```java
// Java program to print BFS traversal from a given source vertex.
// BFS(int s) traverses vertices reachable from s.
import java.io.*;
import java.util.*;

// This class represents a directed graph using adjacency list
// representation
class BFSmain
{
    private int V; // No. of vertices
    private LinkedList<Integer> adj[]; //Adjacency Lists

    // Constructor
    BFSmain(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);
    }

    // prints BFS traversal from a given source s
    void BFS(int s)
    {
        // Mark all the vertices as not visited(By default
        // set as false)
        boolean visited[] = new boolean[V];

        // Create a queue for BFS
        LinkedList<Integer> queue = new LinkedList<Integer>();

        // Mark the current node as visited and enqueue it
        visited[s]=true;
```

```
40          queue.add(s);
41
42          while (queue.size() != 0)
43          {
44              // Dequeue a vertex from queue and print
    it
45              s = queue.poll();
46              System.out.print(s+" ");
47
48              // Get all adjacent vertices of the
    dequeued vertex s
49              // If a adjacent has not been visited,
    then mark it
50              // visited and enqueue it
51              Iterator<Integer> i = adj[s].listIterator
    ();
52              while (i.hasNext())
53              {
54                  int n = i.next();
55                  if (!visited[n])
56                  {
57                      visited[n] = true;
58                      queue.add(n);
59                  }
60              }
61          }
62      }
63
64      // Driver method to
65      public static void main(String args[])
66      {
67          BFSmain g = new BFSmain(4);
68
69          g.addEdge(0, 1);
70          g.addEdge(0, 2);
71          g.addEdge(1, 2);
72          g.addEdge(2, 0);
73          g.addEdge(2, 3);
74          g.addEdge(3, 3);
75
76          System.out.println("Following is Breadth
    First Traversal "+ "(starting from vertex 2)");
77
78          g.BFS(2);
79      }
```

```
80 }
81 // This code is contributed by Aakash Hasija
82
```