Author: Christopher Tam

**Entity class**

For this class, I tested the getter setter methods by seeing if the correct values were returned and if values were set correctly. I tested the constructor by seeing if the class fields were correctly initialized by the constructor. For the toString and equals method, I tested to see if the correct string was returned, as well as if two entity objects were compared correctly – via their name fields. For the getRelations method, I tested to see if an entity object added to the relations list would equal whatever element was returned from the front of the relations list. If the both objects were equal, then that means that the method returns the relations list correctly. Ultimately I tested to see if values (and elements) were set correctly.

**Network class**

For this class, I tested the constructor and getter methods the same as in the entity class, however the other methods were of a more complex nature. I tested the addRelation, removeRelation, and addRelations (that takes a string) methods by seeing if each method essentially manipulated entity objects from the network list correctly. That is, if the method were to remove an entity from the list, I checked to see if that entity’s spot in the list contained a ‘null’ value after the method was called, and if the method were to add any entities to a list, I did the same test as for the getRelations method of the entity class.

To test the addRelations method that takes a file, I performed the same test as for the getRelations method, except manually since I could junit test that method; I checked to see if the names from a text file were added to the network as an entity relation.

**PropogationStudy class**

For this class, the constructor, getter, and setter methods were tested the same as in the entity class. For the infect and inoculate methods that take an entity parameter, I tested them by seeing if an entity in the network equal to the parameter entity has its health status changed to the respective health status of the method (either infected or inoculated) after the method is called.

For the infect and inoculate methods that take a double parameter, I did not junit test them due to the random numbers. Instead, to test them I would adjust to the method to save the Math.random() value generated to a variable, and print that variable to see if that method performed correctly or not according to the value.

For the getNumUninfected, getNumInoculated, and getNumInfected methods, I would test them by creating networks with a set number of entities with the appropriate health status. I would call the method and see if the int value returned matches the set number (to call the method I also created a PropogationStudy object)..

To test the resetNetwork method, I would simply see if each entity in a propogation study network which had a health status not equal to Uninfected would have an Uninfected health status after the method was called.

Due to the system.out.println function I could not test the printNetwork method with junit testing. Instead, I created a network of entities and would see if each entity, along with its health status, was printed correctly and in the correct order by the method (to call the method I also created a PropogationStudy object).

I could not junit test the runPropogation method due to the use of random numbers. As such to test it I would create a propogation study network containing each of the health statuses to be affected by the method, run the method, and see if the entities containing each health status are affected appropriately according to the method.

To test the main method of the class, I used a combination of the same testing methods previously described, consisting of checking if LinkedList elements as well as their healthstatuses are correctly modified/added after certain methods are called and if correct entities are created after reading from a text file.