**Bluetooth Analysis Project - Part 3: Class Files**

**BTmain**

// BTmain

// creates analysis and specifies which graphs to display

**public** **class** BTmain {

 **public** **static** **void** main(String[] args) {

 BTanalysis analysis = **new** BTanalysis("BTdata1.csv");

 analysis.contactGraph("Contact Graph");

/\*

 analysis.infectionGraph("User01", 0.2);

 analysis.infectionGraph("User02", 0.2);

 analysis.infectionGraph("User03", 0.2);

\*/

 **for**(**int** i=1; i<=5; i++) {

 analysis.infectionGraph("User01", 0.3);

 }

 }

}

**BTanalysis**

// BTanalaysis

// reads data and creates graphs

**public** **class** BTanalysis {

 // contact data, graph of data, number of users in data

 BTdata data;

 BTgraph graph;

 **int** userCount;

 // constructor gets the data from the text file

 **public** BTanalysis2(String filename) {

 data = **new** BTdata(filename);

 // System.out.println(data);

 }

 // creates a graph from all contact data

 **public** **void** contactGraph(String name) {

 System.*out*.println(name);

 graph = **new** BTgraph();

 BTuser user1, user2;

 BTcontact contact;

 data.resetIndex();

 **while**(data.hasNext()) {

 contact = data.readNext();

 // System.out.println(contact);

 user1 = **new** BTuser(contact.getUserID());

 user2 = **new** BTuser(contact.getSeenID());

 graph.addVertex(user1);

 graph.addVertex(user2);

 graph.addEdge(contact);

 }

 graph.viewGraph("Contact Graph");

 // gets the userCount from graph

 // this should really be moved to BTdata or BTgraph

 userCount = graph.getVertexSize();

 System.*out*.println(userCount);

 }

 // creates a graph of disease transmission

 // infection starts with the initial infected user

 // infectiousness is a percent probability of infection for each contact

 // prints disease transmission log to the console

 **public** **void** infectionGraph(String infectedUserID, **double** infectiousness) {

 System.*out*.println("\ninfectionGraph: " + infectedUserID + " " + infectiousness);

 graph = **new** BTgraph();

 BTcontact contact;

 BTuser user1, user2;

 **int** infectionCount = 0;

 data.resetIndex();

 **while**(data.hasNext() && infectionCount < userCount) {

 contact = data.readNext();

 System.*out*.println(contact);

 user1 = **new** BTuser(contact.getUserID());

 **if** (user1.getUserID().equals(infectedUserID) && infectionCount==0) {

 user1.setInfected();

 infectionCount++;

 }

 user2 = **new** BTuser(contact.getSeenID());

 **if** (user2.getUserID().equals(infectedUserID) && infectionCount==0) {

 user2.setInfected();

 infectionCount++;

 }

 graph.addVertex(user1);

 graph.addVertex(user2);

 graph.addEdge(contact);

 **if** (graph.checkUserInfected(user1.getUserID()) && !graph.checkUserInfected(user2.getUserID())) {

 **if** (Math.*random*() < infectiousness) {

 graph.setUserInfected(user2.getUserID());

 user2.setInfected();

 infectionCount++;

 System.*out*.println(user1.getUserID()+" infects " + user2.getUserID());

 } **else** {

 System.*out*.println(user1.getUserID()+" infection of " + user2.getUserID() + " fails");

 }

 }

 **if** (graph.checkUserInfected(user2.getUserID()) && !graph.checkUserInfected(user1.getUserID())) {

 **if** (Math.*random*() < infectiousness) {

 graph.setUserInfected(user1.getUserID());

 user1.setInfected();

 infectionCount++;

 System.*out*.println(user2.getUserID()+" infects " + user1.getUserID());

 } **else** {

 System.*out*.println(user2.getUserID()+" infection of " + user1.getUserID() + " fails");

 }

 }

 }

 graph.viewGraph("infectionGraph: " + infectedUserID + " " + infectiousness);

 }

}