```java
import java.util.LinkedList;

public class FordFulkerson {
    private int[] parent;
    private Queue<Integer> queue;
    private int numVertices;
    private boolean[] visited;

    public FordFulkerson(int numVertices) {
        this.numVertices = numVertices;
        this.queue = new LinkedList<Integer>();
        parent = new int[numVertices + 1];
        visited = new boolean[numVertices + 1];
    }

    public boolean bfs(int s, int goal, int graph[][]) {
        boolean pathFound = false;
        int d, element;

        for(int vertex = 1; vertex <= numVertices; vertex++) {
            parent[vertex] = -1;
            visited[vertex] = false;
        }
        queue.add(s);
        parent[s] = -1;
        visited[s] = true;

        while (!queue.isEmpty()) {
            element = queue.remove();
            d = 1;

            while (d <= numVertices) {
                if (graph[element][d] > 0 &&  !visited[d]) {
                    parent[d] = element;
                    queue.add(d);
                    visited[d] = true;
                }
                d++;
            }
        }
        if(visited[goal]) {
```

```
44              pathFound = true;
45          }
46          return pathFound;
47      }
48
49      public int fordFulkerson(int graph[][], int s, int d) {
50          int u, v;
51          int maxFlow = 0;
52          int pathFlow;
53
54          int[][] residualGraph = new int[numVertices + 1]
    [numVertices + 1];
55          for (int sVertex = 1; sVertex <= numVertices; sVertex++) {
56              for (int dVertex = 1; dVertex <= numVertices; dVertex+
    +) {
57                  residualGraph[sVertex][dVertex] = graph[sVertex]
    [dVertex];
58              }
59          }
60
61          while (bfs(s ,d, residualGraph)) {
62              pathFlow = Integer.MAX_VALUE;
63              for (v = d; v != s; v = parent[v]) {
64                  u = parent[v];
65                  pathFlow = Math.min(pathFlow, residualGraph[u][v]);
66              }
67              for (v = d; v != s; v = parent[v]) {
68                  u = parent[v];
69                  residualGraph[u][v] -= pathFlow;
70                  residualGraph[v][u] += pathFlow;
71              }
72              maxFlow += pathFlow;
73          }
74          return maxFlow;
75      }
76
77      public static void main(String...arg) {
78          int[][] graph;
79          int numNodes;
80          int source;
81          int sink;
```

```
82          int maxFlow;
83
84          Scanner scanner = new Scanner(System.in);
85          System.out.println("Enter the number of nodes");
86          numNodes = scanner.nextInt();
87          graph = new int[numNodes + 1][numNodes + 1];
88
89          System.out.println("Enter the graph matrix");
90          for (int sVertex = 1; sVertex <= numNodes; sVertex++) {
91             for (int dVertex = 1; dVertex <= numNodes; dVertex++) {
92                 graph[sVertex][dVertex] = scanner.nextInt();
93             }
94          }
95          System.out.println("Source:");
96          source= scanner.nextInt();
97          System.out.println("Sink:");
98          sink = scanner.nextInt();
99          FordFulkerson fordFulkerson = new FordFulkerson(numNodes);
100         maxFlow = fordFulkerson.fordFulkerson(graph, source, sink);
101         System.out.println("The Max Flow is " + maxFlow);
102         scanner.close();
103     }
104 }
```