## Code:

```
package graphc;
import java.util.*;
import java.util.LinkedList;

public class BFSGraph{
 private int V;   // No. of vertices
 private LinkedList<Integer> adj[]; //Adjacency List

 public BFSGraph(int v)
 {
     V = v;
     adj = new LinkedList[v];
     for (int i=0; i<v; ++i)
         adj[i] = new LinkedList();
 }

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

 public Boolean hasConnection(int s, int d)
 {
     LinkedList<Integer>temp;

     // Marking vertices as not visited
     boolean visited[] = new boolean[V];

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

     // Mark current node visited
     visited[s]=true;
     queue.add(s);

     // 'i' used for adjacent vertices of a vertex
     Iterator<Integer> i;
     while (queue.size()!=0)
     {
         s = queue.poll();

         int n;
         i = adj[s].listIterator();

         // If a adjacent not visited, marking it visited
         while (i.hasNext())
         {
             n = i.next();

             // Return true if adjacent node is end node
             if (n==d)
                 return true;

             // Else, continue to do BFS
             if (!visited[n])
             {
                 visited[n] = true;
                 queue.add(n);
             }
         }
     }

     //BFS complete without visiting "d" then, Falsse
     return false;
 } }
```

## Reference:

https://codedump.io/share/q0VU2X946Bmw/1/depth-first-search-on-graph-using-iterator

## Junit Testing:

```
import graphc.BFSGraph;
import org.junit.Test;

public class Junit {

	@Test
	public void test1(){

		BFSGraph testcase1 = new BFSGraph(4);

	    testcase1.addEdge(0, 1);
	    testcase1.addEdge(0, 2);
	    testcase1.addEdge(1, 2);
	    testcase1.addEdge(2, 0);
	    testcase1.addEdge(2, 3);
	    testcase1.addEdge(3, 3);

		int u = 3;
		int v = 1;

	    if (testcase1.hasConnection(u, v))
	        System.out.println("Path connected");
	    else
	        System.out.println("Path not connected");;
	}

	// FAILED TEST CASE
	@Test
	public void test2() {

			int u = 1;
			int v = 3;

			BFSGraph testcase2 = new BFSGraph(1);

			if (testcase2.hasConnection(u, v))
			    System.out.println("Path connected");
			else
			    System.out.println("Path not connected");;
	}

}
```