```
package presentation.epiphyte;

import java.util.TreeSet;

/** manage traps and tagged rodents list, computes the global indicators from the retrieved
individual values indicator definitions
 * @see C_RodentBandia
 * @author Le Fur and Diakhate, sept.2013 */

public class C_InspectorBandia extends A_Inspector {

    // FIELDS
    private double currentDRS, currentDMR;
    private int currentMNA;
    protected TreeSet<C_Trap> trapList = new TreeSet<C_Trap>();
    public static TreeSet<C_RodentBandia> taggedRodentList = new TreeSet<C_RodentBandia>();
    public int taggedRodentsNumber = 0; // current increment of tagged rodents,
    private C_OutputData dataSaverCMR;// Writer of an outer csv file

    // CONSTRUCTOR
    public C_InspectorBandia() {
        super();
        resetIndicators();
        taggedRodentsNumber = 0;
        dataSaverCMR = new C_OutputData("CMR.csv");
        dataSaverCMR.writeln("tick;date;popSize;NbTaggedRodents;DRS;DMR;MNA");
    }

    // METHODS
    /** close private files */
    @Override
    public void closeSimulation() {
        dataSaverCMR.closeFile();
    }

    /** stores the current state of indicators - JLF 10.2013 */
    public void storeCMRValues() {
        dataSaverCMR.writeln(RepastEssentials.GetTickCount() + ";" + C_Calendar.fullDate + ";"
                + C_InspectorPopulation.listRodents.size() + ";" + taggedRodentList.size() +
";"
                + getCurrentDRS() + ";" + getCurrentDMR() + ";" + getCurrentMNA());

    }

    public void resetIndicators() {
        currentDRS = 0.;
        currentDMR = 0.;
        currentMNA = 0;
    }

    /** Mean distance between successive catches within a given session (DRS: "Distance entre
Recaptures Successives")
     * @param session : the session number */
    public void computeDRS(int session) {
        double sumDRS = 0., rodentDRS = 0.;
        int nbTaggedRodents = taggedRodentList.size();
        for (C_RodentBandia rodent : taggedRodentList) {
            rodentDRS = rodent.computeDRS(session);
            if (rodentDRS == 0.) nbTaggedRodents--;// DRS = 0 when rodent has not been
recaptured
            else sumDRS += rodentDRS;
        }
        currentDRS = sumDRS / nbTaggedRodents;
    }
    /** compute every distances between catches and keep the largest one within a session
(DMR: Distance maximum de recapture) */
    public void computeDMR(int session) {
        double dmr = 0., rodentDMR = 0.;
        for (C_RodentBandia rodent : taggedRodentList) {// taggedRodents have been catched at
least 1 time
            rodentDMR = rodent.computeDMR(session);
            if (dmr < rodentDMR) dmr = rodentDMR;
        }
        currentDMR = dmr;
    }
    /** MNA : Minimum Number Alive for each session */
    public void computeMNA(int session) {
        int mna = 0;
        for (C_RodentBandia rodent : taggedRodentList) {
            if (rodent.aliveInSession(session)) mna++;
        }
        currentMNA = mna;
    }
    /** Give a tag number to rodent */
    public void tag(C_RodentBandia rodent) {
        if (taggedRodentList.add(rodent)) {
            taggedRodentsNumber++;
            rodent.setTag(taggedRodentsNumber);
        }
    }
    /** Add trap in trapList */
    public void addTrap(C_Trap oneTrap) {
        trapList.add(oneTrap);
    }

    // GETTERS AND SETTERS

    public TreeSet<C_Trap> getTrapList() {
        return trapList;
    }
    public double getCurrentDRS() {
        return currentDRS;
    }
    public double getCurrentDMR() {
        return currentDMR;
    }
    public double getCurrentMNA() {
        return currentMNA;
    }
}
```