CSE222

HomeWork #5

Report

1901042676

Mustafa Mercan

Part A :

For this part of the assignment, we were asked to read the data from a .txt file and then create a tree structure based on the parsed data. Since the data in the .txt file is separated by ";" characters, I needed to parse it appropriately and store it in an ArrayList.
Here are the steps I followed:

1. First, I created a rootNode for our tree. This node will be our root node.
2. I created the necessary structures for our tree to be displayed on the screen.
3. Then, I read the data from the txt file line by line, parsed it in the appropriate format, and added it to the ArrayList I created.
4. After making the necessary additions to the ArrayList, I created the necessary nodes by iterating through the ArrayList and established the connections between them, then added them to the tree.
5. Finally, I printed the created structure to the screen.

Part B (BFS) :

In this part of the assignment, we were asked to implement a search method in the Tree we created in Part A. This search method will use the BFS algorithm to perform the search. In the BFS algorithm, nodes on the same level are visited first, and then the search continues with the child nodes.
Here are the steps I followed:

1. First, I created a queue structure to be able to use the BFS algorithm.
2. I added our rootNode to the queue structure first because the search operation will start from here.
3. I defined a variable named "step" to keep track of which step we are at while printing the nodes to the screen.
4. I created a loop to ensure that the search process does not end before all nodes in the queue have been traversed.
5. If the node in the queue is the one we are looking for, we print the information to the screen and end the search. If we have not yet reached the node we are looking for, we continue our search by adding the child nodes of the current node to our queue structure.
6. If our loop ends and we still haven't found the node we are looking for, we print an information message to the screen and end the search process.

Part C (DFS) :

In this part of the assignment, we are asked to implement the same method we created in Part B, but using DFS algorithm. In DFS algorithm, the nodes are visited until the lowest child node is reached before visiting the other nodes.

1. First, we need a stack structure to be able to use DFS algorithm. I created this structure.
2. I first added our rootNode to the stack structure because the search process will start here.
3. I defined a variable named int step. This variable will hold the information of which step we are on when printing our nodes to the screen.
4. I created a loop so that all nodes in the stack structure are traversed before the process ends.
5. If the node in the stack is the Node we are looking for, we print the information to the screen and the process ends. If we have not yet reached our desired node, we add the child nodes of the current node to our stack structure and continue our process.
6. If our loop ends, meaning that all nodes have been visited and we could not find the node we are looking for, we print a message to the screen and end our process.

Part D (Post Order Traversal) :

In this part of the assignment, we were asked to create a method similar to the one we created in Part B, but using the Post Order Traversal algorithm. In the Post Order Traversal algorithm, we traverse to the bottom nodes of the tree and then move up to visit other nodes.

1. Since we need to use a recursive function to apply the Post Traversal algorithm, I first arranged the base case condition.
2. Then I called our function again for all child nodes in the current node.
3. Afterwards, I checked whether the current node is the node we are looking for.
4. Throughout all these steps, I printed the information of our node to the screen and sent true or false values as necessary.
5. Then, I defined an integer variable "step" within the class to print the steps.
6. I applied overriding to this method. The reason for this is to reset the steps each time and to print the message that we could not find the node we were looking for if our method returns false.

Part E ：

In the final part of the assignment, we were asked to move any of the problem/ lecture/course nodes in our created tree to a different year than the one they currently exist in. If the path we want to move does not exist, we need to display an error message.

1. First, I created two queue structures. One is used to hold the necessary parsed strings in the "input→input→input" format entered by the user. The other queue is created to hold the nodes that contain these strings.
2. Added the necessary parse operations to the queue.
3. If the year we want to copy the file to does not exist, I created a new node and added it as the child of our rootNode.
4. After identifying the node we want to move, I deleted it from its parent.
5. I then checked whether the nodes in the queue containing our nodes were the same as the path we wanted to move. If such a path does not exist, a new path is created. If such a path exists, no action is taken.
6. After processing all the nodes in the queue, when only the last node is left, we add the node we want to move to the desired location. If another node with the same name as this node exists, it is replaced by the node we want to move.
7. If there are no child nodes in the year and course information, this node is removed from the parent node.
8. Finally, after completing all the changes, I update the visual representation of our tree and display the updated version on the screen.

How do you run this assignment?

1. javac Main.java
2. java Main

→ After compiling and running the program, a tree will be displayed on the screen according to the data in the txt file, and then the user will be prompted to enter certain inputs.

→For part (B-C-D), one input is expected from the user.

→For part E, two inputs are expected from the user.
input = The path of the node to be moved. The format should be as follows: command1→command2→command3→command4
input = The year to which we want to move the node.