```
public class Grid {
    private final int width;
    private final int height;
    private final Status[][] grid;
    private boolean empty = true;

    public Grid(int width, int height) {

        this.width = width;
        this.height = height;
        grid = new Status[width][height];
        initialiseGrid();
    }

    public boolean isEmpty() {
        return empty;
    }

    public void addBomb(int columnIndex, int rowIndex) {
        grid[columnIndex][rowIndex] = Status.BOMB;
        empty = false;

        for (int i = -1; i <= 1; i++) {
            for (int j = -1; j <= 1; j++) {
                int col = columnIndex + i;
                int row = rowIndex + j;

                if (isIndexInsideGrid(col, row)) {
                    grid[col][row] = grid[col][row].increment();
                }
            }

        }

    }

    private boolean isIndexInsideGrid(int col, int row) {
        return col >=0 && col < width && row >=0 && row<height;
    }

    public Status getStatus(int columnIndex, int rowIndex) {
        return grid[columnIndex][rowIndex];

    }
```

```
    private void initialiseGrid() {
        for (int i = 0; i < grid.length; i++) {
            for (int j = 0; j < grid[i].length; j++) {
                grid[i][j] = Status.ZERO;
            }
        }
    }

}
```

```java
public enum Status {
    ONE(null), BOMB(), ZERO(ONE);

    private Status next;

    private Status() {
        next = this;
    }

    private Status(Status next) {

        this.next = next;
    }

    public Status increment() {
        return next;
    }
}
```

```
import junit.framework.Assert;
import org.junit.Test;

public class GridTest {

    @Test
    public void shouldGetAnEmptyGridWhenThereIsNoInput() throws
Exception {
        Grid grid = new Grid(0,0);
        Assert.assertTrue(grid.isEmpty());
    }

    @Test
    public void shouldNotBeEmptyWhenYouAddBombToTheGrid() throws
Exception {
        Grid grid = new Grid(4, 4);
        grid.addBomb(0, 1);

        Assert.assertFalse(grid.isEmpty());
    }

    @Test
    public void shouldReturnCorrectNumberOfBombsFromAdjacentCells()
throws Exception {
        Grid grid = new Grid(4, 4);
        grid.addBomb(0, 1);

        Status actualStatus = grid.getStatus(0, 0);
        Assert.assertEquals(Status.ONE, actualStatus);
    }


}
```