

Graph.java

```
1  package com.hongchuan.app;
2  // Java program to print BFS traversal from a given source vertex.
3  // BFS(int s) traverses vertices reachable from s.
4  import java.io.*;
5  import java.util.*;
6
7  // This class represents a directed graph using adjacency list
8  // representation
9  public class Graph
10 {
11     private int V; // No. of vertices
12     private LinkedList<Integer> adj[]; //Adjacency Lists
13     ArrayList<Integer> trace = new ArrayList<Integer>(0);
14     // Constructor
15     public Graph(int v)
16     {
17         V = v;
18         adj = new LinkedList[V];
19         for (int i=0; i<V; ++i)
20             adj[i] = new LinkedList();
21     }
22     public ArrayList<Integer> getTrace(){
23         return trace;
24     }
25     // Function to add an edge into the graph
26     public void addEdge(int v,int w)
27     {
28         adj[v].add(w);
29     }
30
31     // prints BFS traversal from a given source s
32     public void BFS(int s)
33     {
34         // Mark all the vertices as not visited(By default
35         // set as false)
36         boolean visited[] = new boolean[V];
37
38         // Create a queue for BFS
39         LinkedList<Integer> queue = new LinkedList<Integer>();
40
41         // Mark the current node as visited and enqueue it
42         visited[s]=true;
43         queue.add(s);
44
45         while (queue.size() != 0)
46         {
47             // Dequeue a vertex from queue and print it
48             s = queue.poll();
49             // System.out.print(s+" ");
50             trace.add(s);
```

```

51
52         // Get all adjacent vertices of the dequeued vertex s
53         // If a adjacent has not been visited, then mark it
54         // visited and enqueue it
55         Iterator<Integer> i = adj[s].listIterator();
56 1         while (i.hasNext())
57         {
58             int n = i.next();
59 1             if (!visited[n])
60             {
61                 visited[n] = true;
62                 queue.add(n);
63             }
64         }
65     }
66 }
67
68 // // Driver method to
69 // public static void main(String args[])
70 // {
71 //     Graph g = new Graph(4);
72
73 //     g.addEdge(0, 1);
74 //     g.addEdge(0, 2);
75 //     g.addEdge(1, 2);
76 //     g.addEdge(2, 0);
77 //     g.addEdge(2, 3);
78 //     g.addEdge(3, 3);
79
80 //     System.out.println("Following is Breadth First Traversal "+
81 //                         "(starting from vertex 2)");
82
83 //     g.BFS(2);
84 // }
85 }
86 // This code is contributed by Aakash Hasija

```

Mutations

- [19](#) 1. changed conditional boundary → KILLED
- 2. Changed increment from 1 to -1 → KILLED
- 3. negated conditional → KILLED
- [23](#) 1. mutated return of Object value for com/hongchuan/app/Graph::getTrace to (if (x != null) null else throw new RuntimeException) → KILLED
- [45](#) 1. negated conditional → KILLED
- [56](#) 1. negated conditional → KILLED
- [59](#) 1. negated conditional → KILLED

Active mutators

- INCREMENTS_MUTATOR
- VOID_METHOD_CALL_MUTATOR
- RETURN_VALS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR

- INVERT_NEGS_MUTATOR
- CONDITIONALS_BOUNDARY_MUTATOR

Tests examined

- com.hongchuan.app.GraphTest.testCase1(com.hongchuan.app.GraphTest) (6 ms)

Report generated by [PIT](#) 1.4.3