```
package datastructure.Graph;

import java.io.*;

import java.util.*;

public class GraphTraversals {
    private int vertexCount;
    private LinkedList<Integer> adjListArray[];

    GraphTraversals(int v) {
        this.vertexCount = v;
        adjListArray = new LinkedList[v];
        for (int i = 0; i < v; ++i)
            adjListArray[i] = new LinkedList();
    }

    void addEdge(int source, int destination) {
        adjListArray[source].add(destination);
    }

    void DFSUtil(int v, boolean[] visited) {
        visited[v] = true;
        System.out.print(v + " ");

        Iterator<Integer> i = adjListArray[v].listIterator();
        while (i.hasNext()) {
            int n = i.next();
```

```
        if (!visited[n])
            DFSUtil(n, visited);
    }
}

void DFS(int v) {
    boolean visited[] = new boolean[vertexCount];
    DFSUtil(v, visited);
}

void BFS(int source) {
    boolean visited[] = new boolean[vertexCount];
    LinkedList<Integer> queue = new LinkedList<Integer>();
    visited[source] = true;
    queue.add(source);
    while (queue.size() != 0) {
        source = queue.poll();
        System.out.print(source + " ");
        Iterator<Integer> i = adjListArray[source].listIterator();
        while (i.hasNext()) {
            int n = i.next();
            if (!visited[n]) {
                visited[n] = true;
                queue.add(n);
            }
        }
    }
}
```

```
    public static void main(String[] args) {
        GraphTraversals graph = new GraphTraversals(4);

        graph.addEdge(0, 1);
        graph.addEdge(0, 2);
        graph.addEdge(1, 2);
        graph.addEdge(2, 0);
        graph.addEdge(2, 3);

        System.out.println("Breadth First Traversal ");
        graph.BFS(0);

        System.out.println("Depth First Traversal ");
        graph.DFS(0);
    }

}
```