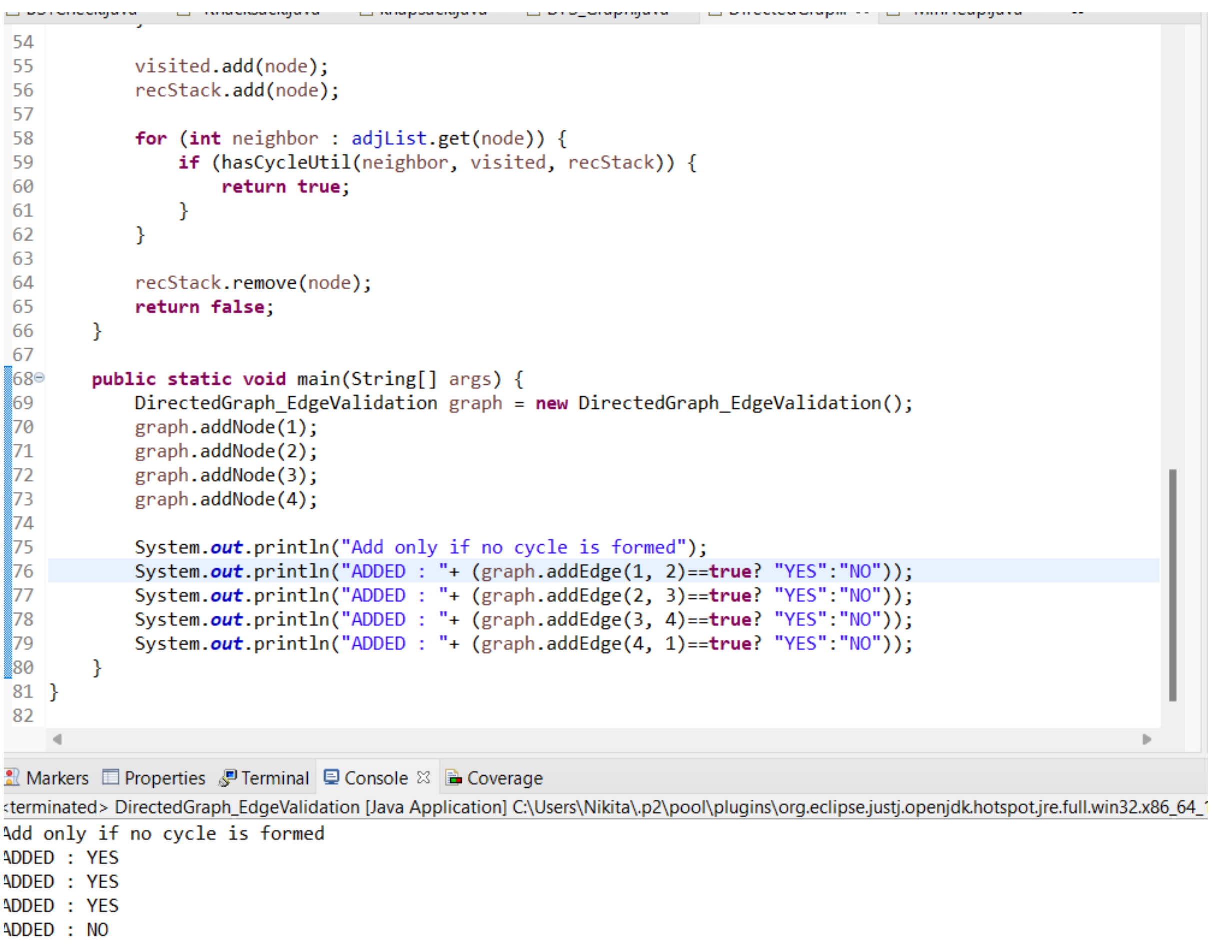
DirectedGraph_EdgeValidation

```java
import java.util.*;


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


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


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


    public boolean addEdge(int from, int to) {
        if (!adjList.containsKey(from) || !adjList.containsKey(to)) {
            throw new IllegalArgumentException("Node does not exist");
        }



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



        if (hasCycle()) {


            adjList.get(from).remove((Integer) to);
            return false;
        }
        return true;
    }

    private boolean hasCycle() {
        Set<Integer> visited = new HashSet<>();
        Set<Integer> recStack = new HashSet<>();

        for (int node : adjList.keySet()) {
            if (hasCycleUtil(node, visited, recStack)) {
                return true;
            }
        }

        return false;
    }

    private boolean hasCycleUtil(int node, Set<Integer> visited, Set<Integer> recStack) {
        if (recStack.contains(node)) {
            return true;
        }
        if (visited.contains(node)) {
            return false;
        }

        visited.add(node);
        recStack.add(node);

        for (int neighbor : adjList.get(node)) {
            if (hasCycleUtil(neighbor, visited, recStack)) {
                return true;
            }
        }

        recStack.remove(node);
        return false;
    }

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

        System.out.println("Add only if no cycle is formed");
        System.out.println("ADDED : "+ (graph.addEdge(1, 2)==true? "YES":"NO"));
        System.out.println("ADDED : "+ (graph.addEdge(2, 3)==true? "YES":"NO"));
        System.out.println("ADDED : "+ (graph.addEdge(3, 4)==true? "YES":"NO"));
        System.out.println("ADDED : "+ (graph.addEdge(4, 1)==true? "YES":"NO"));
    }
}
```



```java
        visited.add(node);
        recStack.add(node);

        for (int neighbor : adjList.get(node)) {
            if (hasCycleUtil(neighbor, visited, recStack)) {
                return true;
            }
        }

        recStack.remove(node);
        return false;
    }

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

        System.out.println("Add only if no cycle is formed");
        System.out.println("ADDED : "+ (graph.addEdge(1, 2)==true? "YES":"NO"));
        System.out.println("ADDED : "+ (graph.addEdge(2, 3)==true? "YES":"NO"));
        System.out.println("ADDED : "+ (graph.addEdge(3, 4)==true? "YES":"NO"));
        System.out.println("ADDED : "+ (graph.addEdge(4, 1)==true? "YES":"NO"));
    }
}
```

Markers Properties Terminal Console Coverage

```
<terminated> DirectedGraph_EdgeValidation [Java Application] C:\Users\Nikita\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_
Add only if no cycle is formed
ADDED : YES
ADDED : YES
ADDED : YES
ADDED : NO
```