

```

1  #pragma once
2
3  #include <iostream>
4  #include <chrono>
5
6  #include "door.hpp"
7
8  /// CHECKED FOR THREAD SAFTEY //
9
10 enum FloorNum {
11     FG = 0,
12     F1,
13     F2,
14     F3
15 };
16
17 class invalid_floor_reached : public std::exception {
18 public:
19     std::string what() {
20         return "Umm, ya you killed people. Invalid floor reached";
21     }
22 };
23
24 /// I know I know What am I doing here, Do I even know C++.
25 /// Ill fix it later
26 inline void inc(std::atomic<FloorNum>& a) {
27     switch (a) {
28         case FG: a = F1; return;
29         case F1: a = F2; return;
30         case F2: a = F3; return;
31         default: throw invalid_floor_reached();
32     }
33 }
34
35 inline void dec(std::atomic<FloorNum>& a) {
36     switch (a) {
37         case F1: a = FG; return;
38         case F2: a = F1; return;
39         case F3: a = F2; return;
40         default: throw invalid_floor_reached();
41     }
42 }
43
44 // inline void operator++ (FloorNum& a) {
45 //     switch (a) {
46 //         case FG: a = F1; return;
47 //         case F1: a = F2; return;
48 //         case F2: a = F3; return;
49 //         default: throw invalid_floor_reached();
50 //     }
51 // }
52
53 // inline void operator++ (FloorNum& a, int) {
54 //     switch (a) {
55 //         case FG: a = F1; return;
56 //         case F1: a = F2; return;
57 //         case F2: a = F3; return;
58 //         default: throw invalid_floor_reached();
59 //     }
60 // }
61
62 // inline void operator-- (FloorNum& a) {
63 //     switch (a) {
64 //         case F1: a = FG; return;
65 //         case F2: a = F1; return;
66 //         case F3: a = F2; return;
67 //         default: throw invalid_floor_reached();
68 //     }
69 // }

```

```

70
71 // inline void operator-- (FloorNum& a, int) {
72 //     switch (a) {
73 //         // case F1: a = FG; return;
74 //         // case F2: a = F1; return;
75 //         // case F3: a = F2; return;
76 //         // default:         throw invalid_floor_reached();
77 //     }
78 // }
79
80 enum ElevState {
81     ES_WAIT = 0b00,
82     ES_DOWN = 0b01,
83     ES_UP = 0b10,
84     ES_DC = 0b11 // wont be used (dont care)
85 };
86
87 inline std::string pretty(ElevState s) {
88     switch (s) {
89         case ES_WAIT:
90             return "Waiting";
91         case ES_DOWN:
92             return "Down";
93         case ES_UP:
94             return "Up";
95         default:
96             return "Broken";
97     }
98 }
99
100 struct Elevator {
101     std::atomic<FloorNum> mFloor;
102     std::atomic<ElevState> mState;
103     Door mDoor;
104     std::thread mThread;
105
106     std::atomic_bool mStop;
107
108     Elevator();
109     ~Elevator();
110
111     void start();
112     void reset(FloorNum flr);
113
114 };

```