Benson Tran
10/20/14

README: Khan Academy Coding Challenge

Instructions

Note: Preferably it would be best to import the project and run it on the Eclipse IDE.

This project allows the user to implement infections by typing commands into the console:

- 'print' – prints the list of classrooms (teacher & student combinations) with the current infections.
- 'end' – quits the current execution of the program
- 'total <String> <String>' – this performs a total infection.
    - The first string denotes the name of the infection.
    - The second string denotes the person we want the infection to start from.
        - e.g. total version1 Alan
- 'limited <String> <int>' – this performs a limited infection.
    - The string denotes the name of the infection.
    - The integer denotes the number of people we hope to infect.
        - e.g. total version1 4

Design

Six classes were implemented for this challenge:

Node – This class serves as either a 'teacher' or 'student'. This class also contains information such as the node's name, whether or not the node is infected, the current version of such infection, and the grade of the node, given that the node is a student. Each node also contains a unique number, which is useful for markings during traversals.

Graph – The Graph Class essentially serves as a representation of classrooms. This graph contains:

- an adjacency list, which is implemented via Map<Node, List<Node>>.
    - Teachers map to a list of students.
    - Students map to a list of a single teacher.
    - Students do not map to other students and are not intended to map to multiple teachers.
- reference maps for teachers and students: Map<String, Node>.
    - These maps are useful for not only retrieving the corresponding node for a particular name, but also useful for checking whether a student or teacher exists.
- void print() method that prints current status of classrooms
- (1) void findBestStudent(Node teacher),
- (2) void createNewClassroom(Node teachingAssistant, Node oldTeacher, int infectCount),
- (3) void studentToTeacher(Node teachingAssistant, Node oldTeacher)
- The three methods above provide the framework required to implement limited infection.

Infection – Performs total infection or limited infection.

- void totalInfection(String name)
    - Depending on whether the name is that of a student or teacher, a depth first search is performed to infect.
- void limitedInfection(int count)
    - Unlike totalInfection(), which takes a name, limitedInfection() takes an integer and searches through the sizes of the classrooms to ascertain whether a large enough classroom exists to infect. This is done via updateNumberOfStudents() and updateNumberOfStudents().
        - If a classroom is of exact size to the size we want to infect, then we perform a totalInfection() on that classroom.
        - If a classroom is of larger size, we 'find the best student' via the student's grade and create a new classroom with that 'best student' as the teacher and add students to this new classroom.

Parser – Parses an input text file and generates data structures required to create an instance of a Graph.

- Map<Node, List<Node>> parseInput(String fileName)
    - Creates adjacency list

Command – Parses input from console line and runs commands. There are two different types of commands: graph commands ('print' & 'end') and infection commands ('total' & 'limited'). Reflection is implemented to run certain methods based on the commands parsed

- void runCommand(String[] command, Graph classrooms)
    - void runGraphCommands(String[] command, Graph classrooms)
    - runInfectionCommands(String[] command, Graph classrooms)

Main – Runs program.

Areas for Improvement

- Many more commands can be implemented to provide for more robust testing. For instance, a 'revert' command can be implemented to reset to the original graph before infections. Moreover, commands for finding a student's grade or moving students to and from classrooms can be useful for limited infections.

- On the note of limited infections, a 'merge' command could be implemented. It might be useful to merge classes into a single class in the case that we want to infect a larger population than is given in any single classroom. Also, instead of the current implementation for limited infection, which creates a new classroom (new teacher and new students), we can

move students into a class until the correct size is reached for an infection to occur. Also, implementing a limited infection via name rather than count might be a possibility then.

Testing

Classrooms.txt is provided for testing. It should reside in the workspace directory. TestResults.txt & TestResults2.txt are provided to check testing cases.