Class : PhaseOne, java

CSC212 Data structures – Project phases I

Spring 2023

DATE : 20-05-2023

TEAM :

442103210

AUTHORS :

Moath Ali Almoghyirah 441101342

Abdulmohesen sultan alqahtani 442103210

Nawaf mohammed Alhomaidhi 442101288

This method return true if the path is valid

```
private boolean follow MNode<T> t, String path) {
        // Check if the current node is null
        if (t == null) {
            return false;
        }
        // Compare last node
        if (path.length() == 1) {
            return compare(t, path.charAt(0));
        }
        // Cleaning the given string
        if (path.contains("-"))
            path = path.replace("-", "");
        path = path.toUpperCase();
        System.out.println(path);

        char currentDirection = path.charAt(0);
        char nextDirection = path.charAt(1);

        if (compare(t, currentDirection)) {// Compare the
current node to current direction path(index:0)
            if (t.left != null && compare(t.left,
nextDirection)) {// If left child data equals next
path(index:1)
                return follow(t.left, path.substring(1));
            } else if (t.right != null &&
compare(t.right, nextDirection)) {// If right child data
equals next path(index:1)
                return follow(t.right,
path.substring(1));
            }
        }
        return false;
    }
```

```
Return true if it found child

private boolean escape MNode <T> t)
    {
        if (t == null) // stop condition and if there is
no child
            return false;

        if (t.data.equals(Character1.X ) // when exit
found
            return true;

        return escape(t.left) || escape(t.right); // to
move to the next level
    }
```

```
(this method return the shorter path)
private String short1(MNode<T> node) {
        if (node == null) {
            return "";
        }
        // Check if the current node is an exit
        if (node.data.equals(Character1.X)) {
            return node.data.toString();
        }

        // Recursively get the shortest path
        String leftPath = short1(node.left);
        String rightPath = short1(node.right);

        // Determine the shortest path
        if (leftPath.isEmpty() && rightPath.isEmpty()) {
            // No exit found
            return "";
        } else if (leftPath.isEmpty()) {
            // Exit found in the right child
            return node.data + "-" + rightPath;
        } else if (rightPath.isEmpty()) {
            // Exit found in the left child
            return node.data + "-" + leftPath;
        } else {
            // Exit found in both, return the shortest
path
```

```
            return node.data + "-" + (leftPath.length() <
rightPath.length() ? leftPath : rightPath);

        }
    }
```