunner;
19
20 public class LaunchesByWeekday extends Configured
21     implements Tool {
22
23     @Override
24     public int run(String[] args) throws Exception {
25         int reducers = Integer.parseInt(args[0]);
26         Path inputPath = new Path(args[1]);
27         Path outputPath = new Path(args[2]);
28
29         Job job = Job.getInstance(getConf());
30         job.setJobName("LaunchesByWeekday");
31         job.setJarByClass(LaunchesByWeekday.class);
32
33         FileInputFormat.addInputPath(job, inputPath);
34         FileOutputFormat.setOutputPath(job,
35             outputPath);
36
37         job.setInputFormatClass(TextInputFormat.class);
38         job.setOutputFormatClass(TextOutputFormat.
39             class);
40
41     }
42 }
```

```
34         job.setMapperClass(LaunchesByWeekdayMapper.  
35             class);  
36         job.setMapOutputKeyClass(Text.class);  
37         job.setMapOutputValueClass(IntWritable.class  
38 );  
39         job.setReducerClass(LaunchesByWeekdayReducer.  
40             class);  
41         job.setOutputKeyClass(Text.class);  
42         job.setOutputValueClass(IntWritable.class);  
43         job.setNumReduceTasks(reducers);  
44     return job.waitForCompletion(true) ? 0 : 1;  
45 }  
46  
47 public static void main(String[] args) throws  
Exception {  
48     int res = ToolRunner.run(new Configuration(),  
49             new LaunchesByWeekday(), args);  
50     System.exit(res);  
51 }  
52 }  
53 }
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.io.Text;
4 import org.apache.hadoop.mapreduce.Reducer;
5
6 import java.io.IOException;
7 import java.util.HashMap;
8 import java.util.Map;
9
10 /**
11  * For each date, sums Times Opened per app, then
12  * selects the app with max total.
13  * Input values: "AppName\tTimesOpened"
14  * Output: date "AppName (TOTAL times)"
15 */
16 public class DailyTopAppReducer extends Reducer<Text,
17 , Text, Text, Text> {
18
19     private final Text outValue = new Text();
20
21     @Override
22     protected void reduce(Text key, Iterable<Text>
23     values, Context context)
24         throws IOException, InterruptedException
25     {
26
27         // 1) Sum timesOpened per app for this date
28         Map<String, Integer> totalsByApp = new
29         HashMap<>();
30
31         for (Text v : values) {
32             String[] parts = v.toString().split("\t");
33             if (parts.length != 2) continue;
34
35             String app = parts[0].trim();
36             int times;
37             try {
38                 times = Integer.parseInt(parts[1].
39                 trim());
40             } catch (NumberFormatException e) {
```

```
35             continue;  
36     }  
37  
38     totalsByApp.put(app, totalsByApp.  
39         getOrDefault(app, 0) + times);  
40     }  
41  
42     // 2) Find the app with the maximum total  
43     String bestApp = null;  
44     int bestTotal = -1;  
45  
46     for (Map.Entry<String, Integer> entry :  
47         totalsByApp.entrySet()) {  
48         if (entry.getValue() > bestTotal) {  
49             bestTotal = entry.getValue();  
50             bestApp = entry.getKey();  
51         }  
52     }  
53  
54     // 3) Output result  
55     if (bestApp != null) {  
56         outValue.set(bestApp + " (" + bestTotal  
57         + " times));  
58         context.write(key, outValue);  
59     }  
59 }
```

```
1 package edu.univ.haifa.bigdata;
2
3 import java.io.IOException;
4
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.LongWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.mapreduce.Mapper;
9
10 /**
11  * Mapper for TopAppsByUsage.
12  * Input: one CSV line -> "Date,App,Usage (minutes),Notifications,Times Opened"
13  * Output: (app, usageMinutes)
14 */
15 public class TopAppsByUsageMapper extends Mapper<
16     LongWritable, Text, Text, IntWritable> {
17     private Text outKey = new Text();
18     private IntWritable outValue = new IntWritable();
19
20     @Override
21     protected void map(LongWritable key, Text value,
22         Context context)
23             throws IOException, InterruptedException
24     {
25         String line = value.toString().trim();
26         if (line.isEmpty()) {
27             return; // skip empty lines
28         }
29         // Skip header line (starts with "Date,App
30         ,..."))
31         if (line.startsWith("Date,")) {
32             return;
33         }
34         String[] parts = line.split(",");
35         // Basic sanity check
```

```
37     if (parts.length < 3) {
38         return; // malformed line
39     }
40
41     String app = parts[1].trim();
42     String usageStr = parts[2].trim();
43
44     try {
45         int usageMinutes = Integer.parseInt(
46             usageStr);
47         outKey.set(app);
48         outValue.set(usageMinutes);
49         context.write(outKey, outValue);
50
51     } catch (NumberFormatException e) {
52         // bad number, just skip this record
53     }
54 }
55 }
56 }
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.conf.Configured;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapreduce.Job;
8 import org.apache.hadoop.mapreduce.lib.input.
  FileInputFormat;
9 import org.apache.hadoop.mapreduce.lib.input.
  TextInputFormat;
10 import org.apache.hadoop.mapreduce.lib.output.
   FileOutputFormat;
11 import org.apache.hadoop.mapreduce.lib.output.
   TextOutputFormat;
12 import org.apache.hadoop.util.Tool;
13 import org.apache.hadoop.util.ToolRunner;
14
15 public class UsageVsNotifications extends Configured
  implements Tool {
16
17     @Override
18     public int run(String[] args) throws Exception {
19         int reducers = Integer.parseInt(args[0]);
20         Path inputPath = new Path(args[1]);
21         Path outputPath = new Path(args[2]);
22
23         Job job = Job.getInstance(getConf());
24         job.setJobName("UsageVsNotifications");
25         job.setJarByClass(UsageVsNotifications.class
26 );
27         FileInputFormat.addInputPath(job, inputPath);
28         FileOutputFormat.setOutputPath(job,
29             outputPath);
30         job.setInputFormatClass(TextInputFormat.class
31 );
31         job.setOutputFormatClass(TextOutputFormat.
32             class);
```

```
33         job.setMapperClass(UsageVsNotificationsMapper
 .class);
34         job.setMapOutputKeyClass(Text.class);
35         job.setMapOutputValueClass(Text.class);
36
37         job.setReducerClass(
38             UsageVsNotificationsReducer.class);
39         job.setOutputKeyClass(Text.class);
40         job.setOutputValueClass(Text.class);
41
42         job.setNumReduceTasks(reducers);
43
44     }
45
46     public static void main(String[] args) throws
Exception {
47         int res = ToolRunner.run(new Configuration(),
48             new UsageVsNotifications(), args);
49         System.exit(res);
50     }
51 }
52
```

```
1 package edu.univ.haifa.bigdata;
2
3 import java.io.IOException;
4
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapreduce.Reducer;
8
9 /**
10  * Reducer for TopAppsByUsage.
11  * For each app, sums all usage minutes.
12  * Input: (app, [usage1, usage2, ...])
13  * Output: (app, totalUsageMinutes)
14 */
15 public class TopAppsByUsageReducer extends Reducer<
16     Text, IntWritable, Text, IntWritable> {
17
18     private IntWritable outValue = new IntWritable();
19
20     @Override
21     protected void reduce(Text key, Iterable<
22         IntWritable> values, Context context)
23             throws IOException, InterruptedException
24     {
25
26         int sum = 0;
27         for (IntWritable v : values) {
28             sum += v.get();
29         }
30
31         outValue.set(sum);
32         context.write(key, outValue);
33     }
34 }
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.io.IntWritable;
4 import org.apache.hadoop.io.LongWritable;
5 import org.apache.hadoop.io.Text;
6 import org.apache.hadoop.mapreduce.Mapper;
7
8 import java.io.IOException;
9 import java.time.DayOfWeek;
10 import java.time.LocalDate;
11
12 /**
13 * Converts Date -> DayOfWeek and outputs (dayOfWeek,
14 , TimesOpened).
15 */
16 public class LaunchesByWeekdayMapper
17     extends Mapper<LongWritable, Text, Text,
18 IntWritable> {
19
20     private final Text outKey = new Text();
21     private final IntWritable outValue = new
22 IntWritable();
23
24     @Override
25     protected void map(LongWritable key, Text value,
26 Context context)
27         throws IOException, InterruptedException
28     {
29
30         String line = value.toString().trim();
31         if (line.isEmpty() || line.startsWith("Date,")) {
32             return;
33         }
34
35         String dateStr = parts[0].trim();
```

```
36         String timesStr = parts[4].trim();
37
38     try {
39         LocalDate date = LocalDate.parse(dateStr
40 ); // expects YYYY-MM-DD
41         DayOfWeek dow = date.getDayOfWeek
42 (); // MONDAY, TUESDAY, ...
43         String dayName = dow.toString
44 (); // UPPERCASE; fine for output
45
46         int timesOpened = Integer.parseInt(
47 timesStr);
48
49         outKey.set(dayName);
50         outValue.set(timesOpened);
51         context.write(outKey, outValue);
52     } catch (Exception ignored) {
53     }
54 }
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.io.IntWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Reducer;
6
7 import java.io.IOException;
8
9 /**
10  * Sums launches per day of week.
11 */
12 public class LaunchesByWeekdayReducer
13     extends Reducer<Text, IntWritable, Text,
14     IntWritable> {
15
16     private final IntWritable outValue = new
17     IntWritable();
18
19     @Override
20     protected void reduce(Text key, Iterable<
21     IntWritable> values, Context context)
22         throws IOException, InterruptedException
23     {
24
25         int sum = 0;
26         for (IntWritable v : values) {
27             sum += v.get();
28         }
29
30         outValue.set(sum);
31         context.write(key, outValue);
32     }
33 }
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.conf.Configured;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.FloatWritable;
7 import org.apache.hadoop.io.IntWritable;
8 import org.apache.hadoop.io.Text;
9 import org.apache.hadoop.mapreduce.Job;
10 import org.apache.hadoop.mapreduce.lib.input.
    FileInputFormat;
11 import org.apache.hadoop.mapreduce.lib.input.
    TextInputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.
    FileOutputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.
    TextOutputFormat;
14 import org.apache.hadoop.util.Tool;
15 import org.apache.hadoop.util.ToolRunner;
16
17 public class AverageNotificationsPerApp extends
    Configured implements Tool {
18
19     @Override
20     public int run(String[] args) throws Exception {
21         int numberReducers = Integer.parseInt(args[
22             0]);
22         Path inputPath = new Path(args[1]);
23         Path outputPath = new Path(args[2]);
24
25         Job job = Job.getInstance(getConf());
26         job.setJobName("AverageNotificationsPerApp");
27         job.setJarByClass(AverageNotificationsPerApp.
28             class);
29
30         FileInputFormat.addInputPath(job, inputPath);
31         FileOutputFormat.setOutputPath(job,
32             outputPath);
32         job.setInputFormatClass(TextInputFormat.class
33     );
```

```
33         job.setOutputFormatClass(TextOutputFormat.  
34             class);  
35         job.setMapperClass(  
36             AverageNotificationsPerAppMapper.class);  
37         job.setMapOutputKeyClass(Text.class);  
38         job.setMapOutputValueClass(IntWritable.class)  
39     );  
40         job.setReducerClass(  
41             AverageNotificationsPerAppReducer.class);  
42         job.setOutputKeyClass(Text.class);  
43         job.setOutputValueClass(FloatWritable.class);  
44  
45         job.setNumReduceTasks(numberOfReducers);  
46     }  
47  
48     public static void main(String[] args) throws  
Exception {  
49         int res = ToolRunner.run(new Configuration(),  
50             new AverageNotificationsPerApp(),  
args);  
51         System.exit(res);  
52     }  
53 }  
54
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.io.LongWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Mapper;
6
7 import java.io.IOException;
8
9 /**
10  * Output: key = app, value = "usage,notifications"
11 */
12 public class UsageVsNotificationsMapper
13         extends Mapper<LongWritable, Text, Text, Text> {
14
15     private final Text outKey = new Text();
16     private final Text outValue = new Text();
17
18     @Override
19     protected void map(LongWritable key, Text value,
20                        Context context)
21                     throws IOException, InterruptedException
22     {
23         String line = value.toString().trim();
24         if (line.isEmpty() || line.startsWith("Date,")) {
25             return;
26         }
27
28         String[] parts = line.split(",");
29         if (parts.length < 4) {
30             return;
31         }
32
33         String app = parts[1].trim();
34         String usageStr = parts[2].trim();
35         String notifStr = parts[3].trim();
36
37         outKey.set(app);
38         outValue.set(usageStr + "," + notifStr);
```

```
38         context.write(outKey, outValue);
39     }
40 }
41
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.io.Text;
4 import org.apache.hadoop.mapreduce.Reducer;
5
6 import java.io.IOException;
7
8 /**
9  * For each app, sums total usage and total
10 notifications.
11 * Output value format: "totalUsage\
12     totalNotifications"
13 */
14 public class UsageVsNotificationsReducer
15     extends Reducer<Text, Text, Text, Text> {
16
17     @Override
18     protected void reduce(Text key, Iterable<Text>
19     values, Context context)
20         throws IOException, InterruptedException
21     {
22
23         int totalUsage = 0;
24         int totalNotif = 0;
25
26         for (Text v : values) {
27             String[] parts = v.toString().split(",");
28             if (parts.length != 2) continue;
29
30             try {
31                 int usage = Integer.parseInt(parts[0]
32                     .trim());
33                 int notif = Integer.parseInt(parts[1]
34                     .trim());
35
36                 totalUsage += usage;
37                 totalNotif += notif;
38             } catch (NumberFormatException ignored) {
39             }
40         }
41     }
42 }
```

```
36
37         outValue.set(totalUsage + "\t" + totalNotif);
38         context.write(key, outValue);
39     }
40 }
41
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.io.IntWritable;
4 import org.apache.hadoop.io.LongWritable;
5 import org.apache.hadoop.io.Text;
6 import org.apache.hadoop.mapreduce.Mapper;
7
8 import java.io.IOException;
9
10 /**
11  * Input CSV: Date,App,Usage (minutes),Notifications,
12  * Times Opened
13  * Output: (app, notificationsForThatDay)
14 */
15 public class AverageNotificationsPerAppMapper
16     extends Mapper<LongWritable, Text, Text,
17     IntWritable> {
18
19     private final Text outKey = new Text();
20     private final IntWritable outValue = new
21     IntWritable();
22
23     @Override
24     protected void map(LongWritable key, Text value,
25     Context context)
26         throws IOException, InterruptedException
27     {
28
29         String line = value.toString().trim();
30         if (line.isEmpty() || line.startsWith("Date,")) {
31             return; // skip empty or header
32         }
33
34         String[] parts = line.split(",");
35         if (parts.length < 4) {  
            return;  
        }  

36
37         String app = parts[1].trim();
38         String notifStr = parts[3].trim();
```

```
36
37         try {
38             int notifications = Integer.parseInt(
39                 notifStr);
40             outKey.set(app);
41             outValue.set(notifications);
42             context.write(outKey, outValue);
43         } catch (NumberFormatException ignored) {
44     }
45 }
46
```

```
1 package edu.univ.haifa.bigdata;
2
3 import org.apache.hadoop.io.FloatWritable;
4 import org.apache.hadoop.io.IntWritable;
5 import org.apache.hadoop.io.Text;
6 import org.apache.hadoop.mapreduce.Reducer;
7
8 import java.io.IOException;
9
10 /**
11 * For each app, computes average notifications per day.
12 * Input: (app, [n1, n2, ...])
13 * Output: (app, avgNotificationsPerDay)
14 */
15 public class AverageNotificationsPerAppReducer
16     extends Reducer<Text, IntWritable, Text,
17     FloatWritable> {
18
19     private final FloatWritable outValue = new
20     FloatWritable();
21
22     @Override
23     protected void reduce(Text key, Iterable<
24     IntWritable> values, Context context)
25         throws IOException, InterruptedException
26     {
27
28         int sum = 0;
29         int count = 0;
30
31         for (IntWritable v : values) {
32             sum += v.get();
33             count++;
34         }
35
36         float avg = (float) sum / count;
```

```
37         outValue.set(avg);
38         context.write(key, outValue);
39     }
40 }
41
```