```cpp
#include "Board.h"
#include "Slot.h"
#include <time.h>
#include <stdlib.h>
#include <iostream>
#include <iomanip>
using namespace std;

string center(int width, const string& str) {
    int len = str.length();
    if (width < len) { return str; }

    int diff = width - len;
    int pad1 = diff / 2;
    int pad2 = diff - pad1;
    return string(pad1, ' ') + str + string(pad2, ' ');
}

void draw_edge_line(int width, const string* line)
{
    cout << "|" << string((width) * 5 + 4, '-') << "|" << endl;
    cout << "|" << string((width) * 5 + 4, ' ') << "|" << endl;
    cout << "|";
    for (int col = 0; col < 5; col++)
        cout << center(width, line[col]) << "|";
    cout << endl;
    cout << "|" << string((width) * 5 + 4, ' ') << "|" << endl;
    cout << "|" << string((width) * 5 + 4, '-') << "|" << endl;
}

void draw_inner_line(int width, const string* line, bool last)
{
    cout << "|" << string(width * 5 + 4, ' ') << "|" << endl;
    cout << "|" << center(width, line[0]) << "|";
    cout << string(width * 3 + 2, ' ') << "|";
    cout << center(width, line[4]) << "|" << endl;
    cout << "|" << string(width * 5 + 4, ' ') << "|" << endl;
    if (!last)
    {
        cout << "|" << string(width, '-') << "|" << string((width * 3 + 3), ' 
          ');
        cout << string(width, '-') << "|" << endl;
    }
}


ostream& operator<<(ostream& os, const Board& b)
{
    draw_edge_line(b.m_slot_width, b.m_board_image[0]);
    for (int row = 1; row < 5; row++)
        draw_inner_line(b.m_slot_width, b.m_board_image[row], row == 4);
    draw_edge_line(b.m_slot_width, b.m_board_image[5]);
    return os;
}

void Board::init_board_image()
```

```cpp
{
    m_board_image[0][0] = m_arr[9]->get_name();
    m_board_image[0][1] = m_arr[10]->get_name();
    m_board_image[0][2] = m_arr[11]->get_name();
    m_board_image[0][3] = m_arr[12]->get_name();
    m_board_image[0][4] = m_arr[13]->get_name();
    m_board_image[1][0] = m_arr[8]->get_name();
    m_board_image[1][1] = "";
    m_board_image[1][2] = "";
    m_board_image[1][3] = "";
    m_board_image[1][4] = m_arr[14]->get_name();
    m_board_image[2][0] = m_arr[7]->get_name();
    m_board_image[2][1] = "";
    m_board_image[2][2] = "";
    m_board_image[2][3] = "";
    m_board_image[2][4] = m_arr[15]->get_name();
    m_board_image[3][0] = m_arr[6]->get_name();
    m_board_image[3][1] = "";
    m_board_image[3][2] = "";
    m_board_image[3][3] = "";
    m_board_image[3][4] = m_arr[16]->get_name();
    m_board_image[4][0] = m_arr[5]->get_name();
    m_board_image[4][1] = "";
    m_board_image[4][2] = "";
    m_board_image[4][3] = "";
    m_board_image[4][4] = m_arr[17]->get_name();
    m_board_image[5][0] = m_arr[4]->get_name();
    m_board_image[5][1] = m_arr[3]->get_name();
    m_board_image[5][2] = m_arr[2]->get_name();
    m_board_image[5][3] = m_arr[1]->get_name();
    m_board_image[5][4] = m_arr[0]->get_name();

    m_slot_width = 0;
    for (int row = 0; row < 6; row++)
    {
        for (int col = 0; col < 5; col++)
            if (m_board_image[row][col].size() > m_slot_width)
                m_slot_width = m_board_image[row][col].size();
    }
    m_slot_width += 2;
}

Board::Board()
{
    srand(time(NULL));
    m_size = 0;
    add_go_slot("GO!");
    add_asset_slot("Jerusalem", "zoo");
    add_asset_slot("Jerusalem", "David_tower");
    add_asset_slot("Jerusalem", "Western_wall");
    add_jail_slot("JAIL! Wait 1 turn");

    add_asset_slot("Tel_Aviv", "Hilton");
    int num = rand() % 1000 + 500;
    add_chance_slot("You won the lottery", num);
    add_asset_slot("Tel_Aviv", "Azrieli");
    add_asset_slot("Tel_Aviv", "Habima");
    num = rand() % 200 + 100;
    add_chance_slot("You have to pay the IRS", -num);

    add_asset_slot("Carmiel", "Rocks_park");
    num = rand() % 70 + 30;
    add_chance_slot("Congratulations to your birthday, get a bouquet", num);
    add_asset_slot("Carmiel", "Big");
    add_asset_slot("Carmiel", "Ort_Braude");

    add_asset_slot("Eilat", "Dolpin_reef");
    add_asset_slot("Eilat", "Kings_town");
    add_asset_slot("Eilat", "Ramon_airport");
    add_asset_slot("Eilat", "Almog_beach");
    init_board_image();
}

void Board::increase_board()
{
    Slot ** tmp = new Slot *[m_size + 1];
    int i;
    for (i = 0; i < m_size; i++)
        tmp[i] = m_arr[i];
    m_size++;
    if (m_arr)
        delete[] m_arr;
    m_arr = tmp;
}

void Board::add_asset_slot(const string& city, const string& asset_name)
{
    increase_board();
    m_arr[m_size-1] = new Asset(m_size, city, asset_name);
}


void Board::add_go_slot(const string& text)
{
    increase_board();
    m_arr[m_size - 1] = new Go(m_size, text);
}

void Board::add_jail_slot(const string& text)
{
    increase_board();
    m_arr[m_size - 1] = new Jail(m_size, text);
}

void Board::add_chance_slot(const string& text, float amount)
{
    increase_board();
    m_arr[m_size - 1] = new Chance(m_size, text, amount);
}

int Board::size() const
{
```

```cpp
    return m_size;
}

Slot * Board::operator[](int idx) const
{
    return m_arr[idx];
}

istream& operator >> (istream& is, Board::action& i)
{
    int tmp;
    if (is >> tmp)
        i = (Board::action)(tmp);
    return is;
}

void Board::print_help()
{
    cout << "\nto continue press (" << PLAY << "),";
    cout << " To print board press(" << PRINT_BOARD << "),";
    cout << " To end game press(" << END_GAME << ")\n";
}

Board::action Board::get_command() const
{
    Board::action cmd;
    cin >> cmd;
    if (cin.fail() || cmd < 0 || cmd > 2)
    {
        cin.clear();
        cin.ignore();
        return get_command();
    }
    return cmd;
}

void Board::play(Player* players)
{
    int player = 0;
    action a;
    while (1)
    {
        cout << players[player].get_name() << "'s turn: ";
        print_help();
        a = (action)get_command();

        if (a == END_GAME)
            break;
        else if (a == PRINT_BOARD)
        {
            cout << *this;
            continue;
        }
        else if (a == PLAY)
        {
            if (!(players[player]).draw_dice())
```

```
                break;
            cout << players[player];
            player = (player + 1) % Player::get_counter();
        }
    }
    cout << "End of Game! Bye!" << endl;
}

```