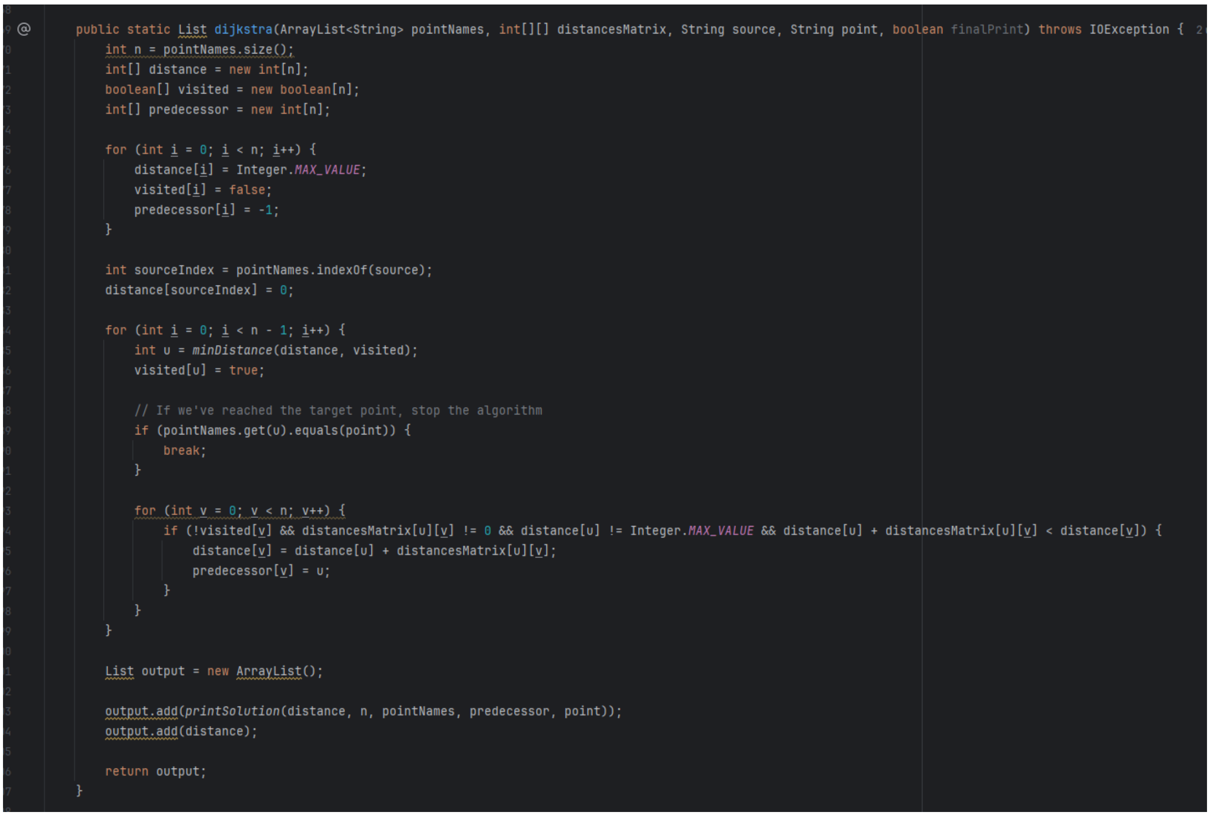
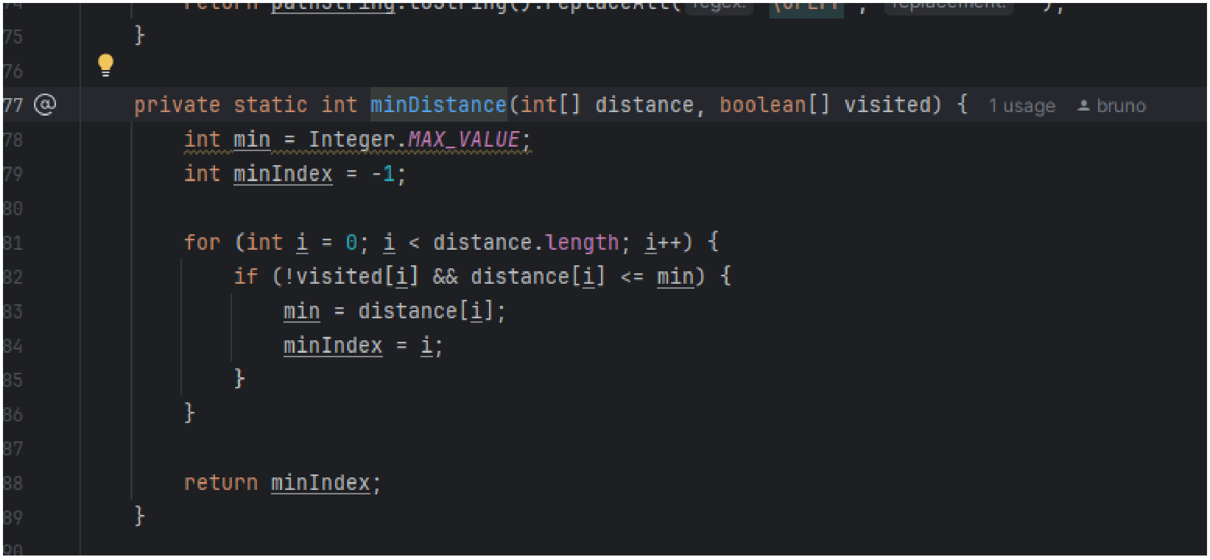


```java
public static List dijkstra(ArrayList<String> pointNames, int[][] distancesMatrix, String source, String point, boolean finalPrint) throws IOException {
    int n = pointNames.size();
    int[] distance = new int[n];
    boolean[] visited = new boolean[n];
    int[] predecessor = new int[n];

    for (int i = 0; i < n; i++) {
        distance[i] = Integer.MAX_VALUE;
        visited[i] = false;
        predecessor[i] = -1;
    }

    int sourceIndex = pointNames.indexOf(source);
    distance[sourceIndex] = 0;

    for (int i = 0; i < n - 1; i++) {
        int u = minDistance(distance, visited);
        visited[u] = true;

        // If we've reached the target point, stop the algorithm
        if (pointNames.get(u).equals(point)) {
            break;
        }

        for (int v = 0; v < n; v++) {
            if (!visited[v] && distancesMatrix[u][v] != 0 && distance[u] != Integer.MAX_VALUE && distance[u] + distancesMatrix[u][v] < distance[v]) {
                distance[v] = distance[u] + distancesMatrix[u][v];
                predecessor[v] = u;
            }
        }
    }

    List output = new ArrayList();

    output.add(printSolution(distance, n, pointNames, predecessor, point));
    output.add(distance);

    return output;
}
```



```java
    }

    private static int minDistance(int[] distance, boolean[] visited) {
        int min = Integer.MAX_VALUE;
        int minIndex = -1;

        for (int i = 0; i < distance.length; i++) {
            if (!visited[i] && distance[i] <= min) {
                min = distance[i];
                minIndex = i;
            }
        }

        return minIndex;
    }
```