```
import java.io.IOException;

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.lib.NullOutputFormat;
import org.apache.hadoop.util.*;

/*
 * This importer is used to read the raw input data on HDFS and write it to HBase
 * prior to running this, create the appropriate tables in hbase (via hbase shell)
 * create 'stations',{NAME => 'info', VERSIONS => 1}
 * create 'observations, {NAME => 'data', VERSIONS => 1}
 * for this case, we are only interested in the latest version, hence VERSIONS is set to 1.
 */
public class HBaseTemperatureImporter extends Configured implements Tool {

  // Inner-class for map
  static class HBaseTemperatureMapper<K, V> extends MapReduceBase implements
      Mapper<LongWritable, Text, K, V> {
    private NcdcRecordParser parser = new NcdcRecordParser();
    private HTable table;

    public void map(LongWritable key, Text value,
      OutputCollector<K, V> output, Reporter reporter)
    throws IOException {
            // input key-value as TextInputFormat (default), with the byte offset and a long string of
integers containing data about
            // weather station, air temperature etc.
            // using the parser to parse the string and determine if temperature is value
      parser.parse(value.toString());
      if (parser.isValidTemperature()) {
                    // adds valid temperatures to the observations HBase table into the data:air-temp
column
                    // row key generated via custom function as stationID+dateInReverseEpoch
        byte[] rowKey = RowKeyConverter.makeObservationRowKey(parser.getStationId(),
          parser.getObservationDate().getTime());
        Put p = new Put(rowKey);
        p.add(HBaseTemperatureCli.DATA_COLUMNFAMILY,
          HBaseTemperatureCli.AIRTEMP_QUALIFIER,
          Bytes.toBytes(parser.getAirTemperature()));
        table.put(p);
      }
    }

          /*
```

```
         * Creates a HTable instance once against the observations table and use it afterward in
map invocations talking to HBase
         */
    public void configure(JobConf jc) {
      super.configure(jc);
      // Create the HBase table client once up-front and keep it around
      // rather than create on each map invocation.
      try {
        this.table = new HTable(new HBaseConfiguration(jc), "observations");
      } catch (IOException e) {
        throw new RuntimeException("Failed HTable construction", e);
      }
    }

    @Override
    public void close() throws IOException {
      super.close();
      table.close();
    }
  }

  public int run(String[] args) throws IOException {
    if (args.length != 1) {
      System.err.println("Usage: HBaseTemperatureImporter <input>");
      return -1;
    }
    JobConf jc = new JobConf(getConf(), getClass());
    FileInputFormat.addInputPath(jc, new Path(args[0]));
    jc.setMapperClass(HBaseTemperatureMapper.class);
    jc.setNumReduceTasks(0);
    jc.setOutputFormat(NullOutputFormat.class);
    JobClient.runJob(jc);
    return 0;
  }

  public static void main(String[] args) throws Exception {
    int exitCode = ToolRunner.run(new HBaseConfiguration(),
        new HBaseTemperatureImporter(), args);
    System.exit(exitCode);
  }
}
```