

DFS GRAPH

```
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);
    }
}
```

```
1 package DAY7;
2 import java.util.*;
3
4 public class DFS_Graph {
5     private Map<Integer, List<Integer>> adjList;
6
7     public DFS_Graph() {
8         this.adjList = new HashMap<>();
9     }
10
11     public void addNode(int node) {
12         adjList.putIfAbsent(node, new ArrayList<>());
13     }
14
15     public void addEdge(int node1, int node2) {
16         adjList.putIfAbsent(node1, new ArrayList<>());
17         adjList.putIfAbsent(node2, new ArrayList<>());
18         adjList.get(node1).add(node2);
19         adjList.get(node2).add(node1);
20     }
21
22     public void dfs(int startNode) {
23         Set<Integer> visited = new HashSet<>();
24         dfsRecursive(startNode, visited);
25     }
26
27     private void dfsRecursive(int node, Set<Integer> visited) {
28         visited.add(node);
29         System.out.print(node + " ");
30     }
31 }
```

DAY7

- DFS_Graph
 - adjList : Map<Integer, List<Integer>>
 - DFS_Graph()
 - addNode(int) : void
 - addEdge(int, int) : void
 - dfs(int) : void
 - dfsRecursive(int, Set<Integer>) : void
 - main(String[]) : void

Markers Properties Terminal Console Coverage

<terminated> DFS_Graph [Java Application] C:\Users\Nikita\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (Jun 4, 2024, 2:23:59 PM - 2:24:00 PM)

DFS starting from node 4: 4 2 1 3