1번) /shared-data/reviews_Books_5.json 의 전체 상품 평균 "overall" 점수 분석

```
<modelVersion>4.0.0</modelVersion>
<groupId>kr.kookmin.hadoop</groupId>
<artifactId>MapReduce</artifactId>
<version>0.0.1-SNAPSHOT</version>
  <dependencies>
      <dependency>
          <groupId>org.apache.hadoop</groupId>
          <artifactId>hadoop-core</artifactId>
          <version>1.1.2</version>
       </dependency>
       <dependency>
          <groupId>com.googlecode.json-simple</groupId>
          <artifactId>json-simple</artifactId>
          <version>1.1</version>
      </dependency>
  </dependencies>
```

위의 그림은 Maven을 이용하여 hadoop을 이용하기 위한 hadoop-core를 추가하였고 json파일을 파싱하기 위해 json-simple 라이브러리를 추가하였습니다.

```
package com.kookmin.one;

import org.apache.hadoop.conf.Configuration;

/**
 * JobConf를 통해서 Hadoop의 MapReduce를 이용할 것이며, JobConf가 알아서 map function과 reducer funtion call
 * JobConf를 통해서 Mapper와 Reducer의 input 그리고 output Type을 설정
 */
public class Overall {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        if (args.length != 2) {
            System.out.println("Usage: average <input> <output>");
            System.exit(2);
        }
        Job job = new Job(conf, "Overall");
        job.setJarByClass(Overall.class);
        job.setMapperClass(OverallMapper.class);
        job.setReducerClass(OverallReducer.class);

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

        job.setOutputKeyClass(IntWritable.class); // input으로 받아서 mapper에서 output type
        job.setOutputValueClass(DoubleWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0])); //input 경로
        FileOutputFormat.setOutputPath(job, new Path(args[1])); //ouput 경로
        job.waitForCompletion(true); // 실행
    }
}
```

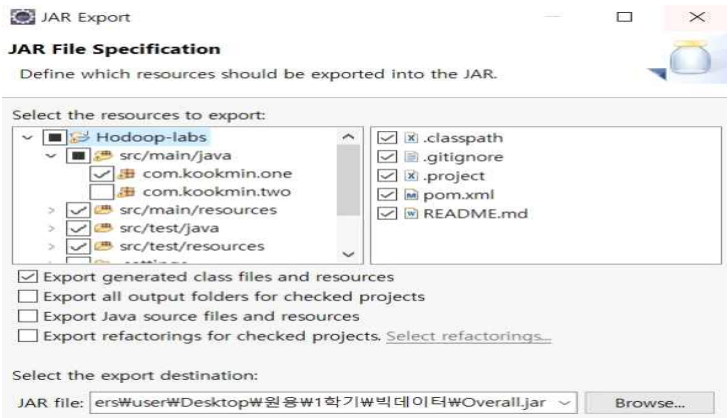
위의 사진은 전체 상품 평균을 구하기 위한 Driver 클래스입니다. job.setOutputKeyClass(IntWritable.class) 이부분은 input 으로 받아온 값을 mapper에서 처리한 후 어떠한 타입으로 reducer에게 전달할지 정하는 부분이며, 전체 키 값은 1인 int 형으로 전달하고 overall은 double타입으로 전달해서 reducer에서 처리할 것입니다.

```
package com.kookmin.one;

import java.io.IOException;

/**
 *  맵을 만들기 위해서 매퍼를 상속
 *  매퍼 클래스는 4가 인자를 받음 => 하둡 전용 자료형 ( String 은 text로 사용함)
 *  리듀스로 key, value 쌍을 보내는 작업을 context.write
 */
public class OverallMapper extends Mapper<LongWritable, Text, IntWritable, DoubleWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, IntWritable, DoubleWritable>.Context context
            throws IOException, InterruptedException {
        double overall;
        String line = value.toString();
        String[] tuple = line.split("\\n"); // 라인별로
        try {
            for(int i=0; i<tuple.length; i++) {
                JSONObject obj = new JSONObject(tuple[i]); // json으로 파싱
                overall = obj.getDouble("overall"); //key값 overall을 이용해서 value 값을 찾는다.
                context.write(one , new DoubleWritable(overall));
            }
        }catch(JSONException e) {
            e.printStackTrace();
        }
```

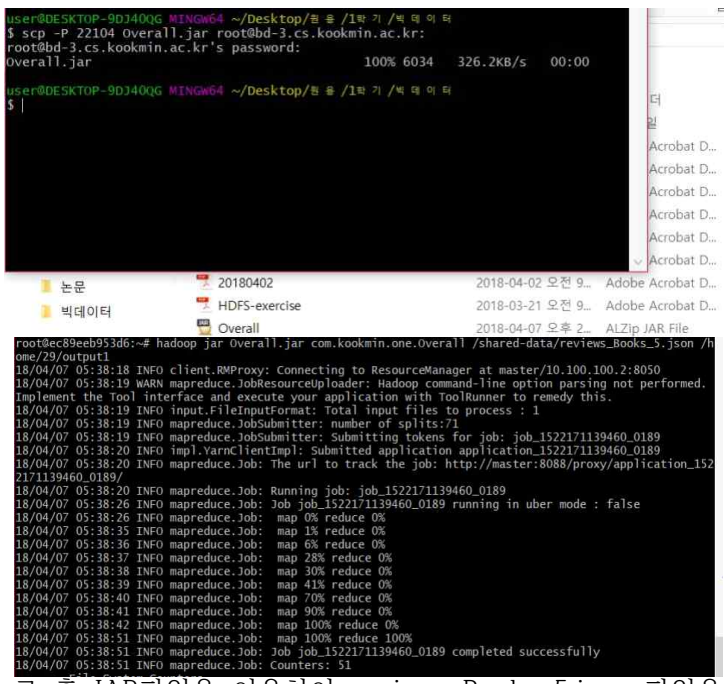
위의 사진은 Mapper 클래스를 나타낸 그림이며, 라인 별로 Text를 받아오며 Json 으로 파싱을 한 뒤 분석을 시작합니다. overall 이라는 key를 이용하여 value 값을 찾아서 (1, overall) 형태로 reducer에게 전달합니다.

```
/**
 * 맵의 output 인자와 리듀스의 input 인자는 같아야 한다.
 *
 */
public class OverallReducer extends Reducer<IntWritable, DoubleWritable, NullWritable, DoubleWritable>{
    private DoubleWritable result = new DoubleWritable();

    @Override
    protected void reduce(IntWritable key, Iterable<DoubleWritable> values,
            Reducer<IntWritable, DoubleWritable, NullWritable, DoubleWritable>.Context context) throws IOException, Interru
        double sum = 0;
        double count=0;
        double average=0;
        for (DoubleWritable val : values) { //키값은 1로 같고 전체 평균을 구해 나간다.
            sum += val.get(); //전체 합 구하기
            count++; // 평균을 구하기 위한 count
        }
        average = sum / count; // 평균구하기
        result.set(average); // 전체 평균 set

        context.write(NullWritable.get(), result);
    }
}
```

위의 사진은 reducer의 클래스입니다. 키 값은 1로 같기 때문에 모든 overall 값을 더하면서 count를 올려줍니다. 모든 작업이 끝난 후 평균을 구하고 결과를 write 합니다.



JAR 파일을 구하기 위해 Export를 직접 했으며 그 후 JAR 파일을 서버로 보내 주는 과정으로 진행 했습니다.



그 후 JAR파일을 이용하여 reviews_Books_5.json 파일을 분석을 진행했습니다. 위의 그림은 분석 완료 된 그림입니다.

```
root@ec89eeb953d6:~# hdfs dfs -cat /home/29/output1/part-r-00000
4.2499322041784255
root@ec89eeb953d6:~#
```

위의 그림은 마지막 결과 사진입니다.

2. /shared-data/reviews_Books_5.json 에서 가장 많은 리뷰를 남긴 사용자 아이디("reviewerID") 및 리뷰 횟수는?

=> setup(), cleanup() 함수를 override해서 진행 하였습니다. 분석과정은 아래와 같습니다.

1) setup 메소드는 사전 초기화 등이 필요한 작업을 수행
2) 만들어진 리뷰스 레코드들을 하나씩 reduce 메소드의 입력으로 넘긴다.
3) 마지막으로 cleanup에서 최종 정리나 수행

```
public class MostReviews {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        if (args.length != 2) {
            System.out.println("Usage: MostReviews <input> <output>");
            System.exit(2);
        }
        Job job = new Job(conf, "MostReviews");
        job.setJarByClass(MostReviews.class);
        job.setMapperClass(MostReviewsMapper.class);
        job.setReducerClass(MostReviewsReducer.class);

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

        job.setOutputKeyClass(Text.class); //맵에서 받아올 key value type 정해주기
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.waitForCompletion(true); // 실행
    }
}
```

Driver 클래스에서는 job.setOutputKeyClass(Text.class) 부분을 수정하였습니다. 맵에서 input 값을 받아와서 reducer로 넘겨줄 때 (reviewerId, 1) 형식으로 넘겨주기 위하여 타입을 지정해 줍니다.

```
public class MostReviewsMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context
            throws IOException, InterruptedException {
        String id;
        String line = value.toString();
        String[] tuple = line.split("\\n"); // 라인별로 가져옴
        try {
            for(int i=0; i<tuple.length; i++) {
                JSONObject obj = new JSONObject(tuple[i]); // json으로 파싱
                id = obj.getString("reviewerID");
                word.set(id);
                context.write(word , one); // (revierId, 1)를 리턴
            }
        }catch(JSONException e) {
            e.printStackTrace();
        }
    }
}
```

Mapper에서 마찬가지로 json으로 파싱을 하고 리뷰ID값을 받아온뒤 context.write(id, 1) 으로 작성합니다.

```
public class MostReviewsReducer extends Reducer<Text, IntWritable, Text, IntWritable>{
    //private IntWritable result = new IntWritable();

    private int maxCount;
    private String reviewer;
    @Override
    protected void setup(Context context) throws IOException, InterruptedException{
        maxCount = context.getConfiguration().getInt("maxCount", 0);
        reviewer = context.getConfiguration().get("reviewer"); // global 변수 사용하기 위해
    }

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
            Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, Int
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();  // reviewer 별로 갯수 count
        }
        if(sum > maxCount) { // 갯수가 더 많은 reviewer 가 있다면
            maxCount = sum;
            reviewer = key.toString();
        }
    }
    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException {
        context.write(new Text(reviewer),new IntWritable(maxCount)); //Reducer가 끝난 후
    }
```

위의 그림은 reducer 이며 setup()에서 가장 많은 리뷰를 찾기 위한 global 변수 maxCount와 reviewer로 정했습니다. reduce함수에서는 같은 id 의 리뷰 개수를 count를 먼저하고 나서 maxCount 와 비교해서 더 큰 값이 있다면 chage해줍니다. 모든 작업이 끝나구 나면 cleanup함수에서 write해줍니다.

```
root@ec89eeb953d6:~# hadoop jar MostReviews.jar com.kookmin.two.MostReviews /shared-data/reviews_Books_5.json /home/29/output2_5
18/04/07 07:01:36 INFO client.RMProxy: Connecting to ResourceManager at master/10.100.100.2:8050
18/04/07 07:01:36 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool inter
face and execute your application with ToolRunner to remedy this.
18/04/07 07:01:37 INFO input.FileInputFormat: Total input files to process : 1
18/04/07 07:01:37 INFO mapreduce.JobSubmitter: number of splits:71
18/04/07 07:01:37 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1522171139460_0237
18/04/07 07:01:37 INFO impl.YarnClientImpl: Submitted application application_1522171139460_0237
18/04/07 07:01:37 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1522171139460_0237/
18/04/07 07:01:37 INFO mapreduce.Job: Running job: job_1522171139460_0237
18/04/07 07:01:43 INFO mapreduce.Job: Job job_1522171139460_0237 running in uber mode : false
18/04/07 07:01:43 INFO mapreduce.Job:  map 0% reduce 0%
18/04/07 07:01:55 INFO mapreduce.Job:  map 4% reduce 0%
18/04/07 07:01:56 INFO mapreduce.Job:  map 35% reduce 0%
18/04/07 07:01:57 INFO mapreduce.Job:  map 52% reduce 0%
18/04/07 07:01:58 INFO mapreduce.Job:  map 61% reduce 0%
18/04/07 07:01:59 INFO mapreduce.Job:  map 76% reduce 0%
18/04/07 07:02:00 INFO mapreduce.Job:  map 86% reduce 0%
18/04/07 07:02:02 INFO mapreduce.Job:  map 92% reduce 0%
18/04/07 07:02:03 INFO mapreduce.Job:  map 95% reduce 0%
18/04/07 07:02:04 INFO mapreduce.Job:  map 98% reduce 0%
18/04/07 07:02:05 INFO mapreduce.Job:  map 99% reduce 0%
18/04/07 07:02:06 INFO mapreduce.Job:  map 100% reduce 0%
18/04/07 07:02:13 INFO mapreduce.Job:  map 100% reduce 86%
18/04/07 07:02:14 INFO mapreduce.Job:  map 100% reduce 100%
18/04/07 07:02:15 INFO mapreduce.Job: Job job_1522171139460_0237 completed successfully
```

```
root@ec89eeb953d6:~# hdfs dfs -cat /home/29/output2_5/part-r-00000
AFVQZQ8PW0L      23222
```

3. /shared-data/reviews_Books_5.json 에서 helpful 필드는 [a,b] 형식을 가지며, b명의 사용자가 해당 리뷰가 도움이 되는지 투표했다는 의미이며, 이중 a 명의 사용자가 도움이 된다는 의견을 남겼다는 의미이다. b값이 10보다 큰 사용자 중에서 도움이 된다고 하는 사람의 비율(a/b)이 가장 높은 아이템 (asin) 의 아이디는?

```
public class Ratio {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        if (args.length != 2) {
            System.out.println("Usage: Ratio <input> <output>");
            System.exit(2);
        }
        Job job = new Job(conf, "Ratio");
        job.setJarByClass(Ratio.class);
        job.setMapperClass(RatioMapper.class);
        job.setReducerClass(RatioReducer.class);

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

        job.setOutputKeyClass(Text.class); //맵에서 받아올 key value type 정해주기
        job.setOutputValueClass(FloatWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.waitForCompletion(true); // 실행
    }
}
```

같은 방식으로 Text.class 와 FloatWritable 타입으로 변경하였다.

```
public class RatioMapper extends Mapper<LongWritable, Text, Text, FloatWritable> {
    private Text word = new Text();

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, FloatWritable>
            throws IOException, InterruptedException {
        String asin, tempHelpful;
        int a=0, b=0;
        float resultRatio= 0;
        String line = value.toString();
        String[] tuple = line.split("\\n"); // 라인별로 가져올
        try {
            for(int i=0; i<tuple.length; i++) {
                JSONObject obj = new JSONObject(tuple[i]); // json으로 파싱
                tempHelpful = obj.getString("helpful");  //helpful value 가져오기 [a, b]
                a= tempHelpful.charAt(1) - '0'; //  char -> int
                b = tempHelpful.charAt(4) - '0';
                if(b > 10) // b가 10보다 크다면
                {
                    resultRatio = (float)a/(float)b; // 비율 구해서 context에 write
                    asin = obj.getString("asin"); // asin value 값
                    word.set(asin);
                    context.write(word , new FloatWritable(resultRatio));
                }
            }
        }catch(JSONException e) {
            e.printStackTrace();
        }

    }
}
```

Mapper에서는 helpful을 받아와서 [a, b] a와 b 값을 각각 charAt()으로 받아옵니다. 그 후 char 타입이기 때문에 int형으로 변환한 뒤 b 값이 10보다 크면 비율과 asin 값을 구해서 reducer에게 넘겨 줍니다.

```
public class RatioReducer extends Reducer<Text, FloatWritable, Text, FloatWritable>{
    private FloatWritable result = new FloatWritable(0);

    private float maxCount;
    private String asin;
    @Override
    protected void setup(Context context) throws IOException, InterruptedException{
        maxCount = context.getConfiguration().getFloat("maxCount",0);
        asin = context.getConfiguration().get("asin"); // global 변수 사용
    }
    @Override
    protected void reduce(Text key, Iterable<FloatWritable> values,
            Reducer<Text, FloatWritable, Text, FloatWritable>.Context context) throws IOException, Interru
        float temp = 0;
        for (FloatWritable val : values) {
            temp = val.get();
            if(temp > maxCount) { // 비율중에 maxCount 보다 큰 값이 있다면
                maxCount = temp;
                asin = key.toString();
            }
        }
    }
    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException {
        context.write(new Text(asin),new FloatWritable(maxCount)); // 최종 write
    }

}
```

마지막 Reducer에서는 (asin, 비율) 값을 받아와서 가장 큰 비율을 구하고 cleanup에서 최종적으로 가장 큰 비율을 write해줍니다.

```
root@ec89eeb953d6:~# hadoop jar Ratio.jar com.kookmin.three.Ratio /shared-data/reviews_Books_5.json /home/29/output3_10
18/04/08 07:48:20 INFO client.RMProxy: Connecting to ResourceManager at master/10.100.100.2:8050
18/04/08 07:48:21 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool inte
cute your application with ToolRunner to remedy this.
18/04/08 07:48:21 INFO input.FileInputFormat: Total input files to process : 1
18/04/08 07:48:21 INFO mapreduce.JobSubmitter: number of splits:71
18/04/08 07:48:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1522171139460_0373
18/04/08 07:48:22 INFO impl.YarnClientImpl: Submitted application application_1522171139460_0373
18/04/08 07:48:22 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1522171139460_0373/
18/04/08 07:48:22 INFO mapreduce.Job: Running job: job_1522171139460_0373
18/04/08 07:48:28 INFO mapreduce.Job: Job job_1522171139460_0373 running in uber mode : false
18/04/08 07:48:28 INFO mapreduce.Job:  map 0% reduce 0%
18/04/08 07:48:36 INFO mapreduce.Job:  map 1% reduce 0%
18/04/08 07:48:37 INFO mapreduce.Job:  map 3% reduce 0%
18/04/08 07:48:38 INFO mapreduce.Job:  map 17% reduce 0%
18/04/08 07:48:39 INFO mapreduce.Job:  map 21% reduce 0%
18/04/08 07:48:40 INFO mapreduce.Job:  map 27% reduce 0%
18/04/08 07:48:41 INFO mapreduce.Job:  map 30% reduce 0%
18/04/08 07:48:42 INFO mapreduce.Job:  map 31% reduce 0%
18/04/08 07:48:43 INFO mapreduce.Job:  map 34% reduce 0%
18/04/08 07:48:45 INFO mapreduce.Job:  map 38% reduce 0%
18/04/08 07:48:48 INFO mapreduce.Job:  map 42% reduce 0%
18/04/08 07:48:49 INFO mapreduce.Job:  map 56% reduce 0%
18/04/08 07:48:50 INFO mapreduce.Job:  map 57% reduce 0%
18/04/08 07:48:51 INFO mapreduce.Job:  map 62% reduce 0%
18/04/08 07:48:52 INFO mapreduce.Job:  map 81% reduce 0%
18/04/08 07:48:53 INFO mapreduce.Job:  map 93% reduce 0%
18/04/08 07:48:54 INFO mapreduce.Job:  map 97% reduce 0%
18/04/08 07:48:55 INFO mapreduce.Job:  map 100% reduce 30%
18/04/08 07:49:01 INFO mapreduce.Job:  map 100% reduce 85%
18/04/08 07:49:02 INFO mapreduce.Job:  map 100% reduce 100%
18/04/08 07:49:02 INFO mapreduce.Job: Job job_1522171139460_0373 completed successfully
18/04/08 07:49:02 INFO mapreduce.Job: Counters: 51
```

```
root@ec89eeb953d6:~# hdfs dfs -cat /home/29/output3_10/part-r-00000
0001055178      0.2
```

위의 그림은 Jar파일을 가지고 분석한 결과입니다.

4) /shared-data/reviews_Books_5.json 에서 각 reviewer 별로 helpful 필드 값을 모았을때 ([a,b] 를 a 값과 b 값으로 각각 더함), 가장 높은 a 값을 가지는 사용자의 (reviewerID) 아아디는?

```
public class MostBiggest {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        if (args.length != 2) {
            System.out.println("Usage: MostBiggest <input> <output>");
            System.exit(2);
        }
        Job job = new Job(conf, "MostBiggest");
        job.setJarByClass(MostBiggest.class);
        job.setMapperClass(MostBiggestMapper.class);
        job.setReducerClass(MostBiggestReducer.class);

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

        job.setOutputKeyClass(Text.class); // input으로 받아서 mapper에서 output type
        job.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job, new Path(args[0])); //input 경로
        FileOutputFormat.setOutputPath(job, new Path(args[1])); //ouput 경로
        job.waitForCompletion(true); // 실행
    }
}
```

위의 Driver 클래스에서 Text.class / Text.class 는 reviewerID 와 helpfule value값을 각각 reducer로 전달하기 위하여 설정하였다.

```
public class MostBiggestMapper extends Mapper<LongWritable, Text, Text, Text> {

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, Text>.Context co
            throws IOException, InterruptedException {
        String reviewerID, helpful;
        String line = value.toString();
        String[] tuple = line.split("\\n"); // 라인별로
        try {
            for(int i=0; i<tuple.length; i++) {
                JSONObject obj = new JSONObject(tuple[i]); // json으로 파싱
                helpful = obj.getString("helpful"); // helpful value
                reviewerID = obj.getString("reviewerID"); //id값
                context.write(new Text(reviewerID), new Text(helpful));
            }
        }catch(JSONException e) {
            e.printStackTrace();
        }
    }
}
```

위의 Mapper 클래스에서는 reviewerID와 helpful을 각각 json으로 파싱후 context에 write해준 내용이다.

```
public class MostBiggestReducer extends Reducer<Text, Text, Text, IntWritable>{
    private DoubleWritable result = new DoubleWritable();

    private int valueA;
    private String reviewerID;
    @Override
    protected void setup(Context context) throws IOException, InterruptedException{
        valueA = context.getConfiguration().getInt("valueA", 0);
        reviewerID = context.getConfiguration().get("reviewerID");
    }
    @Override
    protected void reduce(Text key, Iterable<Text> values,
            Reducer<Text, Text, Text, IntWritable>.Context context) throws IOException, Inter
        int a=0, b=0;
        String tempHelpful;
        for (Text val : values) {
            tempHelpful = val.toString();
            a= tempHelpful.charAt(1) - '0'; //  char -> int
            b = tempHelpful.charAt(4) - '0';
            if(a > valueA) { // 가장 큰 a 찾기
                valueA = a;
                reviewerID = key.toString();
            }
        }
    }
    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException {
        context.write(new Text(reviewerID),new IntWritable(valueA));
    }
}
```

Reducer 클래스에서는 helpful value를 가져와서 a,b 각각 값을 char에서 int로 변환 해주어서 가장 큰 a값을 찾는다. 같은 방식으로 setup()에서는 global 변수를 이용하고 cleanup() 함수에서는 reducer를 끝내구 최종 값을 써준다.

```
root@ec89eeb953d6:~# hadoop jar MostBiggest.jar com.kookmin.four.MostBiggest /shared-data/reviews_Books_5.json /home/29/output4_10
18/04/08 08:33:36 INFO client.RMProxy: Connecting to ResourceManager at master/10.100.100.2:8050
18/04/08 08:33:37 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and exe
cute your application with ToolRunner to remedy this.
18/04/08 08:33:37 INFO input.FileInputFormat: Total input files to process : 1
18/04/08 08:33:37 INFO mapreduce.JobSubmitter: number of splits:71
18/04/08 08:33:37 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1522171139460_0374
18/04/08 08:33:37 INFO impl.YarnClientImpl: Submitted application application_1522171139460_0374
18/04/08 08:33:37 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1522171139460_0374/
18/04/08 08:33:37 INFO mapreduce.Job: Running job: job_1522171139460_0374
18/04/08 08:33:44 INFO mapreduce.Job: Job job_1522171139460_0374 running in uber mode : false
18/04/08 08:33:44 INFO mapreduce.Job:  map 0% reduce 0%
18/04/08 08:33:56 INFO mapreduce.Job:  map 1% reduce 0%
18/04/08 08:33:57 INFO mapreduce.Job:  map 14% reduce 0%
18/04/08 08:33:58 INFO mapreduce.Job:  map 38% reduce 0%
18/04/08 08:33:59 INFO mapreduce.Job:  map 52% reduce 0%
18/04/08 08:34:00 INFO mapreduce.Job:  map 54% reduce 0%
18/04/08 08:34:01 INFO mapreduce.Job:  map 58% reduce 0%
18/04/08 08:34:02 INFO mapreduce.Job:  map 69% reduce 0%
18/04/08 08:34:03 INFO mapreduce.Job:  map 87% reduce 0%
18/04/08 08:34:04 INFO mapreduce.Job:  map 100% reduce 0%
18/04/08 08:34:13 INFO mapreduce.Job:  map 100% reduce 98%
18/04/08 08:34:14 INFO mapreduce.Job:  map 100% reduce 100%
18/04/08 08:34:14 INFO mapreduce.Job: Job job_1522171139460_0374 completed successfully
```

```
root@ec89eeb953d6:~# hdfs dfs -cat /home/29/output4_10/part-r-00000
A01911642JTFI4ONWRUNF   9
```

위의 그림은 결과 화면입니다.

5) /shared-data/reviews_Books_5.json에서 각 reviewerText에서 가장 긴 리뷰를 남긴 reviewrName 값은?

```
public class MostLength {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        if (args.length != 2) {
            System.out.println("Usage: MostLength <input> <output>");
            System.exit(2);
        }
        Job job = new Job(conf, "MostLength");
        job.setJarByClass(MostLength.class);
        job.setMapperClass(MostLengthMapper.class);
        job.setReducerClass(MostLengthReducer.class);

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

        job.setOutputKeyClass(Text.class); // input으로 받아서 mapper에서 output type
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0])); //input 경로
        FileOutputFormat.setOutputPath(job, new Path(args[1])); //ouput 경로
        job.waitForCompletion(true); // 실행
    }
}
```

각각 reviewerName 별로 reviewerText 길이를 넘겨주기 위해서 job.setOutputValueClass(IntWritable.class) 부분을 수정 하였습니다.

```
public class MostLengthMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context
            throws IOException, InterruptedException {
        String reviewerName, reviewText;
        int len;
        String line = value.toString();
        String[] tuple = line.split("\\n"); // 라인별로
        try {
            for(int i=0; i<tuple.length; i++) {
                JSONObject obj = new JSONObject(tuple[i]); // json으로 파싱
                reviewText = obj.getString("reviewText");
                len = reviewText.length(); // 리뷰텍스트 길이 구하기
                reviewerName = obj.getString("reviewerName"); //key값 overall을 이용해서 value 값을 찾는다.
                context.write(new Text(reviewerName), new IntWritable(len));
            }
        }catch(JSONException e) {
            e.printStackTrace();
        }
    }
}
```

위의 Mapper에서 똑같은 방식으로 라인별로 json으로 파싱을 하고 reviewerText의 전체 스트링 길이를 구해서 (reviewerName, length) 형식으로 reducer에게 전달한다.

```
public class MostLengthReducer extends Reducer<Text, IntWritable, Text, IntWritable>{
    private DoubleWritable result = new DoubleWritable();

    private int maxLen;
    private String reviewerName;
    @Override
    protected void setup(Context context) throws IOException, InterruptedException{
        maxLen = context.getConfiguration().getInt("maxLen", 0); //global 변수
        reviewerName = context.getConfiguration().get("reviewerName"); //global 변수
    }
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
            Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedE
        int cnt = 0;
        for (IntWritable val : values) {
            cnt = val.get();
            if(cnt > maxLen) { //가장긴 텍스트 찾기
                maxLen = cnt;
                reviewerName = key.toString();
            }
        }

    }
    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException {
         context.write(new Text(reviewerName),new IntWritable(maxLen)); //결과 write
    }
}
```

위의 마지막 reducer에서는 setup(), cleanup() 함수를 각각 사용해서 가장 긴 텍스트 길이를 구해서 name 과 context에 write해준다.

```
root@ec89eeb953d6:~# hadoop jar MostLength.jar com.kookmin.five.MostLength /shared-data/reviews_Books_5.json /home/29/output5_10
18/04/08 08:43:26 INFO client.RMProxy: Connecting to ResourceManager at master/10.100.100.2:8050
18/04/08 08:43:26 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and
tute your application with ToolRunner to remedy this.
18/04/08 08:43:27 INFO input.FileInputFormat: Total input files to process : 1
18/04/08 08:43:27 INFO mapreduce.JobSubmitter: number of splits:71
18/04/08 08:43:27 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1522171139460_0375
18/04/08 08:43:27 INFO impl.YarnClientImpl: Submitted application application_1522171139460_0375
18/04/08 08:43:27 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1522171139460_0375/
18/04/08 08:43:27 INFO mapreduce.Job: Running job: job_1522171139460_0375
18/04/08 08:43:33 INFO mapreduce.Job: Job job_1522171139460_0375 running in uber mode : false
18/04/08 08:43:33 INFO mapreduce.Job:  map 0% reduce 0%
18/04/08 08:43:42 INFO mapreduce.Job:  map 1% reduce 0%
18/04/08 08:43:44 INFO mapreduce.Job:  map 10% reduce 0%
18/04/08 08:43:45 INFO mapreduce.Job:  map 11% reduce 0%
18/04/08 08:43:47 INFO mapreduce.Job:  map 13% reduce 0%
18/04/08 08:43:48 INFO mapreduce.Job:  map 14% reduce 0%
18/04/08 08:43:50 INFO mapreduce.Job:  map 20% reduce 0%
18/04/08 08:43:51 INFO mapreduce.Job:  map 31% reduce 0%
18/04/08 08:43:52 INFO mapreduce.Job:  map 59% reduce 0%
18/04/08 08:43:53 INFO mapreduce.Job:  map 62% reduce 0%
18/04/08 08:43:55 INFO mapreduce.Job:  map 70% reduce 0%
18/04/08 08:43:56 INFO mapreduce.Job:  map 92% reduce 0%
18/04/08 08:43:57 INFO mapreduce.Job:  map 100% reduce 0%
18/04/08 08:44:00 INFO mapreduce.Job:  map 100% reduce 54%
18/04/08 08:44:05 INFO mapreduce.Job:  map 100% reduce 100%
18/04/08 08:44:06 INFO mapreduce.Job: Job job_1522171139460_0375 completed successfully
```

```
root@ec89eeb953d6:~# hdfs dfs -cat /home/29/output5_10/part-r-00000
Christian Reader        32658
```

위의 사진은 결과 화면입니다.