Lista de adjacência:

```
package org.example;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class Graph {
    private HashMap<String, ArrayList<String>> adjList = new
HashMap<>();

    public boolean addVertex(String vertex){
        if(adjList.get(vertex) == null){
            adjList.put(vertex, new ArrayList<String>());

            return true;
        }

        return false;
    }

    public boolean addEdge(String vertex1, String vertex2){
        if(adjList.get(vertex1) != null && adjList.get(vertex2) !=
null){
            adjList.get(vertex1).add(vertex2);
            adjList.get(vertex2).add(vertex1);

            return true;
        }
        return false;
    }

    public boolean removeEdge(String vertex1, String vertex2){
        if(adjList.get(vertex1) != null && adjList.get(vertex2) !=
null){
            adjList.get(vertex1).remove(vertex2);
            adjList.get(vertex2).remove(vertex1);

            return true;
        }

        return false;
    }

    public boolean removeVertex(String vertex){
        if (adjList.get(vertex) == null) return false;
        for(String otherVertex : adjList.get(vertex)){
            adjList.get(otherVertex).remove(vertex);
        }
        adjList.remove(vertex);
        return true;
    }

    public List<String> adjacency(String vertex) {
        if (!adjList.containsKey(vertex)) {
            throw new IllegalArgumentException("O vértice \"" + vertex
+ "\" não existe no grafo.");
        }
        return new ArrayList<>(adjList.get(vertex));
    }
```

```
    public int getVertexDegree(String vertex){
        if (!adjList.containsKey(vertex)) {
            throw new IllegalArgumentException("O Vertice \"" + vertex
+ "\" nao existe no grafo.");
        }
        return adjList.get(vertex).size();
    }

    public HashMap<String, Integer> getAllVertexDegree(){
        HashMap<String, Integer> degreeMap = new HashMap<>();
        for(String vertex : adjList.keySet()){
            degreeMap.put(vertex, getVertexDegree(vertex));
        }

        return degreeMap;
    }

    @Override
    public String toString() {
        return "Graph{" +

                "adjList=" + adjList +

                '}';
    }

    public static void main(String[] args) {

        Graph myGraph = new Graph();

        myGraph.addVertex("A");
        myGraph.addVertex("B");
        myGraph.addVertex("C");
        myGraph.addVertex("D");

        myGraph.addEdge("A", "B");
        myGraph.addEdge("A", "C");
        myGraph.addEdge("A", "D");
        myGraph.addEdge("B", "D");
        myGraph.addEdge("C", "D");

        System.out.println(myGraph.toString());

        System.out.println("###################################");

        System.out.println(myGraph.getVertexDegree("A"));

        System.out.println("###################################");
        System.out.println(myGraph.getAllVertexDegree());
        System.out.println("###################################");
        System.out.println(myGraph.adjacency("A"));
        System.out.println("###################################");


        myGraph.removeVertex("D");

        System.out.println(myGraph.toString());

        System.out.println("###################################");
```

```
        System.out.println(myGraph.getVertexDegree("A"));

        System.out.println("################################");
        System.out.println(myGraph.getAllVertexDegree());
        System.out.println("################################");

    }
}
```

Matriz:

```
package org.example;

import java.util.*;

public class GraphMatrix {
    private Map<String, Integer> vertexIndexMap = new HashMap<>();
    private List<String> vertices = new ArrayList<>();
    private boolean[][] adjMatrix = new boolean[10][10]; // capacidade
inicial
    private int size = 0;

    private void ensureCapacity() {
        if (size >= adjMatrix.length) {
            int newSize = adjMatrix.length * 2;
            boolean[][] newMatrix = new boolean[newSize][newSize];
            for (int i = 0; i < adjMatrix.length; i++) {
                System.arraycopy(adjMatrix[i], 0, newMatrix[i], 0,
adjMatrix[i].length);
            }
            adjMatrix = newMatrix;
        }
    }

    public boolean addVertex(String vertex) {
        if (vertexIndexMap.containsKey(vertex)) return false;
        ensureCapacity();
        vertexIndexMap.put(vertex, size);
        vertices.add(vertex);
        size++;
        return true;
    }

    public boolean addEdge(String vertex1, String vertex2) {
        Integer i = vertexIndexMap.get(vertex1);
        Integer j = vertexIndexMap.get(vertex2);
        if (i == null || j == null) return false;
        adjMatrix[i][j] = true;
        adjMatrix[j][i] = true;
        return true;
    }

    public boolean removeEdge(String vertex1, String vertex2) {
        Integer i = vertexIndexMap.get(vertex1);
        Integer j = vertexIndexMap.get(vertex2);
        if (i == null || j == null) return false;
        adjMatrix[i][j] = false;
        adjMatrix[j][i] = false;
        return true;
```

```
    }

    public boolean removeVertex(String vertex) {
        Integer index = vertexIndexMap.get(vertex);
        if (index == null) return false;

        vertexIndexMap.remove(vertex);
        vertices.remove((int) index);

        for (int i = index; i < size - 1; i++) {
            adjMatrix[i] = adjMatrix[i + 1];
            for (int j = 0; j < size; j++) {
                adjMatrix[j][i] = adjMatrix[j][i + 1];
            }
        }

        for (int i = index; i < vertices.size(); i++) {
            vertexIndexMap.put(vertices.get(i), i);
        }

        size--;
        return true;
    }

    public List<String> adjacency(String vertex) {
        Integer i = vertexIndexMap.get(vertex);
        if (i == null) {
            throw new IllegalArgumentException("O vértice \"" + vertex
+ "\" não existe no grafo.");
        }

        List<String> neighbors = new ArrayList<>();
        for (int j = 0; j < size; j++) {
            if (adjMatrix[i][j]) {
                neighbors.add(vertices.get(j));
            }
        }
        return neighbors;
    }

    public int getVertexDegree(String vertex) {
        return adjacency(vertex).size();
    }

    public Map<String, Integer> getAllVertexDegree() {
        Map<String, Integer> degreeMap = new HashMap<>();
        for (String vertex : vertices) {
            degreeMap.put(vertex, getVertexDegree(vertex));
        }
        return degreeMap;
    }

    public static void main(String[] args) {
        GraphMatrix graph = new GraphMatrix();

        graph.addVertex("A");
        graph.addVertex("B");
        graph.addVertex("C");
        graph.addVertex("D");

        graph.addEdge("A", "B");
```

```
        graph.addEdge("A", "C");
        graph.addEdge("B", "D");

        System.out.println("Adjacência de A: " +
graph.adjacency("A")); // Esperado: [B, C]
        System.out.println("Adjacência de B: " +
graph.adjacency("B")); // Esperado: [A, D]

        System.out.println("Grau de A: " +
graph.getVertexDegree("A")); // Esperado: 2
        System.out.println("Grau de B: " +
graph.getVertexDegree("B")); // Esperado: 2

        System.out.println("Grau de todos os vértices:");
        for (var entry : graph.getAllVertexDegree().entrySet()) {
            System.out.println(entry.getKey() + ": " +
entry.getValue());
        }

        graph.removeEdge("A", "C");
        System.out.println("Adjacência de A após remover aresta com C:
" + graph.adjacency("A")); // Esperado: [B]

        graph.removeVertex("B");
        System.out.println("Adjacência de A após remover vértice B: "
+ graph.adjacency("A")); // Esperado: []

        System.out.println("Grau de todos os vértices após
remoções:");
        for (var entry : graph.getAllVertexDegree().entrySet()) {
            System.out.println(entry.getKey() + ": " +
entry.getValue());
        }
    }
}
```