```java
package graph;

import java.util.LinkedList;

class DFS {

    public static void DFSUsingAdjacentMatrix(AdjacencyMatrix adjacencyMatrix) {

        Integer[][] innerAdjacencyMatrix = adjacencyMatrix.getInnerAdjacencyMatrix();

        boolean[] hasVisited = new boolean[innerAdjacencyMatrix.length];

        for (int i = 1; i < hasVisited.length; i++) {
            DFSUsingAdjacentMatrixHelper(i, innerAdjacencyMatrix, hasVisited, adjacencyMatrix);
        }
    }

    private static void DFSUsingAdjacentMatrixHelper(int currentVertexIndex,
                                                     Integer[][] innerAdjacencyMatrix,
                                                     boolean[] hasVisited,
                                                     AdjacencyMatrix adjacencyMatrix) {

        if (hasVisited[currentVertexIndex]) {
            return;
        }

        hasVisited[currentVertexIndex] = true;
        System.out.println(adjacencyMatrix.getVertexName(currentVertexIndex));


        for (int j = 0; j < innerAdjacencyMatrix[currentVertexIndex].length; j++) {

            if (innerAdjacencyMatrix[currentVertexIndex][j] != null
                    && !hasVisited[j]) {

                DFSUsingAdjacentMatrixHelper(j, innerAdjacencyMatrix, hasVisited, adjacencyMatrix);
            }
        }
    }


    public static void DFSUsingAdjacentTable(AdjacencyTable adjacencyTable) {

        LinkedList<LinkedList<AdjacencyTable.Node>> innerAdjacencyTable = adjacencyTable.getInnerAdjacencyTable();
        boolean[] hasVisited = new boolean[innerAdjacencyTable.size()];

        for (int i = 1; i < innerAdjacencyTable.size(); i++) {

            DFSUsingAdjacentTableHelper(i, innerAdjacencyTable, hasVisited, adjacencyTable);
        }
    }


    private static void DFSUsingAdjacentTableHelper(int currentVertexIndex,
                                                    LinkedList<LinkedList<AdjacencyTable.Node>> innerAdjacencyTable,
                                                    boolean[] hasVisited,
                                                    AdjacencyTable adjacencyTable) {

        if (hasVisited[currentVertexIndex]) {
            return;
        }

        hasVisited[currentVertexIndex] = true;
        System.out.println(innerAdjacencyTable.get(currentVertexIndex).get(0).name);

        for (int i = 1; i < innerAdjacencyTable.get(currentVertexIndex).size(); i++) {

            DFSUsingAdjacentTableHelper(
                    adjacencyTable.getIndex(
                            innerAdjacencyTable.get(currentVertexIndex).get(i).name),
                    innerAdjacencyTable, hasVisited, adjacencyTable);
        }
    }


    public static void main(String[] args) {

/*         String[] vertex = {"1", "2", "3", "4", "5", "6", "7"};

        AdjacencyMatrix adjacencyMatrix = new AdjacencyMatrix(vertex);
        adjacencyMatrix.updateEdge("1", "2", 1);
        adjacencyMatrix.updateEdge("3", "1", 6);
        adjacencyMatrix.updateEdge("4", "1", 3);
        adjacencyMatrix.updateEdge("2", "3", 4);
        adjacencyMatrix.updateEdge("2", "4", 4);
        adjacencyMatrix.updateEdge("4", "3", 9);
        adjacencyMatrix.updateEdge("6", "2", 7);
        adjacencyMatrix.updateEdge("7", "6", 7);
        adjacencyMatrix.updateEdge("7", "4", 7);
        adjacencyMatrix.updateEdge("5", "7", 7);
        adjacencyMatrix.updateEdge("5", "6", 7);

        DFSUsingAdjacentMatrix(adjacencyMatrix); */


        String[] vertex = {"1", "2", "3", "4", "5", "6", "7"};
        AdjacencyTable adjacencyTable = new AdjacencyTable(vertex);

        adjacencyTable.updateEdge("1", "2", 1);
        adjacencyTable.updateEdge("3", "1", 6);
        adjacencyTable.updateEdge("4", "1", 3);
        adjacencyTable.updateEdge("2", "3", 4);
        adjacencyTable.updateEdge("2", "4", 4);
        adjacencyTable.updateEdge("4", "3", 9);
        adjacencyTable.updateEdge("6", "2", 7);
        adjacencyTable.updateEdge("7", "6", 7);
        adjacencyTable.updateEdge("7", "4", 7);
        adjacencyTable.updateEdge("5", "7", 7);
        adjacencyTable.updateEdge("5", "6", 7);

        DFSUsingAdjacentTable(adjacencyTable);
    }
}
```