```java
package sort.newVME;
import jam.data.*;
import jam.sort.*;

/*
 * Test sort file for ADC at LENA.  Modified 2 March 2002 for
 * example purposes by Dale Visser.
 *
 * @author C. Iliadis
 * @author Dale Visser
 */
public class CI extends SortRoutine {

    /*** GLOBAL DECLARATIONS ***/
    final int ADC_BASE = 0xe0000000;
    final int THRESHOLDS = 100;
    final int ADC_CHANNELS = 4096; //num of channels per ADC
    final int TWO_D_CHANNELS = 512;
    //number of channels per dimension in 2-d histograms

    //amount of bits to shift for compression
    final int TWO_D_FACTOR =
        Math.round((float) (Math.log(ADC_CHANNELS / TWO_D_CHANNELS) / Math.log(2.0)));

    int idGe, idNaI, idTAC;//  id numbers for the signals

    Histogram hGe, hNaI, hTAC;//  ungated 1D spectra
    Histogram hGeNaI;//  ungated 2D spectra
    Histogram hGe_TAC;//    gated on TAC
    Histogram hGe_g2d, hTAC_g2d;//  gated on Ge vs. NaI

    Gate gTAC;//  1D gate
    Gate gGeNaI;//2D gate

    Scaler sClock, sBeam, sGe, sAccept, sNaI;//  scalers

    Monitor mDeadTime;
    final String DEAD_TIME = "Dead Time (%)";
    int lastGe, lastAccept;//for calculating dead time

    /*** END OF GLOBAL DECLARATIONS ***/

    /**
     * Method called to initialize objects when the sort routine is loaded.
     */
    public void initialize() throws Exception {

        /*** SCALER SECTION ***/
        sClock = new Scaler("Clock",0);// (name, position in scaler unit)
        sBeam = new Scaler("Beam",1);
        sGe = new Scaler("Ge",2); //Ge provides trigger
        sAccept = new Scaler("Ge Accept",3);
        sNaI = new Scaler("NaI",4);
        vmeMap.setScalerInterval(3);//insert scaler block in event data every 3 seconds

        /*** MONITOR SECTION ***/
        //Monitors associated with scalers, window will return scaler rate in Hz
        Monitor mClock = new Monitor(sClock.getName(), sClock);
        Monitor mBeam = new Monitor(sBeam.getName(), sBeam);
        Monitor mGe = new Monitor(sGe.getName(), sGe);
        Monitor mAccept = new Monitor(sAccept.getName(), sAccept);
        Monitor mNaI = new Monitor(sNaI.getName(), sNaI);
        //Monitor associated with Gate, window will show rate of new counts in Hz
        Monitor mTAC = new Monitor("TAC window", gTAC);
        //User-defined monitor which is calculated in this sort routine
        mDeadTime=new Monitor(DEAD_TIME,this);

        /*** ADC CHANNELS SECTION ***/
        // eventParameters, args = (slot, base address, channel, threshold channel)
        idGe = vmeMap.eventParameter(2, ADC_BASE, 0, THRESHOLDS);
        idNaI = vmeMap.eventParameter(2, ADC_BASE, 1, THRESHOLDS);
        idTAC = vmeMap.eventParameter(2, ADC_BASE, 2, THRESHOLDS);

        /*** HISTOGRAM SECTION ***/
        hGe = new Histogram("Ge", HIST_1D_INT, ADC_CHANNELS, "Germanium");
        hNaI = new Histogram("NaI", HIST_1D_INT, ADC_CHANNELS, "NaI");
        hTAC = new Histogram("TAC", HIST_1D_INT, ADC_CHANNELS, "TAC");

        hGe_TAC = new Histogram("Ge-TAC", HIST_1D_INT, ADC_CHANNELS,
        "Germanium, gated on TAC");
        hGe_g2d = new Histogram("Ge-2dgate", HIST_1D_INT, ADC_CHANNELS,
        "Germanium--gated on NaI vs Ge");
        hTAC_g2d = new Histogram("TAC-2dgate", HIST_1D_INT, ADC_CHANNELS,
        "TAC--gated on NaI vs Ge");

        hGeNaI = new Histogram("GeNaI", HIST_2D_INT, TWO_D_CHANNELS,
        "NaI vs. Germanium", "Germanium", "NaI");

        /*** GATE SECTION ***/
        gTAC = new Gate("TAC", hTAC);
        gGeNaI = new Gate("GeNaI", hGeNaI);
        hTAC_g2d.addGate(gTAC);
    }//end of initialize()

    /**
     * Method for sorting of data into spectra.
     */
    public void sort(int[] data) throws Exception {
        /*** EXTRACT DATA FROM ARRAY ***/
        int eGe = data[idGe];
        int eNaI = data[idNaI];
        int eTAC = data[idTAC];
        int ecGe = eGe >> TWO_D_FACTOR;//bit-shifts are faster than division
        int ecNaI = eNaI >> TWO_D_FACTOR;

        /*** INCREMENT UNGATED SPECTRA ***/
        hGe.inc(eGe);
        hNaI.inc(eNaI);
        hTAC.inc(eTAC);
        hGeNaI.inc(ecGe, ecNaI);// inc(x-channel, y-channel)

        /*** INCREMENT GATED SPECTRA ***/
        if (gTAC.inGate(eTAC)) hGe_TAC.inc(eGe);
        if (gGeNaI.inGate(ecGe, ecNaI)) {
            hGe_g2d.inc(eGe);
            hTAC_g2d.inc(eTAC);
        }
    }

    /**
     * Method for calculating values of user-defined monitors.
     */
    public double monitor(String name) {
        double rval = 0.0;
        if (name.equals(DEAD_TIME)){
            double Ge_val = (double)sGe.getValue();
            double Accept_val = (double)sAccept.getValue();
            rval = 100.0*( 1.0 - (lastAccept - Accept_val)/(Ge_val - lastGe) );
            lastGe = (int)Ge_val;
            lastAccept = (int)Accept_val;
        }
        return rval;
    }

}//end of class Cl
```