Code

```
import java.util.*;

class Graph {
    private int V; // Number of vertices
    private LinkedList<Integer> adj[]; // Adjacency List

    // Constructor
    Graph(int v) {
        V = v;
        adj = new LinkedList[v];
        for (int i = 0; i < v; ++i)
            adj[i] = new LinkedList();
    }

    // Function to add an edge into the graph
    void addEdge(int v, int w) {
        adj[v].add(w);
    }

    // BFS traversal from a given source
    void BFS(int s) {
        boolean visited[] = new boolean[V];
        LinkedList<Integer> queue = new LinkedList<Integer>();
        visited[s] = true;
        queue.add(s);
```

```
    while (queue.size() != 0) {
      s = queue.poll();
      System.out.print(s + " ");

      Iterator<Integer> i = adj[s].listIterator();
      while (i.hasNext()) {
        int n = i.next();
        if (!visited[n]) {
          visited[n] = true;
          queue.add(n);
        }
      }
    }
}

// DFS traversal from a given source
void DFSUtil(int v, boolean visited[]) {
    visited[v] = true;
    System.out.print(v + " ");

    Iterator<Integer> i = adj[v].listIterator();
    while (i.hasNext()) {
      int n = i.next();
      if (!visited[n])
        DFSUtil(n, visited);
    }
}

void DFS(int v) {
```

```
        boolean visited[] = new boolean[V];
        DFSUtil(v, visited);
    }
}

public class Main {
    public static void main(String args[]) {
        Graph g = new Graph(4);

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

        System.out.println("BFS starting from vertex 2:");
        g.BFS(2);
        System.out.println("\nDFS starting from vertex 2:");
        g.DFS(2);
    }
}
```