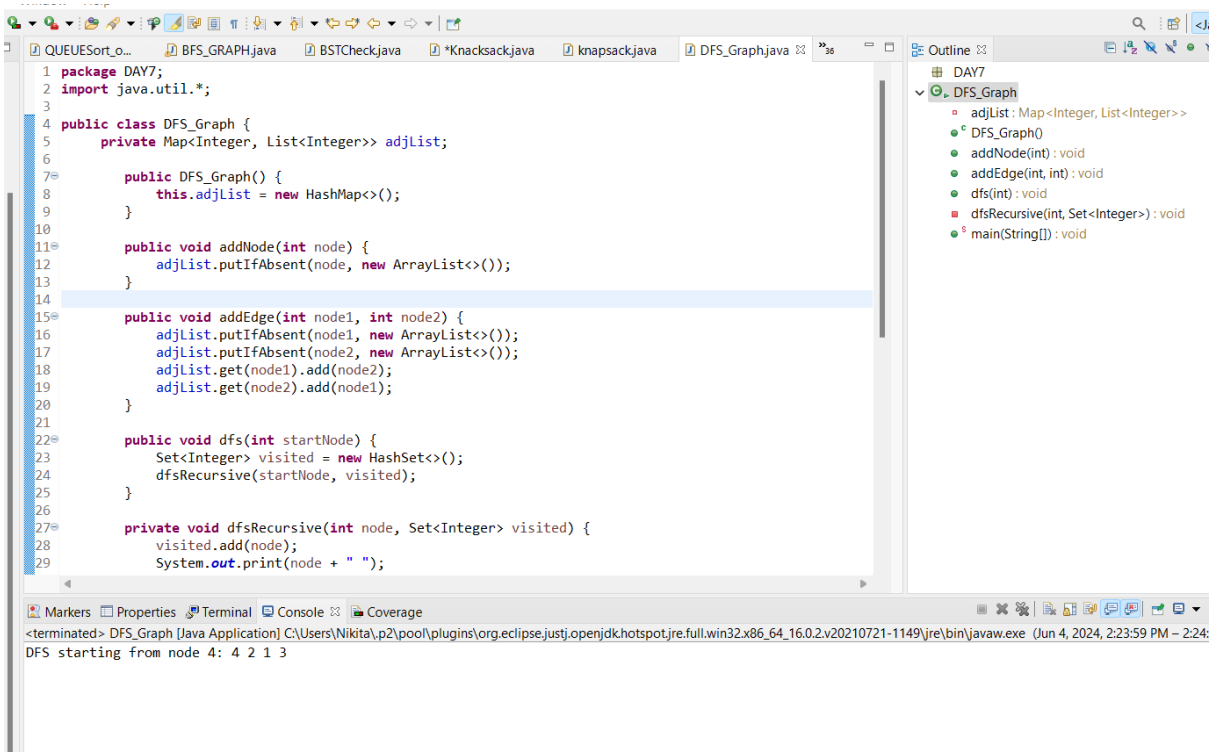
DFS GRAPH

```java
import java.util.*;

public class DFS_Graph {
        private Map<Integer, List<Integer>> adjList;

            public DFS_Graph() {
                this.adjList = new HashMap<>();
            }

            public void addNode(int node) {
                adjList.putIfAbsent(node, new ArrayList<>());
            }

            public void addEdge(int node1, int node2) {
                adjList.putIfAbsent(node1, new ArrayList<>());
                adjList.putIfAbsent(node2, new ArrayList<>());
                adjList.get(node1).add(node2);
                adjList.get(node2).add(node1);
            }

            public void dfs(int startNode) {
                Set<Integer> visited = new HashSet<>();
                dfsRecursive(startNode, visited);
            }

            private void dfsRecursive(int node, Set<Integer> visited) {
                visited.add(node);
                System.out.print(node + " ");

                for (int neighbor : adjList.get(node)) {
                    if (!visited.contains(neighbor)) {
                        dfsRecursive(neighbor, visited);
                    }
                }
            }

            public static void main(String[] args) {
                DFS_Graph graph = new DFS_Graph();
                graph.addNode(1);
                graph.addNode(2);
                graph.addNode(3);
                graph.addNode(4);

                graph.addEdge(1, 2);
                graph.addEdge(1, 3);
                graph.addEdge(2, 4);
                graph.addEdge(3, 4);

                System.out.print("DFS starting from node 4: ");
                graph.dfs(4);
            }
        }
```




```java
package DAY7;
import java.util.*;

public class DFS_Graph {
    private Map<Integer, List<Integer>> adjList;

        public DFS_Graph() {
            this.adjList = new HashMap<>();
        }

        public void addNode(int node) {
            adjList.putIfAbsent(node, new ArrayList<>());
        }

        public void addEdge(int node1, int node2) {
            adjList.putIfAbsent(node1, new ArrayList<>());
            adjList.putIfAbsent(node2, new ArrayList<>());
            adjList.get(node1).add(node2);
            adjList.get(node2).add(node1);
        }

        public void dfs(int startNode) {
            Set<Integer> visited = new HashSet<>();
            dfsRecursive(startNode, visited);
        }

        private void dfsRecursive(int node, Set<Integer> visited) {
            visited.add(node);
            System.out.print(node + " ");
```