Give a brief answers to the questions below;

1. What is the purpose of RecordReader in Hadoop?

RecordReader reads <key, value> pairs from an [InputSplit](https://hadoop.apache.org/docs/r2.6.2/api/org/apache/hadoop/mapred/InputSplit.html" \o "interface in org.apache.hadoop.mapred).

RecordReader, typically, converts the byte-oriented view of the input, provided by the InputSplit, and presents a record-oriented view for the [Mapper](https://hadoop.apache.org/docs/r2.6.2/api/org/apache/hadoop/mapred/Mapper.html) & [Reducer](https://hadoop.apache.org/docs/r2.6.2/api/org/apache/hadoop/mapred/Reducer.html) tasks for processing. It thus assumes the responsibility of processing record boundaries and presenting the tasks with keys and values.

1. What happens if the number of reducers is 0?

 Setting the **number of reducers** to **zero** is a valid configuration in Hadoop.

When the **reducers is set** to **zero** no **reducers** will be executed, and the output of each mapper will be stored to a separate file on HDFS.

**But the framework does not sort the Map outputs before writing them out to the HDFS output directory.**

3. What is meant by Map-side and Reduce-side join in Hadoop?

Map side join performs join before data reached to Map. Map function expects a strong prerequisites before joining data at map side. Map side join is efficient compare to reduce side but it require strict format.

Prerequisites:

* Data should be partitioned and sorted in particular way.
* Each input data should be divided in same number of partition.
* Must be sorted with same key.
* All the records for a particular key must reside in the same partition.

Reduce side join also called as Repartitioned join or Repartitioned sort merge join and also it is mostly used join type. It will have to go through sort and shuffle phase which would incur network overhead. Reduce side join uses few terms like data source, tag and group key lets be familiar with it.

* Data Source is referring to data source files, probably taken from RDBMS
* Tag would be used to tag every record with it’s source name, so that it’s source can be identified at any given point of time be it is in map/reduce phase. why it is required will cover it later.
* Group key is referring column to be used as join key between two data sources.

4.What is the significance of conf.setMapper class?

conf.setMapper class sets the Mapper class and all the stuff related to map job such as reading a data and generating a key-value pair out of the mapper.

5. Give an example scenario on the usage of counters.

It is very handy to define your own counters to track any kind of statistics about the records you are manipulating in the mapper and the reducer. The most natural use of that is to use counters to track the number of malformed records.

If you are executing a job  and you see an abnormally high number of malformed records, it can give a good hint that you perhaps have a bug in your code or some problem with your data (note this is actually a much simpler way to spot issues than tracking error messages in a distributed set of log files). But you can actually use counters for any kind of other statistics on your records.

We will create a counter group called WordsNature that will count how many unique tokens there is in all, how many unique tokens starts with a digit and how many unique tokens starts with a letter.

So our enum declaration will look like that:

static enum WordsNature { STARTS\_WITH\_DIGIT, STARTS\_WITH\_LETTER, ALL }

We will also need a very basic StringUtils class:

package com.philippeadjiman.hadooptraining;

public class StringUtils {

public static boolean startsWithDigit(String s){

if( s == null || s.length() == 0 )

return false;

return Character.isDigit(s.charAt(0));

}

public static boolean startsWithLetter(String s){

if( s == null || s.length() == 0 )

return false;

return Character.isLetter(s.charAt(0));

}

}

Since we are interested in unique tokens, we will put the code related with the counter into the reduce method. So here how the reduce method will look like:

public void reduce(Text key, Iterable values, Context context)

throws IOException, InterruptedException {

int sum = 0;

String token = key.toString();

if( StringUtils.startsWithDigit(token) ){

context.getCounter(WordsNature.STARTS\_WITH\_DIGIT).increment(1);

}

else if( StringUtils.startsWithLetter(token) ){

context.getCounter(WordsNature.STARTS\_WITH\_LETTER).increment(1);

}

context.getCounter(WordsNature.ALL).increment(1);

for (IntWritable value : values) {

sum += value.get();

}

context.write(key, new IntWritable(sum));

}

6. Elaborate some problems which can only be solved by MapReduce and cannot be solved by PIG?

* Really nasty data formats or completely unstructured data(video, audio, raw human-readable text).
* Pig is definitely slow compared to Map Reduce jobs.
* When you would like more power to optimize your code.

7. In what kind of scenarios, MR jobs will be more useful than PIG?

Let us take a scenario where we want to count the population in two cities. I have a data set and sensor list of different cities. I want  to count the population by using one mapreduce for two cities. Let us assume that one is Bangalore and the other is Noida. So I need to consider key of  Bangalore city  similar to Noida through which I can bring the population data of these two cities to one reducer. The idea behind this is some how I have to instruct map reducer program – whenever you find city with the name ‘Bangalore‘ and city with the name ‘Noida’,  you create the alias name which will be the common name for these two cities so that  you create a common key for both the cities and it get passed to the same reducer. For this, we have to write  custom partitioner.  
  
In mapreduce when you create a ‘key’ for city,  you have to consider ’city’ as the key. So, whenever the framework comes across a different city, it considers it as a different key. Hence, we need to use customized partitioner. There is a provision in mapreduce only, where you can write your custom partitioner and mention if city = bangalore or noida then pass similar hashcode.  However, we cannot create custom partitioner in Pig. As Pig is not a framework, we cannot direct execution engine to customize the partitioner. In such scenarios, MapReduce works better than Pig.

8. What are combiners and when are these used in a MapReduce job?

Combiner is a semi-reducer in mapreduce. This is an optional class which can be specified in mapreduce driver class to process the output of map tasks before submitting it to reducer tasks

To increase the efficiency of MapReduce Program, Combiners are used. The amount of data can be reduced with the help of combiner’s that need to be transferred across to the reducers. If the operation performed is commutative and associative you can use your reducer code as a combiner. The execution of combiner is not guaranteed in Hadoop