FLORIDA SOUTHERN COLLEGE

Data Structures
CSC 3280

Fall 2021

Program 5: FSC Sharing is Caring (BST)
Assigned: Tuesday, November 16, 2021
Due: Thursday, December 2nd, 2021 by 11:59 PM
(Monday, Tuesday 23, 2021 by 5:00pm for the comment file)

Purpose:

1. Practice the implementation and use of trees.

Read Carefully:

- This program is worth 7% of your final grade.

- **WARNING**: This is an individual assignment; you must solve it by yourself. Please review the definition of cheating in the syllabus, the FSC Honor Code, and the serious consequences of those found cheating.
    - **The FSC Honor Code pledge MUST be written as a comment at the top of your program**.

- When is the assignment due? The date is written very clearly above.
    - **Note:** once the clock becomes 11:59 PM, the submission will be closed! Therefore, in reality, you must submit by 11:58 and 59 seconds.

- LATE SUBMISSION: you are allowed to make a late submission according to the rules defined in the syllabus. Please see course syllabus for more information.

- **Canvas Submission**:
    - This assignment must be submitted online via Canvas.
    - Within IntelliJ, your project should be named as follows:
        - `FSCsharingiscaring`
        - You should NOT have a package
    - You should submit a single **ZIP** file, which has **ONLY** your Java files inside it.
    - You should NOT submit the entire IntelliJ project.

- **Coding Rooms Submission**:
    - This assignment must ALSO be submitted at Coding Rooms in order to receive credit.

Program 5: FSC Internet Sharing System

Purpose

Learn to implement the functionality of a binary search tree and to combine that functionality with linked lists.

The Problem

The FSC, college-wide wireless internet connection often works fine, but many students complain that the range and quality is very poor inside some buildings. Therefore, a group of "techy" CS students have come up with a solution: **FSC Sharing is Caring** (**FSCsc** for short). All students participating in FSCsc must allow their laptops to act as Ad-Hoc Wireless Networks (aka Virtual Routers or Wireless Hotspots), thereby extending the "range" of the FSC wireless internet connection. Note: although all participating students must allow their laptops to act as Wireless Hotspots for other students, each student (owner of laptop and administrator of their own laptop's Wireless Hotspot) can choose whose laptops can (or cannot) connect (LINK) to their Wireless Hotspot.

> Example:
> Students A, B, and C all participate in the FSCsc program. Let us assume that there is an official, FSC hotspot in Weinstein with a range of 50 meters. Students B and C, whose laptops are 60 meters from said hotspot in Weinstein, will have no internet access, as they are 10 meters out of range. However, if Student A is sitting 45 meters away from the FSC hotspot and, therefore, has internet connection, Students B and C could possibly still get wireless access **IF** Student A has allowed them access to their Wireless Hotspot **and IF** they are in range of the Hotspot from Student A.

As shown in the example above, there are **two conditions** for Student B to share the internet connection from Student A's Wireless Hotspot:

1) They must be LINKed together, meaning that Student B must be added to the allowed list of those who can access Student A's Wireless Hotspot

2) Student B must be within the broadcast range of Student A's Wireless Hotspot.

   Note: range is solely determined based off of the broadcasting capability of the laptop broadcasting the Wireless Hotspot. In this example, if Student A's broadcast range was 10 meters, only those students within 10 meters of Student A could connect to Student A's Wireless Hotspot.

For the sake of this assignment, each laptop will have a limited **broadcast** range. **However**, we will assume that all laptops have an unlimited range with respect to accessing the Wireless Hotspot of another student. Basically, this stipulation simplifies the problem, limiting the range calculation to only the range of the broadcasting laptop.

To be clear, we give one final example:
Student A and Student B both participate in the FSCsc program, and they are LINKed together. Assume they are 7 meters apart, and the broadcast ranges of Student A and B are 10 meters and 5 meters, respectively. Therefore, Student B could use the internet connection of Student A, because Student B is within the 10 meter broadcasting range of Student A. However, Student A could not use the internet connection of Student B, because Student A is not within the 5 meter broadcasting range of Student B.

Range Calculation:
You will determine the range between two laptops based on the Euclidean distance between them. As a reminder, the 2D Euclidean distance between objects, p and q, is given by:

$$EucDis(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Your assignment is to simulate these students/laptops as they move around a 2D grid (x/y coordinate plane) and connect to each other.

Implementation

All students will be stored in a Binary Search Tree based off of their unique MAC address of their laptop (which is just an `int` value). Each student (node) will have several data members, including the MAC address just mentioned, their name, and their x and y coordinates. Also, each node will have a pointer/reference to the head of a linked-list, which will represent the list of all laptops that it is LINKed to.

UML class diagrams are provided at the end of this write-up.

Your FSCsc program will read in a series of instructions from a commands, such as JOIN, QUIT, LINK, UNLINK, etc., and you will need to perform the instruction and then print the appropriate information to the output.

Input Specifications

- File IO will *not* be used for this program. Allow me to repeat: you will *not* read from a file, nor will you write to a file for this program. You will read from the user via System.in and you will print to the standard output (the console).
- That said, sample input and output files have been provided to show you possible input (from the user) and the matching, expected output.

The first line of the input will be a single integer that will be used to seed your random numbers (see MOVEDEVICES command for more info). The second line of the input will have two integers, sizeX and sizeY, representing the size of our 2D coordinate plane. The third line of the input will be an integer, *k*, indicating the number of commands that will be run during the simulation. The following *k* lines will each be a single command, **with the relevant data on the same line**, separated by spaces.

The commands you will have to implement are as follows:

- **JOIN** – A new student/laptop is joining the FSCsc program (a new node will be added to the BST). The command will be followed by the following information: MAC, a non-negative int, which you will use to sort the devices by; first and last name of student; broadcastRange, a non-negative int, which represents the maximum broadcast range of the device; and X and Y, the initial X and Y coordinates of the device. If this student is not already participating in the FSCsc program, they should be added to the BST. Otherwise, an error should be printed (see output).

- **FINDMAC** – Find a device based on its MAC address. This command will be followed by a MAC address to search for. If found in the BST, this command should print the MAC address, the student's name, the current X and Y coordinates of the device, and the number of links it currently has. If said device cannot be found, an error message should be printed (see output).

- **LINK** – This will be followed by two MAC addresses (two ints). When you see this command, this means that the two devices are now linked. Therefore, you should add the MAC address of EACH device to the other's linked list of linked devices (deviceLinks)…SORTED by MACnumber. All connections in the database are always two-way – e.g. if Student A's laptop MAC is in Student B's linked-list of linked (allowed) devices, then Student B's laptop MAC is also in Student A's linked-list of allowed devices. If either MAC in the LINK command is not in the system, or if they are already linked, an error message should be displayed (see output).

- **UNLINK** – This is the opposite of LINK. Like LINK, it will be followed by two MACs. You will have to find both devices in the BST, and remove their MACs from each others' linked-list. Again, if either device is not in the system, or if they are not actually linked, an error message should be displayed (see output).

- **QUIT** – This command is followed by a single MAC address, and it means the user is quitting the FSCsc program (so you must delete this laptop from the BST).

  NOTE: Before you do so, however, you must remove this MAC address from any other laptop's linked-list of allowed laptops. Therefore, for the MAC address you are deleting, you will need to loop through its own linked-list of allowed laptops (linked devices), and run the UNLINK command for each laptop in the list, thereby removing them from each others' lists. If the device you are trying to delete is not in the database already, an error message should display (see output).

- **SHOWCONNECTIONS** – This command is followed by a single MAC address. The command prints the MAC address, the student's name, first and then last, the number of links they have, the number of currently **active** connections (linked **and** in range), and then it prints a numerically ascending list of the MAC address that are currently active, along with other required data of those MAC addresses. If the MAC address is not found in the database, an error message should be printed (see output).

  NOTE: Student A's active connections are defined as those MAC addresses that are (1) found in the linked-list of allowed laptops and (2) that are within the broadcasting range of Student A's laptop. **So here is where you need to use the Euclidean formula to calculate distance between two points.

- **PRINTMEMBERS** – This command prints all participants, in the FSCsc program, in ascending numerical order (based on MAC addresses), along with printing basic information about these devices (see output).

- **MOVEDEVICES** – This command randomly assigns new values to the X and Y coordinates of each device, thereby simulating the random movement of the students/laptops as they travel around campus, resulting in them possibly entering and exiting various Wireless Hotspots. **In code, this command has you travel to each node of the BST, in ascending order based on MAC address. At each node, you first assign a randomly generated X coordinate and then you assign a randomly generated Y coordinate.

As mentioned at the top of this section, the first line of the input file will be an integer that will be used to seed your random numbers; think of this as a starting point for the pseudo-random number generator. So if the seed integer was 8424956, in code, you would create and "seed" the random number generator as follows:

```
Random rng = new Random(8424956);
```

This simple line of code creates an object of type Random and "seeds" it with the number 8424956. Now, if you want to get a NEW random number, you could type the following:

```
int x = rng.nextInt();
```

This line would generate the next random number and then save it into the variable x.

And you can call this method, `.nextInt()`, again and again. Each time it will generate a new random number. ***Important: this random number is not 100% random. Actually, it can be predetermined if the seed is known. Therefore, every time you run the program with the same seed, the numbers generated will be exactly the same. For our program, this is a GOOD thing, as it will guarantee the same output.

Assume you have a reference variable in your program called `currentNode`, and this variable points to a given node in the BST. **The following two lines of code would be used to assign new X and Y coordinates**, respectively:

```
currentNode.setX(rng.nextInt(sizeX));
currentNode.setY(rng.nextInt(sizeY));
```

Where `sizeX` and `sizeY` are the sizes of the X and Y coordinate plane, respectively. As such, a student (device) can move to any location in the X/Y coordinate plane based off of a MOVEDEVICES command.

Output ***WARNING***

Your program MUST adhere to the EXACT format shown in the sample output (spacing capitalization, use of dollar signs, periods, punctuation, etc). As always, the grading inputs will be large and comprehensive, resulting in very large outputs. As such, we will use text comparison programs to compare your output to the correct output. If, for example, you

have two spaces between in the output when there should be only one space, this will show up as an error even through you may have the program correct. You WILL get points off if this is the case, which is why this is being explained in detail. Minimum deduction will be 10% of the grade, as the graders will be forced to go to text editing of your program in order to give you an accurate grade. Again, your output MUST ADHERE EXACTLY to the sample output.

Grading Details:

The grade for this assignment is composed of two parts:

1. 20%: Your submission of a COMMENT file by Monday 11/23 by 5:00pm.
   - This will be an *independent* submission on Canvas.
   - What should you submit? You should submit a SINGLE Java file, which should be named **`FSCsc.java`**.
   - What should be *inside* this Java file?
     Answer: ONLY comments should be inside. And these comments should clearly depict/represent a working solution to this problem. You should make detailed comments for each of the methods you plan to make. And all of these comments should be indented, in just the same way that your program will ultimately be indented.
2. 80%: Your submission of the actual program (working solution) to both Canvas and Coding Rooms by the date specified at the top of this PDF.

Your program code (Part 2) will be graded upon the following criteria:

1) Adhering to the implementation specifications listed on this write-up.
2) Your algorithmic design.
3) Correctness.
4) The frequency and utility of the comments in the code, as well as the use of white space for easy readability. (We're not kidding here. If your code is poorly commented and spaced and works perfectly, you could earn as low as 80-85% on it.)
   ***Please RE-READ the above note on comments.
5) Compatibility to the **newest version** of IntelliJ. (If your program does not compile in IntelliJ, you will get a large deduction from your grade.)
6) Your program should include a header comment with the following information: your name, **email**, date, AND HONOR CODE.
7) Your output MUST adhere to the EXACT output format shown in the sample output.
8) **Use of Binary Search Trees with Linked Lists. Since the purpose of this assignment is to teach you to Binary Search Trees, you will only receive credit if you use your own BSTs (based on the starter code provided).**

Deliverables

You should submit a zip file with **FIVE** files inside:

1. `FSCstudent.java`
   Class to create objects of type FSC student. These objects will be nodes of the BST.
2. `FSCscBST.java`
   Class to create BST of `FSCstudent` objects (THIS class is your BST class)
3. `FSCscDevice.java`
   Class to create objects of type device (to be used in the linked list of devices)
4. `FSCscLinkedDevices.java`
   Class to create linked list of devices
5. `FSCsc.java`
   This is your main program.

NOTE: your name, ID, section and EMAIL should be included as comments in all files!

UML Diagrams on the next two pages.

Here are the UML Diagrams for the required classes:

FSCstudent

Data Members

```
private String firstName;
private String lastName;
private int MACnumber;
private int broadcastRange;
private int x;
private int y;
private int numLinks;
private FSCscLinkedDevices allowedMACs;
private FSCstudent right;
private FSCstudent left;
```

Operations/Methods

CONSTRUCTOR (one or more constructors as needed)
public String getFirstName( )
public String getLastName( )
public int getMACnumber( )
public int getBroadcastRange( )
public int getX( )
public int getY( )
public int getNumLinks( )
public FSCscLinkedDevices getAllowedMACs( )
public FSCstudent getRight( )
public FSCstudent getLeft( )
public void setFirstName(String)
public void setLastName(String)
public void setMACnumber(int)
public void setBroadcastRange(int)
public void setX(int)
public void setY(int)
public void setAllowedMACs(FSCscLinkedDevices)
public void setRight(FSCstudent)
public void setLeft(FSCstudent)

DETAILS:

- This class is for NODES of the BST
- `firstName` and `lastName` should be obvious
- `MACnumber` is the MAC address of this student's computer
- `broadcastRange` is the range (distance) that this student's computer can wireless share internet
- `x` and `y` are the current X and Y coordinates of this Student
- `numLinks` is the number allowed MACs that are in the student's linked list of allowed MACs
- `allowedMACs` is a pointer/reference to the HEAD of a linked list. This linked list represents the list of MACs (devices) that this student has allowed to share their internet.
- `right` is the pointer to the right child
- `left` is the pointer to the left child

FSCscBST

Data Members

```
private FSCstudent root;
```

Operations/Methods

CONSTRUCTOR (one or more constructors as needed)
public boolean isEmpty( )
public boolean search(parameters here)
private boolean search(parameters here)
public FSCstudent findNode(parameters here)
private FSCstudent findNode(parameters here)
public void printMembers(parameters here)
private void printMembers (parameters here)
public void moveDevices(parameters here)
private void moveDevices (parameters here)
public void insert(parameters here)
private FSCstudent insert(parameters here)
public void delete(parameters here)
private FSCstudent delete(parameters here)
private FSCstudent minNode(parameters here)
private FSCstudent parent(parameters here)
private boolean isLeaf(parameters here)
private boolean hasOnlyLeftChild(parameters here)
private boolean hasOnlyRightChild(parameters here)

DETAILS:

- This class is for the BST
- All variables and methods should be clear
- The come directly from the sample code
- Of course, we do not use (or need) some of the methods from the sample code.
- Only additions are the `printMembers()` methods and the `moveDevices()` methods.

FSCscDevice

Data Members

```
private int MACnumber
private FSCscDevice next
```

Operations/Methods

CONSTRUCTOR (one or more constructors as needed)
public int getMACnumber( )
public FSCscDevice getNext( )
public void setMACnumber(int MACnumber)
public void setNext(FSCscDevice next)

DETAILS:

- This is the class for nodes of the linked list (FSCscLinkedDevices)
- All variables and methods should be clear
- They come directly from the sample code

FSCscLinkedDevices

Data Members

```
private FSCscDevice head
```

Operations/Methods

CONSTRUCTOR (one or more constructors as needed)
public boolean isEmpty( )
public boolean search(parameters here)
private boolean search(parameters here)
public void findNode(parameters here)
private FSCscDevice findNode(parameters here)
public void insert(parameters here)
private FSCscDevice insert(parameters here)
public void delete(parameters here)
private FSCscDevice delete(parameters here)
public void showConnections(parameters here)
private void showConnections (parameters here)
private boolean withinRange(parameters here)
public void removeAllLinks(parameters here)
private void removeAllLinks(parameters here)

DETAILS:

- This class is for the linked list
- EACH FSCstudent will have a linked-list of devices that are allowed to share their internet
- Therefore, INSIDE the FSCstudent, you will have a reference to an object of type FSCscLinkedDevices
- All variables and methods should be clear
- The come directly from the sample code
- Note: we do not need (or use) all the linked list methods (such as modify), because we do not need them for our program.
- NOTE: we have THREE extra methods as follows
- showConnections(): this method is used for the SHOWCONNECTIONS command. This method is basically a traversal of the linked list of devices. However, you must traverse one time, keep a count of devices “in range”, and then, if the count is greater than zero, you traverse again to print those devices.
- withinRange(): this method is called from the showConnections() private method. Here we use the Euclidian formula to calculate the distance between two students and see if that distance is less than the broadcast range
- removeAllLinks(): this method is called during the QUIT command. Before a user is deleted from the BST, that user’s MAC address must be removed from the list of linked devices for EACH student that it is linked to.