In this exercise, we will define our custom keys and values and use them in our map reduce program. Following Program runs on 250 mb file and imploys counters.

Defining value class “name”

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;

public class name implements Writable {
	Text first = null;
	Text middle = null;
	Text last = null;

	public name(){
		this.first = new Text();
		this.middle = new Text();
		this.last = new Text();
	}
	public name(Text f,Text m,Text l){
		this.first = f;
		this.middle = m;
		this.last = l;
	}
	public name(String f,String m,String l){
		this.first = new Text(f);
		this.middle = new Text(m);
		this.last = new Text(l);
	}
	@Override
	public void readFields(DataInput in) throws IOException {
		// TODO Auto-generated method stub
		first.readFields(in);
		middle.readFields(in);
		last.readFields(in);
	}
	@Override
	public void write(DataOutput out) throws IOException {
		// TODO Auto-generated method stub
		first.write(out);
		middle.write(out);
		last.write(out);
	}
	public Text getName(){
		return new Text(first.toString()+" "+middle.toString()+" "+last.toString());
	}
}
```

Your value class should import Writable interface

Overloading different constructors

Our value class must override two methods:

1) readFields()
2) write()

Defining key class “patent”

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;

public class patent implements WritableComparable<patent>{
	LongWritable patentNo = null;
	Text country = null;
	public patent(){
		this.patentNo = new LongWritable();
		this.country = new Text();
	}
	public patent(LongWritable l, Text t){
		this.patentNo = l;
		this.country = t;
	}
	public patent(long n,String c){
		this.patentNo = new LongWritable(n);
		this.country = new Text(c);
	}
	@Override
	public void readFields(DataInput in) throws IOException {
		// TODO Auto-generated method stub
		patentNo.readFields(in);
		country.readFields(in);
	}
	@Override
	public void write(DataOutput out) throws IOException {
		// TODO Auto-generated method stub
		patentNo.write(out);
		country.write(out);
	}
	@Override
	public int compareTo(patent o) {
		// TODO Auto-generated method stub
		int cmp = country.compareTo(o.country);
		if(cmp!=0)
			return cmp;
		else
			return patentNo.compareTo(o.patentNo);
	}

}
```

Your value class should import “WritableComparable” interface

compareTo() method must be overridden.

Our compareTo() method first compares two Texts (country name) . if Both country names are same, further comparison is made bu comparing patentNo(LongWritable)

Defining mapper class

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

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

enum missing{      COUNTRY,
                   FIRST,
                   MIDDLE,
                   LAST
                   }
enum Total{        COUNT,
                   WRITTEN,
                   SKIPPED}
public class map_class extends Mapper<LongWritable, Text, patent, name> {
      public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException{
            String line = value.toString();
            StringTokenizer tokens = new StringTokenizer(line,",");
            long pat_no = 0;
            String last=" ",first=" ",middle=" ",country=" ";
            String token=null;
            token = tokens.nextToken();
            if(token.length()!=8){
                  pat_no = Long.parseLong(token.substring(0, token.length()));
                  token = tokens.nextToken();
                  if(token.length()>1)
                        last = token.substring(1, token.length()-1);
                  else
                        context.getCounter(missing.LAST).increment(1);
                  token = tokens.nextToken();
                  if(token.length()>1)
                        first = token.substring(1, token.length()-1);
                  else
                        context.getCounter(missing.FIRST).increment(1);
                  token = tokens.nextToken();
                  if(token.length()>1)
                        middle = token.substring(1, token.length()-1);
                  else
                        context.getCounter(missing.MIDDLE).increment(1);
                  for(int i  = 0 ;i < 5;i++){
                        token = tokens.nextToken();
                  }
                  if(token.length()>1)
                        country = token.substring(1, token.length()-1);
                  else
                        context.getCounter(missing.COUNTRY).increment(1);
                  patent p = new patent(pat_no,country);
                  name n = new name(first,middle,last);
                  context.write(p,n);
                  context.getCounter(Total.WRITTEN).increment(1);}
            else
                  context.getCounter(Total.SKIPPED).increment(1);
            context.getCounter(Total.COUNT).increment(1);
      }

}
```

"missing" counter keeps track of records whose either mentioned fields are missing in data set.

"total" counter keeps track of written and skipped recoreds.

Our defined key and value classes as parameters to mapper

Extracting required fields for our custom defined key and value pairs using substring() method.

Incrementing countes through context()

Defining reducer class

```
import java.io.IOException;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class reduce_class extends Reducer<patent, name, NullWritable, Text> {
      public void reduce(patent key,Iterable<name> values,Context context) throws
IOException, InterruptedException{
            String n;
            NullWritable out = NullWritable.get();
            n = key.country.toString() + " " + key.patentNo.toString();
            context.write(out,new Text("----------------------------------------"));
            context.write(out, new Text(n));
            for(name nn : values){
                  n = nn.first.toString()+" "+nn.middle.toString()+"
"+nn.last.toString();
                  context.write(out, new Text(n));
            }
            context.write(out,new Text("----------------------------------------"));
      }
}
```

Defining runner class

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.*

public class runner {
      public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException {
            Configuration conf = new Configuration();
            conf.set("heading", "This involves custom writables and partitioners");

            Job job = new Job(conf);
            job.setJarByClass(runner.class);

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

            job.setMapperClass(map_class.class);
            job.setReducerClass(reduce_class.class);

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

            job.setMapOutputKeyClass(patent.class);
            job.setMapOutputValueClass(name.class);

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


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

Defining out custom key and value classes

Sample Input

```
3858829,"Mischenko","Nicholas","","","","Chicago","IL","US","",3
3858829,"Prelletz","Edward","R.","","","Chicago","IL","US","",4
3858830,"Deniega","Jose","Castillo","","","Elmhurst","NY","US","",1
3858831,"Halwes","Dennis","R.","","","Arlington","TX","US","",1
3858832,"Baker","Terry","M.","","","Santa Barbara","CA","US","",1
3858833,"Fink","Robert","","","6780 Tanglewood Dr.","Youngstown","OH","US","44512",1
3858834,"Eimen","Shawn","H.","","","Cudahy","WI","US","",1
3858835,"Baren","Louis","","","","Chicago","IL","US","",1
3858836,"Marcyan","Stanley","T.","","515 W. Windsor Rd.","Glendale","CA","US","91204",1
3858837,"Merritt","William","C.","","Valley Rd., R.D.","Mansfield Township,","NJ","US","07863",1
3858838,"Woodhouse","William","E.","","2115 E. 61st Ave.","Vancouver 16, Britis","","CA","",1
3858839,"Bowman","Grover","L.","","P.O. Box 84","Bayside","CA","US","95524",1
3858840,"Kell","Nathaniel","B.","","","Indianapolis","IN","US","",1
3858841,"Haynes","Larry","E.","","23730 Via Kannela","Valencia","CA","US","91355",1
3858842,"Yoshimura","Zyunziro","","","","Okazaki","","JP","",1
3858843,"Hartmann","Leonard","Joseph","","","Maplewood","MO","US","",1
3858844,"Lewis","Ivor","J.","","","Springfield","PA","US","",1
3858844,"McCloskey","Thomas","H.","","","Aston","PA","US","",2
3858845,"Grote","Hugo","","","","Wette","","DE","",1
3858845,"Peithmann","Ludolf","","","","Hagen","","DE","",2
3858846,"Schmid","Josef","","","Liebnizstrabe 18","8900 Augsburg 22","","DE","",1
3858847,"Chambers","Henry","B.","","","Santa Inez","CA","US","",1
3858848,"MacFetrich","Robert","H.","","","Charlotte","NC","US","",1
3858849,"Peirce","Benjamin","F.","Jr.","1040 S.W. 67th Ter.","Plantation","FL","US","33317",1
3858850,"Maxcy","Frederic","R.","","","Ellicott City","MD","US","",1
3858850,"G
```

Running the MapReduce Program

```
[training@localhost Desktop]$ hadoop jar custom.jar runner /user/training/MR/custom/ainventor.txt /user/training/MR/custom/out_customwritables
14/03/05 02:23:09 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
14/03/05 02:23:09 INFO input.FileInputFormat: Total input paths to process : 1
14/03/05 02:23:09 WARN snappy.LoadSnappy: Snappy native library is available
14/03/05 02:23:09 INFO snappy.LoadSnappy: Snappy native library loaded
14/03/05 02:23:10 INFO mapred.JobClient: Running job: job_201403050150_0004
14/03/05 02:23:11 INFO mapred.JobClient:  map 0% reduce 0%
14/03/05 02:23:24 INFO mapred.JobClient:  map 11% reduce 0%
14/03/05 02:23:27 INFO mapred.JobClient:  map 12% reduce 0%
14/03/05 02:23:30 INFO mapred.JobClient:  map 13% reduce 0%
14/03/05 02:23:33 INFO mapred.JobClient:  map 16% reduce 0%
14/03/05 02:23:34 INFO mapred.JobClient:  map 20% reduce 0%
14/03/05 02:23:37 INFO mapred.JobClient:  map 23% reduce 0%
14/03/05 02:23:43 INFO mapred.JobClient:  map 32% reduce 0%
14/03/05 02:23:46 INFO mapred.JobClient:  map 33% reduce 0%
14/03/05 02:23:49 INFO mapred.JobClient:  map 36% reduce 0%
14/03/05 02:23:52 INFO mapred.JobClient:  map 39% reduce 0%
14/03/05 02:24:12 INFO mapred.JobClient:  map 45% reduce 13%
14/03/05 02:24:13 INFO mapred.JobClient:  map 51% reduce 13%
```

Reduce method cannot start unless all the maps are completed. Here start of reduce before map phase completion refers to datamovement for reduce phase.

```
Job Counters
  Launched map tasks=5
  Launched reduce tasks=1
  Data-local map tasks=5
  Total time spent by all maps in occupied slots (ms)=217178
  Total time spent by all reduces in occupied slots (ms)=102112
  Total time spent by all maps waiting after reserving slots (ms)=0
  Total time spent by all reduces waiting after reserving slots (ms)=0
```

Total 5 map tasks (meaning 5 input splits)

```
14/03/05 02:25:43 INFO mapred.JobClient:   Total
14/03/05 02:25:43 INFO mapred.JobClient:     COUNT=4301230
14/03/05 02:25:43 INFO mapred.JobClient:     SKIPPED=1
14/03/05 02:25:43 INFO mapred.JobClient:     WRITTEN=4301229
14/03/05 02:25:43 INFO mapred.JobClient:   missing
14/03/05 02:25:43 INFO mapred.JobClient:     COUNTRY=2079
14/03/05 02:25:43 INFO mapred.JobClient:     MIDDLE=66
[training@localhost Desktop]$
```

Our UserDefined Counters

Output Files:

```
----------------------------------------
----------------------------------------
AU 4339458
David F. O'Keefe
George  Holan
----------------------------------------
----------------------------------------
AU 4339484
Geoffrey L. Harding
----------------------------------------
----------------------------------------
AU 4339954
Alan G. Pettigrew
----------------------------------------
----------------------------------------
AU 4340350
Karl S. Springborn
----------------------------------------
----------------------------------------
AU 4340875
Kevin S. English
----------------------------------------
----------------------------------------
AU 4340945
Diethard  Gothe
----------------------------------------
```