HOMEWORK - SPRING 2018

HOMEWORK 5 - due Tuesday, April 3rd no later than 7:00pm

REMINDERS:

- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty. *Remember, all work you submit for homework assignments MUST be entirely your own work. Also, group efforts are not allowed.***
- **Login to your grading account and click "Submit Assignment" to upload and submit your assignment.**
- **You are not allowed to use ArrayList, Vector or any other Java API Data Structure classes to implement this assignment except where noted.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**

The friendly Mountain Dew and fedora enthusiasts of the Stony Brook Basement Club have decided to set up a massively multiplayer Nintendo 64 Donkey Kong league. Because Wolfie Net is not particularly friendly to Nintendo 64s and their associated hardware (in this case consisting of raspberry pis with suspectly high number of wires sticking out), they have worked tirelessly over days and nights laying down recycled headphone wire between the various quads on campus, but they have realized that they need to create a network management software that will help them keep track of all the devices on their network (as they expect to have many network failures and partial failures, especially whenever the lawn mower people run over their wires or careless students spill drinks on their routers). They will model their network as a tree with two types of nodes: Nintendo Nodes (these are always leaves, and represent one Donkey Kong player with their Nintendo64 device), and Raspberry Nodes (these represent Raspberry Pis, which are used as routers in our network). They want to be able to view their network topology at any time, load and save it from a file (so they can occasionally restart their computer), add nodes, remove subtrees (using the same cut-paste system as in HW 2 - so they can re-route their network in case a raspberry router goes down. Finally, sometimes routers and Nintendo 64s go down, so each device will have a boolean for broken. The final function will be to

identify the root of the minimal subtree that contains all the failures for a given scenario (this will be extra credit). When we load the file, we will assume that all nodes are working, and no node can contain more than 9 children. Each line will contain a position string, as well as a node name. If they are separated by a dash, the node is a Nintendo. The tree will be given and should be saved in preorder.

Note: In this homework you may not use ANY java API data structure classes (this includes ArrayList, LinkedList, Vector,and any of the Tree classes). You must build your own tree from the nodes (like we did in the LinkedList homework).

Here is a sample text file (format: position in tree, dash if nintendo, text).:

```
SBU
1Central
11Mendy
111-Irving
112-Grey
113-LDS
2Hill
21Tabler
211-Tosc
212-TAC
213-Hand
22Roth
3West
31WestApartments
32-GLS Center
4-BasementClub
```

NOTE: All exceptions explicitly thrown in Required Classes except for IllegalArgumentException are custom exceptions that need to be made by you.

Required Classes

NetworkNode

Write a fully documented class called NetworkNode which holds the type of component being represented, an array of children (null if this will be a Nintendo), and string for the text. Be sure to include all getters and setters. You may find it helpful to write helper methods, especially to check input before adding a child to the tree. Defining custom exceptions for actions like trying to add too many children to a node, or adding a child in an invalid position may also be desirable.

- private String name
- private boolean isNintendo
- private boolean isBroken //default is false
- private NetworkNode parent
- private NetworkNode[] children //Just like in hw 1, there should be no holes in the array.
- final int maxChildren=9 //A full node exception may be desirable.

NetworkTree

Write a fully documented class called NetworkTree which will serve as the tree manager for your NetworkTree. This must hold references into a tree (the root and cursor), as well as be able to generate and save the tree to and from a file. Again, defining some custom exceptions may be helpful for code readability.

- private NetworkNode root
- private NetworkNode cursor
- public void cursorToRoot()
  - Sets the cursor to the root of the NetworkTree
- public NetworkNode cutCursor()

        - Removes the child at the specified index of the NetworkTree, **as well as all of its children.**
        - Cursor goes to parent
        - Cutting root clears the tree
    - public void addChild(int index, NetworkNode node)
        - Adds the given node to the corresponding index of the children array.
        - Should throw an **exception** if adding the node at the specified index makes a hole in the array.
    - public void cursorToChild(int index)
        - Moves the cursor to the child node of the of the cursor corresponding to the specified index.
    - public void cursorToParent()
        - Moves the cursor to the parent of the current node.
    - public static NetworkTree readFromFile(String filename)
        - Generates the NetworkTree based on the file name that is passed in.
    - public static void writeToFile(NetworkTree tree, String filename)
        - Generates a text file that reflects the structure of the NetworkTree.
        - The format of the tree of the file should match the format of the input file.
    - public void cursorToMinimalBrokenSubtree()//Extra credit
        - The cursor should be on a node where all of the broken nodes are either the cursor or descendants.
        - There may not be such a node in the subtree.
            - Accompanying method: public void changeCursorBrokenStatus()

NintendoNetwork
Write a fully-documented class named NintendoNetwork that takes in a text file, generates a NetworkTree and provides an interface for a user to manipulate the tree. Please see the UI required functions for the required functionality

- public static void main(String[] args)
    - The main method runs a menu driven application which first creates a NetworkTree based on the passed in file and then prompts the user for a menu command selecting the operation. The required information is then requested from the user based on the selected operation. You can find the list of menu options in the UI required functions section.
- NetworkTree tree;

Note on File I/O:
It is possible to read from a file using Scanner, which you should already be familiar with. Here's an example: https://stackoverflow.com/questions/13185727/reading-a-txt-file-using-scanner-class-in-java. To write to a file, you may use printwriter, with an example here: https://stackoverflow.com/questions/25298582/how-to-use-printwriter-to-create-and-write-to-a-file. Use nextLine() to read the next line of the file. Use standard string manipulation techniques like charAt to get the index of the first non-numeric character.

Note on Exceptions: all exceptions should be handled gracefully - they should be caught in the main, and the user should be notified by a nice printout. Your messages should clearly indicate what the problem is (bad index, full list, negative number, etc.). The program should continue to run normally after an exception is encountered. We will not be checking Input Mismatch cases.

Pretty Printing: Here is a tutorial on how to print tables neatly using printf in java. It is highly encouraged that you print the output neatly, as it makes grading much

easier. https://docs.oracle.com/javase/tutorial/java/data/numberformat.html

---

General Recommendations

You can feel free to add extra methods and variables if you need.

UI Required Functions

- L- Load from file
- P- Print
- C- Cursor to child (index number)
- A- Add child (index, type, prompt for text)
- U- Cursor up (to parent)
- X- Cut/Delete cursor
- V- Paste Subtree (index number)
- R- Cursor to root
- S- Save to Text File
- M-Cursor to root of minimal subtree containing all faults [extra credit]
- B-Mark cursor as broken/fixed (flip the state) [extra credit, required for M]
- Q - Quit

Note: please make sure that the menu is NOT case sensitive (so selecting A should be the same as selecting a).

Program Sample

```
Welcome to the Nintendo Network Manager.

Menu:
    L) Load from file
    P) Print tree
    C) Move cursor to a child node
    R) Move cursor to root
    U) Move cursor up to parent
    A) Add a child //if tree is empty, do not prompt for child index
number
    X) Remove/Cut Cursor and its subtree
    V) Paste Cursor and its subtree
    S) Save to file
    M) Cursor to root of minimal subtree containing all faults
    B) Mark cursor as broken/fixed
    Q) Quit

Please select an option: L
Please enter filename: ashrubbery.txt
ashrubbery.txt not found.

Please select an option: L
Please enter filename: sbutopology.txt
sbutopology.txt loaded

Please select an option: p

->SBU //cursor is automatically at root
     +Central
         +Mendy
             -Irving
             -Grey
             -LDS
    +Hill
        +Tabler
```

```
            -Tosc
            -TAC
            -Hand
        +Roth
    +West
        +WestApartments
        -GLS Center
    -BasementClub

Please select an option: c
Please enter an index: 2
Cursor moved to Hill.

Please select an option: c
Please enter an index: 2
Cursor moved to Roth.

Please select an option: a
Please enter an index: 1
Please enter device name: Hendrix
Is this Nintendo (y/n): Y
Nintendo added.

Please select an option: p

+SBU
    +Central
         +Mendy
             -Irving
             -Grey
             -LDS
    +Hill
        +Tabler
            -Tosc
            -TAC
            -Hand
        +Roth
            ->Hendrix
    +West
        +WestApartments
        -GLS Center
    -BasementClub

Please select an option: u
Cursor moved to Roth.

Please select an option: X
Roth cut, cursor is at Hill.

Please select an option: R
Cursor moved to SBU.

Please select an option: c
Please enter an index: 1
Cursor moved to Central.

Please select an option: v
Please enter an index: 1
Roth pasted as child of Central.

Please select an option: p

+SBU
    +Central
        ->Roth
```

```
            -Hendrix
         +Mendy
             -Irving
             -Grey
             -LDS
    +Hill
        +Tabler
            -Tosc
            -TAC
            -Hand

    +West
        +WestApartments
        -GLS Center
    -BasementClub


Please select an option: S
Please enter a filename: theGivingTree.txt // A good way to test your
program is to load the saved file
File saved.

Please select an option: Q
Make like a tree and leave!

//Extra Credit Example

Please select an option: p

+SBU
    +Central
        ->Roth
            -Hendrix
         +Mendy
             -Irving
             -Grey
             -LDS
    +Hill
        +Tabler
            -Tosc
            -TAC
            -Hand

    +West
        +WestApartments
        -GLS Center
    -BasementClub

Please select an option: b
Roth marked as broken.

//after a few devices have been marked as broken:

Please select an option: p

+SBU
    +Central
        ->Roth ~Fault~
            -Hendrix
         +Mendy
             -Irving ~Fault~
             -Grey
             -LDS  ~Fault~
    +Hill
        +Tabler
```

```
            -Tosc
            -TAC
            -Hand

    +West
        +WestApartments
        -GLS Center
    -BasementClub

Please select an option: M
Cursor moved to Central.

//after a few more devices have been marked as broken:

Please select an option: p

+SBU
    +Central
        ->Roth ~Fault~
            -Hendrix
         +Mendy
             -Irving ~Fault~
             -Grey
             -LDS  ~Fault~
    +Hill
        +Tabler
            -Tosc ~Fault~
            -TAC
            -Hand

    +West ~Fault~
        +WestApartments
        -GLS Center
    -BasementClub

Please select an option: M
Cursor moved to SBU.
```

Extra Credit

You will get up to 5 points extra credit for creating a JavaFX GUI, and up to 8 points for creating an Android App, 12 points for JSP/HTML/CSS. Please discuss with your Grading TA if you choose to make an app or use Scene Builder, so you can coordinate submitting supplementary files with them. Identifying the minimal broken subtree correctly will add up to 5 more points.

Course Info | Schedule | Sections | Announcements | Homework | Exams | Help/FAQ | Grades | HOME