**Task 5: Breadth-First Search (BFS) Implementation**

**For a given undirected graph, implement BFS to traverse the graph starting from a given node and print each node in the order it is visited.**

**import** java.util.\*;

**public** **class** Graph {

**private** **final** Map<Integer, List<Integer>> adjacencyList;

**public** Graph() {

**this**.adjacencyList = **new** HashMap<>();

}

// Method to add a node to the graph

**public** **void** addNode(**int** node) {

adjacencyList.putIfAbsent(node, **new** ArrayList<>());

}

// Method to add an edge to the graph (undirected)

**public** **void** addEdge(**int** from, **int** to) {

adjacencyList.putIfAbsent(from, **new** ArrayList<>());

adjacencyList.putIfAbsent(to, **new** ArrayList<>());

adjacencyList.get(from).add(to);

adjacencyList.get(to).add(from);

}

// BFS method to traverse the graph from a starting node

**public** **void** bfs(**int** start) {

Set<Integer> visited = **new** HashSet<>();

Queue<Integer> queue = **new** LinkedList<>();

visited.add(start);

queue.add(start);

**while** (!queue.isEmpty()) {

**int** node = queue.poll();

System.***out***.print(node + " ");

**for** (**int** neighbor : adjacencyList.get(node)) {

**if** (!visited.contains(neighbor)) {

visited.add(neighbor);

queue.add(neighbor);

}

}

}

}

**public** **static** **void** main(String[] args) {

Graph graph = **new** Graph();

// Adding nodes

graph.addNode(1);

graph.addNode(2);

graph.addNode(3);

graph.addNode(4);

graph.addNode(5);

// Adding edges

graph.addEdge(1, 2);

graph.addEdge(1, 3);

graph.addEdge(2, 4);

graph.addEdge(3, 5);

// Performing BFS from node 1

System.***out***.print("BFS starting from node 1: ");

graph.bfs(1);

System.***out***.println();

}

}