```
    1: // Copyright 2023 Thomas O'Connor
    2: #ifndef SOKOBAN_HPP
    3: #define SOKOBAN_HPP
    4:
    5: #include <algorithm>
    6: #include <fstream>
    7: #include <iostream>
    8: #include <string>
    9: #include <vector>
   10: #include <SFML/Audio.hpp>
   11: #include <SFML/Graphics.hpp>
   12: #include <SFML/System.hpp>
   13: #include <SFML/Window.hpp>
   14:
   15: using sf::Keyboard;
   16: using std::cout;
   17: using std::endl;
   18: using std::pair;
   19: using std::vector;
   20:
   21: #define TILE_SIZE 64
   22:
   23: enum Direction { UP, LEFT, DOWN, RIGHT };
   24:
   25: class Sokoban : public sf::Drawable {
   26:  public:
   27:     // Constructors
   28:     Sokoban() : _h(0), _w(0) {}
   29:     Sokoban(int y, int x) : _h(y), _w(x) {}
   30:
   31:     // Getters
   32:     int getWidth() const { return _w; }
   33:     int getHeight() const { return _h; }
   34:     int getTurn() const { return turn; }
   35:
   36:     // Display
   37:     void drawElapsingTime(sf::RenderWindow &window, sf::Clock &clock);
   38:     // Check and Display
   39:     bool isWon() const;
   40:
   41:     // Interactors
   42:     void movePlayer(Direction dir);
   43:     void restart(sf::Clock& clock);
   44:     void undo();
   45:     void playSound();
   46:
   47:     // Overload extraction operator
   48:     friend Sokoban& operator>>(Sokoban& game, std::ifstream& file);
   49:
   50:  private:
   51:     // Draw game in SFML
   52:     virtual void draw(sf::RenderTarget& target, sf::RenderStates states)
const;
   53:     // Test for conditions
   54:     bool noObstructions(Direction dir, int w, int h) const;
   55:     bool canPushBox(Direction dir, int w, int h) const;
   56:     // Push box if conditions are met
   57:     void pushBox(Direction dir, int w, int h);
   58:
   59:  private:
   60:     // Game state stored in row-major order
   61:     // Vector of 2D game states - 3rd dimension is time
   62:     // Allows for "undo" actions
   63:     vector<vector<vector<char>>> allGameStates;
   64:     // Dimensions of the window/game (read from file)
```

```
65:     int _h, _w;
66:     // Turn number for keeping track of game states
67:     int turn = 0;
68:     // Direction the man faces
69:     vector<Direction> face;
70: };
71:
72: // non-member helper functions
73: vector<vector<char>> deepCopy(const vector<vector<char>> &original);
74:
75: #endif
```