```java
package DES;
import java.util.ArrayList;
import java.util.List;
import java.util.PriorityQueue;

public class Puzzle {

    public int dimension = 3;


    int[] row = { 1, 0, -1, 0 };
    int[] col = { 0, -1, 0, 1 };

    public int calculateCost(int[][] initial, int[][] goal) {
        int count = 0;
        int n = initial.length;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (initial[i][j] != 0 && initial[i][j] != goal[i][j]) {
                    count++;
                }
            }
        }
        return count;
    }

    public void printMatrix(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix.length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }

    public boolean isSafe(int x, int y) {
        return (x >= 0 && x < dimension && y >= 0 && y < dimension);
    }

    public void printPath(Node root) {
        if (root == null) {
            return;
        }
        printPath(root.parent);
```

```java
                printMatrix(root.matrix);
                System.out.println();
        }

        public boolean isSolvable(int[][] matrix) {
                int count = 0;
                List<Integer> array = new ArrayList<Integer>();

                for (int i = 0; i < matrix.length; i++) {
                        for (int j = 0; j < matrix.length; j++) {
                                array.add(matrix[i][j]);
                        }
                }

                Integer[] anotherArray = new Integer[array.size()];
                array.toArray(anotherArray);

                for (int i = 0; i < anotherArray.length - 1; i++) {
                        for (int j = i + 1; j < anotherArray.length; j++) {
                                if (anotherArray[i] != 0 && anotherArray[j] != 0 && anotherArray[i] >
anotherArray[j]) {
                                        count++;
                                }
                        }
                }

                return count % 2 == 0;
        }

        public void solve(int[][] initial, int[][] goal, int x, int y)
        {
                PriorityQueue<Node> pq = new PriorityQueue<Node>(1000, (a, b) -> (a.cost +
a.level) - (b.cost + b.level));
                Node root = new Node(initial, x, y, x, y, 0, null);
                root.cost = calculateCost(initial, goal);
                pq.add(root);

                while (!pq.isEmpty()) {
                        Node min = pq.poll();
                        if (min.cost == 0) {
                                printPath(min);
                                return;
                        }
```

```java
                        for (int i = 0; i < 4; i++) {
                if (isSafe(min.x + row[i], min.y + col[i])) {
                        Node child = new Node(min.matrix, min.x, min.y, min.x + row[i], min.y +
col[i], min.level + 1, min);
                        child.cost = calculateCost(child.matrix, goal);
                        pq.add(child);
                }
            }
                }
        }

        public static void main(String[] args) {
                int[][] initial = { {1, 2, 3}, {0, 5, 4}, {7, 6, 8} };
                int[][] goal    = { {1, 2, 3}, {4, 5, 6}, {7, 8, 0} };


                int x = 1, y = 0;

                Puzzle puzzle = new Puzzle();
                if (puzzle.isSolvable(initial)) {
                        puzzle.solve(initial, goal, x, y);
                }
                else {
                        System.out.println("The given initial is impossible to solve");
                }
        }

}
```

Output:

1 2 3
0 5 4
7 6 8

1 2 3
7 5 4
0 6 8

1 2 3
7 5 4
6 0 8

```
1 2 3
7 0 4
6 5 8


1 2 3
7 4 0
6 5 8


1 2 3
7 4 8
6 5 0


1 2 3
7 4 8
6 0 5


1 2 3
7 4 8
0 6 5


1 2 3
0 4 8
7 6 5


1 2 3
4 0 8
7 6 5


1 2 3
4 6 8
7 0 5


1 2 3
4 6 8
7 5 0


1 2 3
4 6 0
7 5 8


1 2 3
4 0 6
7 5 8
```

```
1 2 3
4 5 6
7 0 8

1 2 3
4 5 6
7 8 0
```