BFS CODE IN JAVA

```
import java.util.*;
class Graph
{
private int V;
private LinkedList<Integer> adj[];
Graph(int v)
{
V = v;
adj = new LinkedList[v];
for(int i = 0; i<v; ++i)
    adj[i] = new LinkedList();
}
void addEdge(int v,int w)
{
adj[v].add(w);
}
void BFS(int s)
{
boolean visited[] = new boolean(V);
LinkedList<Integer> queue = LinkedList<Integer>();
visited[s] = true;
queue.add(s);
While(queue.size() != 0)
{
s = queue.poll();
System.out.println(s + " ");
Iterator<Integer> i = adj[s].listIterator();
while(i.hasnext())
{
int n = i.next();
if(!visited[n])
{
```

```
visited[n] = true;

queue.add(n);

}

}

}

}
```

Code Explanation :-

Sure, let me explain the code for BFS in Java step-by-step:

1. We first create a Graph class that contains the number of vertices V and an adjacency list adj. The Graph class has a constructor that initializes the adjacency list with an empty LinkedList for each vertex.

2. We add a method addEdge() to add edges between two vertices in the graph. The method adds an element w to the adjacency list of vertex v.

3. We implement the BFS() method to traverse the graph using breadth-first search algorithm. The method takes a starting vertex s as input.

4. Inside the BFS() method, we first initialize a boolean array visited that keeps track of which vertices have been visited. We also create a LinkedList called queue to store the vertices that need to be visited.

5. We mark the starting vertex as visited and enqueue it.

6. We implement the core BFS algorithm using a while loop. We dequeue a vertex from the queue, mark it as visited, and print it. We then enqueue all unvisited neighbors of the dequeued vertex, and repeat the process until the queue is empty.

7. Finally, we add a main() method to create a Graph object and call the 'BFS()' method to perform a BFS traversal of the graph starting from a specified vertex.