**Solution**

First we are going to solve the first problem that is finding out unique listeners per track.

First of all we need to understand the data, here the first column is UserId and the second one is Track Id. So we need to write a mapper class which would emit trackId and userIds and intermediate key value pairs. so make it simple to remember the data sequence, let's create a constants class as shown below

package com.mr;

|  |
| --- |
| public class LastFMConstants {        public static final int USER\_ID = 0;      public static final int TRACK\_ID = 1;      public static final int IS\_SHARED = 2;      public static final int RADIO = 3;      public static final int IS\_SKIPPED = 4;    } |

Now, let’s create the mapper class which would emit intermediate key value pairs as (TrackId, UserId) as shown below

### **Step 1. Class Creation**

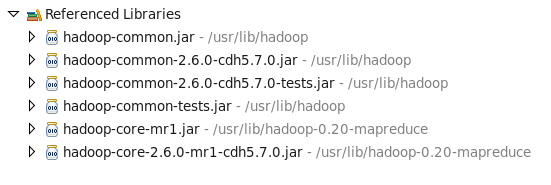
Right click com package->new class-> give class name as "**UniqueListener**” and then

Click **Finish** button

### **Step 2. Add External Jars (Added Already)**

Add JARS file:

Right click “**src**”->**build path->configure build path-> click Libraries pane->add external jars->file system->**



### **Step 3. Type the following MapReduce Program “UniqueListener”**

|  |
| --- |
| public static class UniqueListenersMapper extends Mapper< Object , Text, IntWritable, IntWritable > {          IntWritable trackId = new IntWritable();          IntWritable userId = new IntWritable();    public void map(Object key, Text value, Mapper< Object , Text, IntWritable, IntWritable > .Context context)          throws IOException, InterruptedException {        String[] parts = value.toString().split("[|]");      trackId.set(Integer.parseInt(parts[LastFMConstants.TRACK\_ID]));      userId.set(Integer.parseInt(parts[LastFMConstants.USER\_ID]));          if (parts.length == 5) {          context.write(trackId, userId);      } else {          // add counter for invalid records          context.getCounter(COUNTERS.INVALID\_RECORD\_COUNT).increment(1L);      }      }  } |

You would have also noticed that we are using a counter here named INVALID\_RECORD\_COUNT , to count if there are any invalid records which are not coming the expected format. Remember, if we don't do this then in case of invalid records, our program might fail.

Now let's write a Reducer class to aggregate the results. Here we simply can not use sum reducer as the records we are getting are not unique and we have to count only unique users. Here is how the code would look like

|  |
| --- |
| public static class UniqueListenersReducer extends Reducer< IntWritable , IntWritable, IntWritable, IntWritable> {    public void reduce(IntWritable trackId,Iterable< IntWritable > userIds,Reducer< IntWritable , IntWritable, IntWritable,IntWritable>.Context context)  throws IOException, InterruptedException {          Set< Integer > userIdSet = new HashSet< Integer >();          for (IntWritable userId : userIds) {          userIdSet.add(userId.get());          }          IntWritable size = new IntWritable(userIdSet.size());          context.write(trackId, size);      }  } |

Here we are using Set to eliminate duplicate userIds. Now we can take look at the Driver class

|  |
| --- |
| public static void main(String[] args) throws Exception {          Configuration conf = new Configuration();          if (args.length != 2) {              System.err.println("Usage: uniquelisteners < in >< out >");              System.exit(2);          }          Job job = new Job(conf, "Unique listeners per track");          job.setJarByClass(UniqueListeners.class);          job.setMapperClass(UniqueListenersMapper.class);          job.setReducerClass(UniqueListenersReducer.class);          job.setOutputKeyClass(IntWritable.class);          job.setOutputValueClass(IntWritable.class);          FileInputFormat.addInputPath(job, new Path(args[0]));          FileOutputFormat.setOutputPath(job, new Path(args[1]));          System.exit(job.waitForCompletion(true) ? 0 : 1);          org.apache.hadoop.mapreduce.Counters counters = job.getCounters();          System.out.println("No. of Invalid Records :"                  + counters.findCounter(COUNTERS.INVALID\_RECORD\_COUNT).getValue());      } |

### **Step 4. Export JAR file creation:**

Right click src->Export->Java->JAR File->click Next button

### **Step Music Track Execution:**

[cloudera@quickstart ~]$ hadoop fs -put /home/cloudera/Desktop/UniqueListener.txt /user/cloudera/sriMR

[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/srihadoop/MR/UniqueListener/UniqueListener.jar com.mr. UniqueListener/user/cloudera/sriMR/UniqueListener.txt /user/cloudera/sriMR/UniqueListenerresult