```java
package presentation.epiphyte;

import java.util.Calendar;
import java.util.Iterator;
import java.util.TreeSet;

import presentation.dataOutput.C_OutputData;
import repast.simphony.essentials.RepastEssentials;
import thing.C_Rodent;
import thing.dna.C_GenePair;
import thing.dna.C_GenomeEucaryote;
import thing.dna.C_XsomePairMicrosat;

import com.vividsolutions.jts.geom.Coordinate;

import data.I_sim_constants;

/** Retrieves information on genes and alleles of the whole population listRodents is obtained from the population
 * inspector and updated when needed. This implementation to avoid permanent calls to C_InspectorPopulation.listRodents
 * static field
 * @see C_InspectorPopulation#listRodents
 * @author A Realini 05.2011 */
public class ExtractFromC_InspectorGenetic extends A_Inspector {

    protected TreeSet<C_Rodent> listRodents = C_InspectorPopulation.listRodents; // must be
    protected int coefPourEcriture = 0;// Compteur pour l'écriture du fichier genePop
    protected C_OutputData genePopFile;

    public ExtractFromC_InspectorGenetic() {
        super();
        genePopFile = new C_OutputData("GenePop");
        initTabGenePopFile();
    }
    @Override
    public void step() {
        // save private files
        if (RepastEssentials.GetTickCount() == I_sim_constants.INTERVAL_ECRITURE_GENE_POP * coefPourEcriture) {
            recordGenePopInFile();
            coefPourEcriture++;
        }
    }

    /** Initialise les titres pour le fichier genePop */
    private void initTabGenePopFile() {
        genePopFile
                .writeln(Calendar.getInstance().getTime() + " Run n°" + genePopFile.numRun + " | Espèce: ");
        for (int i = 0; i < I_sim_constants.NUMBER_GENES; i++)
            genePopFile.writeln("locus" + i);
    }

    /** Ecrit les données de gène pop dans le fichier GenePop */
    private void recordGenePopInFile() {
```

```
53          C_GenePair[] paire = new C_GenePair[I_sim_constants.NUMBER_GENES];
54          genePopFile.writeln("Pop " + RepastEssentials.GetTickCount());
55          Iterator<C_Rodent> rodents = listRodents.iterator();
56          while (rodents.hasNext()) {
57              C_Rodent microtus = rodents.next();
58              Coordinate point = microtus.getCoord_Umeters();
59              genePopFile.write(point.x + " " + point.y + " , ");
60              for (int locus = 0; locus < I_sim_constants.NUMBER_GENES; locus++) {
61                  C_XsomePairMicrosat microsat = ((C_GenomeEucaryote) microtus.getGenome()).getMicrosatXsome();
62                  paire[locus] = (C_GenePair) microsat.getLocusAllele(locus);
63              }
64              genePopFile.writeln(paire[0] + " " + paire[1] + " " + paire[2]);
65          }
66      }
67      // close private files
68      public void closeSimulation() {
69          genePopFile.closeFile();
70      }
71  }
72
```