```java
package edu.univ.haifa.bigdata;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
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 org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;


/*
 * Implements the - Driver class
 * */
public class WordCount extends Configured implements
Tool {

    public int run(String[] args) throws Exception {
        int numberOfReducers = Integer.parseInt(args[
0]);
        Path inputDir = new Path(args[1]);
        Path output = new Path(args[2]);
        String wordLenLimit = args[3];

        Configuration conf = getConf();
        conf.setInt("wordLenLimit", Integer.parseInt(
wordLenLimit));

        Job job = Job.getInstance(conf);

        job.setJobName("WordCount");

```

```
35          FileInputFormat.addInputPath(job, inputDir);
36          FileOutputFormat.setOutputPath(job, output);
37
38          job.setInputFormatClass(TextInputFormat.class
    );
39          job.setOutputValueClass(TextOutputFormat.
    class);
40
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
    Exception {
62          int res = ToolRunner.run(new Configuration
    (), new WordCount(), args);
63          System.exit(res);
64      }
65  }
66
```

```java
package edu.univ.haifa.bigdata;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
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 org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class DailyTopApp extends Configured
implements Tool {

    @Override
    public int run(String[] args) throws Exception {
        int reducers = Integer.parseInt(args[0]);
        Path inputPath = new Path(args[1]);
        Path outputPath = new Path(args[2]);

        Job job = Job.getInstance(getConf());
        job.setJobName("DailyTopApp");
        job.setJarByClass(DailyTopApp.class);

        FileInputFormat.addInputPath(job, inputPath);
        FileOutputFormat.setOutputPath(job,
outputPath);

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

        job.setMapperClass(DailyTopAppMapper.class);
```

```
34          job.setMapOutputKeyClass(Text.class);
35          job.setMapOutputValueClass(Text.class);
36
37          job.setReducerClass(DailyTopAppReducer.class
    );
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
```

```java
package edu.univ.haifa.bigdata;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
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 org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/**
 * Driver class for: total usage (minutes) per app.
 * Args:
 *   [0] = number of reducers
 *   [1] = input path (CSV file or folder in HDFS)
 *   [2] = output path (folder in HDFS, must NOT
exist)
 */
public class TopAppsByUsage extends Configured
implements Tool {

    @Override
    public int run(String[] args) throws Exception {

        int numberOfReducers = Integer.parseInt(args[
0]);
        Path inputPath = new Path(args[1]);
        Path outputPath = new Path(args[2]);

        Configuration conf = getConf();
        Job job = Job.getInstance(conf);

```

```
35          job.setJobName("TopAppsByUsage");
36          job.setJarByClass(TopAppsByUsage.class);
37
38          FileInputFormat.addInputPath(job, inputPath);
39          FileOutputFormat.setOutputPath(job,
    outputPath);
40
41          job.setInputFormatClass(TextInputFormat.class
    );
42          job.setOutputFormatClass(TextOutputFormat.
    class);
43
44          job.setMapperClass(TopAppsByUsageMapper.class
    );
45          job.setMapOutputKeyClass(Text.class);
46          job.setMapOutputValueClass(IntWritable.class
    );
47
48
49          job.setReducerClass(TopAppsByUsageReducer.
    class);
50          job.setOutputKeyClass(Text.class);        //
    app name
51          job.setOutputValueClass(IntWritable.class);
    // total usage minutes
52
53          job.setNumReduceTasks(numberOfReducers);
54
55          boolean success = job.waitForCompletion(true
    );
56          return success ? 0 : 1;
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
```

```java
package edu.univ.haifa.bigdata;

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

import java.io.IOException;

public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {
        String[] values = value.toString().split("\\s+");

        int wordLenLimit = context.getConfiguration().getInt("wordLenLimit", 0);

        for (String w : values) {
            // emit (w, 1)
            if (w.length() >= wordLenLimit) {
                context.write(new Text(w.toLowerCase()), new IntWritable(1));
            }
        }
    }
}

```

```java
package edu.univ.haifa.bigdata;

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

import java.io.IOException;

public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable value : values) {
            sum += value.get();
        }
        context.write(key, new IntWritable(sum));
    }
}

```

```java
package edu.univ.haifa.bigdata;

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

import java.io.IOException;

/**
 * Output: key = date, value = "AppName\tTimesOpened"
 */
public class DailyTopAppMapper
        extends Mapper<LongWritable, Text, Text, Text> {

    private final Text outKey = new Text();
    private final Text outValue = new Text();

    @Override
    protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

        String line = value.toString().trim();
        if (line.isEmpty() || line.startsWith("Date,")) {
            return;
        }

        String[] parts = line.split(",");
        if (parts.length < 5) {
            return;
        }

        String date = parts[0].trim();
        String app = parts[1].trim();
        String timesStr = parts[4].trim();

        outKey.set(date);
        outValue.set(app + "\t" + timesStr);
```

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

```java
package edu.univ.haifa.bigdata;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
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 org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class LaunchesByWeekday extends Configured
implements Tool {

    @Override
    public int run(String[] args) throws Exception {
        int reducers = Integer.parseInt(args[0]);
        Path inputPath = new Path(args[1]);
        Path outputPath = new Path(args[2]);

        Job job = Job.getInstance(getConf());
        job.setJobName("LaunchesByWeekday");
        job.setJarByClass(LaunchesByWeekday.class);

        FileInputFormat.addInputPath(job, inputPath);
        FileOutputFormat.setOutputPath(job,
outputPath);

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

```

```
34          job.setMapperClass(LaunchesByWeekdayMapper.
   class);
35          job.setMapOutputKeyClass(Text.class);
36          job.setMapOutputValueClass(IntWritable.class
   );
37
38          job.setReducerClass(LaunchesByWeekdayReducer.
   class);
39          job.setOutputKeyClass(Text.class);
40          job.setOutputValueClass(IntWritable.class);
41
42          job.setNumReduceTasks(reducers);
43
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
```

```java
package edu.univ.haifa.bigdata;

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

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

/**
 * For each date, sums Times Opened per app, then selects the app with max total.
 * Input values: "AppName\tTimesOpened"
 * Output: date  "AppName (TOTAL times)"
 */
public class DailyTopAppReducer extends Reducer<Text, Text, Text, Text> {

    private final Text outValue = new Text();

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {

        // 1) Sum timesOpened per app for this date
        Map<String, Integer> totalsByApp = new HashMap<>();

        for (Text v : values) {
            String[] parts = v.toString().split("\t");
            if (parts.length != 2) continue;

            String app = parts[0].trim();
            int times;
            try {
                times = Integer.parseInt(parts[1].trim());
            } catch (NumberFormatException e) {
```

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

```java
package edu.univ.haifa.bigdata;

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.mapreduce.Mapper;

/**
 * Mapper for TopAppsByUsage.
 * Input:  one CSV line  -> "Date,App,Usage (minutes),Notifications,Times Opened"
 * Output: (app, usageMinutes)
 */
public class TopAppsByUsageMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    private Text outKey = new Text();
    private IntWritable outValue = new IntWritable();

    @Override
    protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

        String line = value.toString().trim();
        if (line.isEmpty()) {
            return; // skip empty lines
        }

        // Skip header line (starts with "Date,App,...")
        if (line.startsWith("Date,")) {
            return;
        }

        String[] parts = line.split(",");

        // 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(
   usageStr);
46
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
```

```java
package edu.univ.haifa.bigdata;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
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 org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class UsageVsNotifications extends Configured
implements Tool {

    @Override
    public int run(String[] args) throws Exception {
        int reducers = Integer.parseInt(args[0]);
        Path inputPath = new Path(args[1]);
        Path outputPath = new Path(args[2]);

        Job job = Job.getInstance(getConf());
        job.setJobName("UsageVsNotifications");
        job.setJarByClass(UsageVsNotifications.class
);

        FileInputFormat.addInputPath(job, inputPath);
        FileOutputFormat.setOutputPath(job,
outputPath);

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

```

```
33          job.setMapperClass(UsageVsNotificationsMapper
    .class);
34          job.setMapOutputKeyClass(Text.class);
35          job.setMapOutputValueClass(Text.class);
36
37          job.setReducerClass(
    UsageVsNotificationsReducer.class);
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(),
48                  new UsageVsNotifications(), args);
49          System.exit(res);
50      }
51  }
52
```

```java
package edu.univ.haifa.bigdata;

import java.io.IOException;

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

/**
 * Reducer for TopAppsByUsage.
 * For each app, sums all usage minutes.
 * Input:  (app, [usage1, usage2, ...])
 * Output: (app, totalUsageMinutes)
 */
public class TopAppsByUsageReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    private IntWritable outValue = new IntWritable();

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {

        int sum = 0;
        for (IntWritable v : values) {
            sum += v.get();
        }

        outValue.set(sum);
        context.write(key, outValue);
    }
}

```

```java
package edu.univ.haifa.bigdata;

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

import java.io.IOException;
import java.time.DayOfWeek;
import java.time.LocalDate;

/**
 * Converts Date -> DayOfWeek and outputs (dayOfWeek, TimesOpened).
 */
public class LaunchesByWeekdayMapper
        extends Mapper<LongWritable, Text, Text, IntWritable> {

    private final Text outKey = new Text();
    private final IntWritable outValue = new IntWritable();

    @Override
    protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

        String line = value.toString().trim();
        if (line.isEmpty() || line.startsWith("Date,")) {
            return;
        }

        String[] parts = line.split(",");
        if (parts.length < 5) {
            return;
        }

        String dateStr = parts[0].trim();
```

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

```java
package edu.univ.haifa.bigdata;

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

import java.io.IOException;

/**
 * Sums launches per day of week.
 */
public class LaunchesByWeekdayReducer
        extends Reducer<Text, IntWritable, Text, IntWritable> {

    private final IntWritable outValue = new IntWritable();

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {

        int sum = 0;
        for (IntWritable v : values) {
            sum += v.get();
        }

        outValue.set(sum);
        context.write(key, outValue);
    }
}

```

```java
package edu.univ.haifa.bigdata;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
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 org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class AverageNotificationsPerApp extends
Configured implements Tool {

    @Override
    public int run(String[] args) throws Exception {
        int numberOfReducers = Integer.parseInt(args[
0]);
        Path inputPath = new Path(args[1]);
        Path outputPath = new Path(args[2]);

        Job job = Job.getInstance(getConf());
        job.setJobName("AverageNotificationsPerApp");
        job.setJarByClass(AverageNotificationsPerApp.
class);

        FileInputFormat.addInputPath(job, inputPath);
        FileOutputFormat.setOutputPath(job,
outputPath);

        job.setInputFormatClass(TextInputFormat.class
);
```

```
33          job.setOutputFormatClass(TextOutputFormat.
    class);
34
35          job.setMapperClass(
    AverageNotificationsPerAppMapper.class);
36          job.setMapOutputKeyClass(Text.class);
37          job.setMapOutputValueClass(IntWritable.class
    );
38
39          job.setReducerClass(
    AverageNotificationsPerAppReducer.class);
40          job.setOutputKeyClass(Text.class);
41          job.setOutputValueClass(FloatWritable.class);
42
43          job.setNumReduceTasks(numberOfReducers);
44
45          return job.waitForCompletion(true) ? 0 : 1;
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
```

```java
package edu.univ.haifa.bigdata;

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

import java.io.IOException;

/**
 * Output: key = app, value = "usage,notifications"
 */
public class UsageVsNotificationsMapper
        extends Mapper<LongWritable, Text, Text, Text> {

    private final Text outKey = new Text();
    private final Text outValue = new Text();

    @Override
    protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

        String line = value.toString().trim();
        if (line.isEmpty() || line.startsWith("Date,")) {
            return;
        }

        String[] parts = line.split(",");
        if (parts.length < 4) {
            return;
        }

        String app = parts[1].trim();
        String usageStr = parts[2].trim();
        String notifStr = parts[3].trim();

        outKey.set(app);
        outValue.set(usageStr + "," + notifStr);
```

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

```java
package edu.univ.haifa.bigdata;

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

import java.io.IOException;

/**
 * For each app, sums total usage and total notifications.
 * Output value format: "totalUsage\ttotalNotifications"
 */
public class UsageVsNotificationsReducer
        extends Reducer<Text, Text, Text, Text> {

    private final Text outValue = new Text();

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {

        int totalUsage = 0;
        int totalNotif = 0;

        for (Text v : values) {
            String[] parts = v.toString().split(",");
            if (parts.length != 2) continue;

            try {
                int usage = Integer.parseInt(parts[0].trim());
                int notif = Integer.parseInt(parts[1].trim());
                totalUsage += usage;
                totalNotif += notif;
            } catch (NumberFormatException ignored) {
            }
        }
```

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

```java
package edu.univ.haifa.bigdata;

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

import java.io.IOException;

/**
 * Input CSV: Date,App,Usage (minutes),Notifications,Times Opened
 * Output: (app, notificationsForThatDay)
 */
public class AverageNotificationsPerAppMapper
        extends Mapper<LongWritable, Text, Text, IntWritable> {

    private final Text outKey = new Text();
    private final IntWritable outValue = new IntWritable();

    @Override
    protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

        String line = value.toString().trim();
        if (line.isEmpty() || line.startsWith("Date,")) {
            return; // skip empty or header
        }

        String[] parts = line.split(",");
        if (parts.length < 4) {
            return;
        }

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

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

```java
package edu.univ.haifa.bigdata;

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

import java.io.IOException;

/**
 * For each app, computes average notifications per day.
 * Input:  (app, [n1, n2, ...])
 * Output: (app, avgNotificationsPerDay)
 */
public class AverageNotificationsPerAppReducer
        extends Reducer<Text, IntWritable, Text, FloatWritable> {

    private final FloatWritable outValue = new FloatWritable();

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {

        int sum = 0;
        int count = 0;

        for (IntWritable v : values) {
            sum += v.get();
            count++;
        }

        if (count == 0) {
            return;
        }

        float avg = (float) sum / count;
```

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