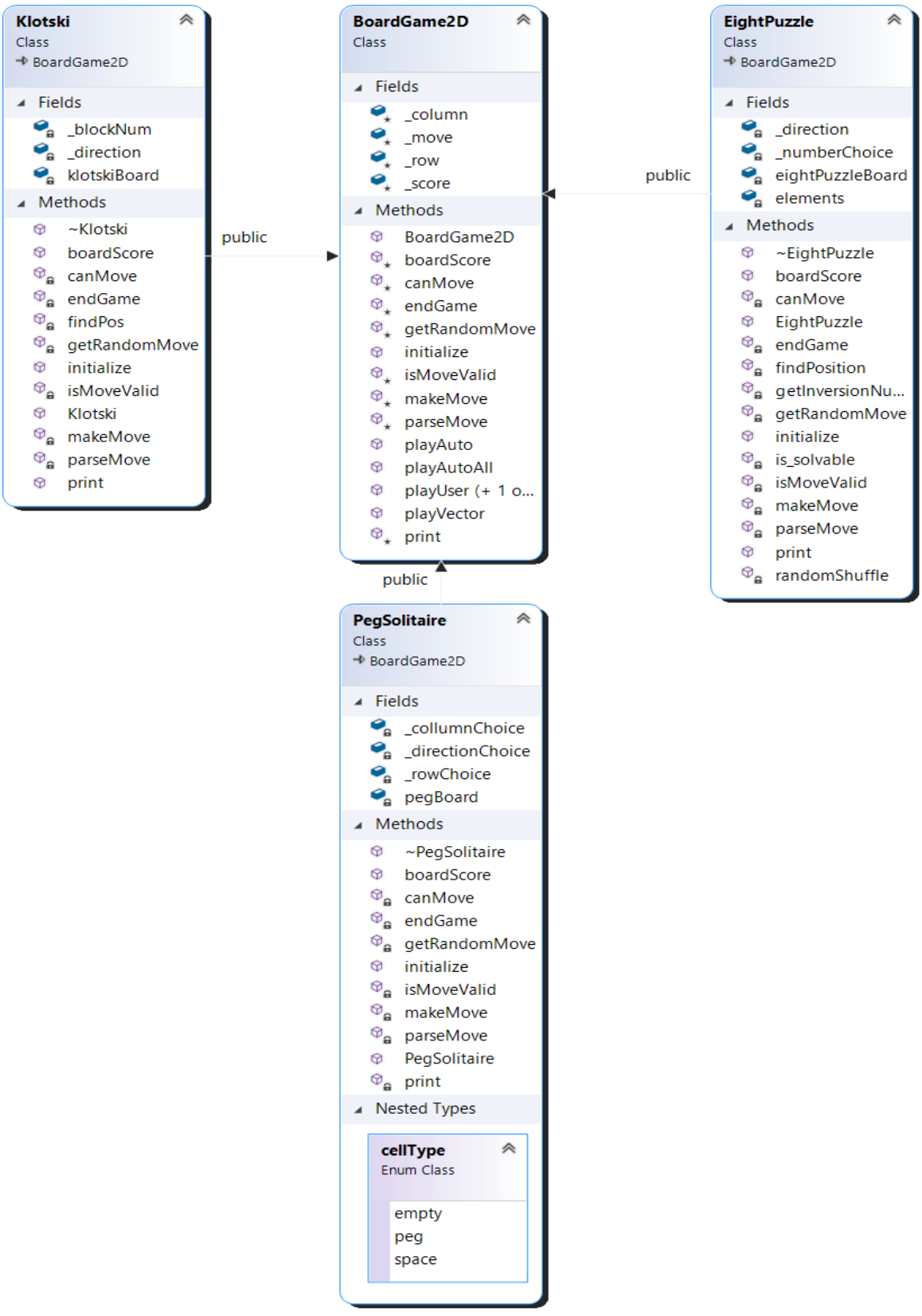
HOMEWORK REPORT

MUHAMMED SINAN PEHLIVANOGLU

1901042664

CLASS DIAGRAM



INITIALIZATION FUNCTIONS

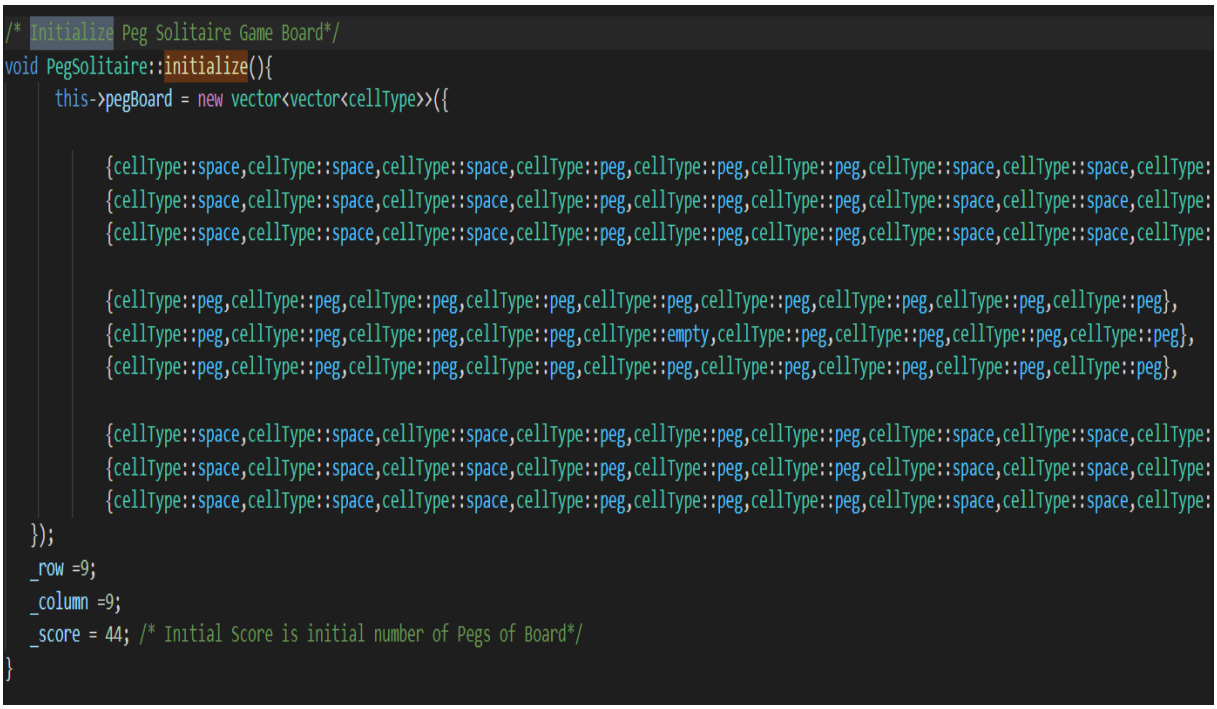
```
/* Initialize Klotski Board*/
void Klotski::initialize(){

    this->klotskiBoard = new vector<vector<char>>({

        {'1', '2', '2','3'},
        {'1', '2', '2','3'},
        {'4', '5', '5','6'},
        {'4', '7', '8','6'},
        {'9', 'e', 'e','0'}

    });
    _row =5;
    _column =4;
    _score =0; /* Set score zero initially, Low score is much better*/
}
```

- The Klotski Board is initialized in run time. Each number represent a block. The ‘e’ character Represent empty block.
- The board’s row and column size is setted.
- The score shows the number of moves made.



```
/* Initialize Peg Solitaire Game Board*/
void PegSolitaire::initialize(){
    this->pegBoard = new vector<vector<cellType>>({

        {cellType::space,cellType::space,cellType::space,cellType::peg,cellType::peg,cellType::peg,cellType::space,cellType::space,cellType:
        {cellType::space,cellType::space,cellType::space,cellType::peg,cellType::peg,cellType::peg,cellType::space,cellType::space,cellType:
        {cellType::space,cellType::space,cellType::space,cellType::peg,cellType::peg,cellType::peg,cellType::space,cellType::space,cellType:

        {cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::peg},
        {cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::empty,cellType::peg,cellType::peg,cellType::peg,cellType::peg},
        {cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::peg,cellType::peg},

        {cellType::space,cellType::space,cellType::space,cellType::peg,cellType::peg,cellType::peg,cellType::space,cellType::space,cellType:
        {cellType::space,cellType::space,cellType::space,cellType::peg,cellType::peg,cellType::peg,cellType::space,cellType::space,cellType:
        {cellType::space,cellType::space,cellType::space,cellType::peg,cellType::peg,cellType::peg,cellType::space,cellType::space,cellType:
  });
  _row =9;
  _column =9;
  _score = 44; /* Initial Score is initial number of Pegs of Board*/
}
```

- Peg Board is nested two diamensional cell type vector. The cellType is enum class that represent celltype. There are 3 Cell type space , peg , and empty.
- The board's row and column size is setted.
- The initial score is 44 becase of the initial number of pegs in the board.

```
/* If the Board is solvable then initialize the board*/
void EightPuzzle::initialize(){
    this->eightPuzzleBoard = new vector<vector<int>>(3, vector<int>(3));
    do
    {

        randomShuffle(elements);
        int counter =0;
        for(int i=0 ; i< _row ; ++i){
            for(int j =0 ; j<_column ; ++j, ++ counter){
                (*eightPuzzleBoard)[i][j] = elements[counter];
            }
        }

    }while (!is_solvable());
    _score =0;
}
```

- Firstly the memory is allocated to the board.
- Integer vector Element represent element of the board. Between(0-8). 0 represent empty cell.
- Element is shuffled by STL random_shuffle method.
- Shuffled elements variable is assigned to the puzzle board. This continues until the board is solvable.
- The score shows the number of moves made.

PLAY FUNCTIONS

```
/* Play automatically only once*/
void BoardGame2D::playAuto(){
    while (1)
    {
        this->_move = getRandomMove();
        parseMove();

        if(canMove()){
            makeMove();
            //cout<<(*this)<<endl;
            break;
        }
    }
}
```

- playAuto function work unilt the getRandomMove is valid to move.
- ParseMove function parse the string move into its element.
- The canMove function check the move , if the random move is valid, then make move only once and exit the while loop.

```
/* Play automatically until the game is finished*/
void BoardGame2D::playAutoAll(){

   while (!endGame()) {   playAuto() ;/*std::this_thread::sleep_for (std::chrono::milliseconds(20))*/;}

}
```

- The playAutoAll function call the playAuto function until the game is end.

```
/* Play by user only once turn*/
void  BoardGame2D::playUser(const string& move){


    bool status = false;
    //cout << (*this);

    _move = move;

    parseMove();
    if(canMove())    makeMove();

    else{
       std::cout<<"Could not make a move\n";
    }
}
```

- PlayUser takes string move parameter. Move parameter is assigned to class member _move.
- The _move is parsed.
- The canMove function check the move and if the move valid return 1 then makeMove function is called.
- If the _move is invalid to make move then the user is warned.

```
/* Play by user until the game is finished*/
void BoardGame2D::playUser(){
   do{
      bool status = false;
      cout << (*this);

      do{

        try{

            cout <<"Please Enter the Move : ";
            getline(std::cin,this->_move);
            isMoveValid();
            status = true;

        }
        catch(const char* e )
        {
            cerr <<e;
        }

      }while (!status);


      parseMove();

      if(canMove())   makeMove();
      else{
         std::cout<<"Could not make a move\n";
      }
   }while (!endGame());
}
```

- The playUser method is work until the game is finished by user.
- It forces the user the make valid move syntax.
- Then the move is parsed.
- canMove function is called. If it returns 1, then move is made.

```
/* Get Board games and play them automatically until the games is finished*/
void BoardGame2D::playVector(vector<BoardGame2D*> games){
    for(auto game : games)  {
        cout <<*game<<endl<<endl; game->playAutoAll(); cout <<endl<<*game<<endl<<endl;;

        }
}
```

- playVector method takes parameter as base class pointer vector. Then class playAutoAll for each BoardGame2d derived game.

SCORE FUNCTION

```
/* Getting Game Score*/

int PegSolitaire::boardScore() const{
    return _score;
}
```

- boardScore only return the _score. Score is incresed or decreasid in every valid move with respecet to the game structure.

STREAM INSERTION OPERATOR

```
/* Stream instertion operator to print game's board*/
ostream& operator<<(ostream & out,const  BoardGame2D & base){
    /*out << "\x1b[2J";
        out << "\033[0;0H"; // move cursor to 0,0*/
        out << endl << endl;
    base.print(out);
    return out;
}
```

- Steram insertion operator takes ostream reference and base class parameter.
- Ostream reference is std::cout.
- Print function takes ostream reference as parameter. Print function vary with respect to the game type.
- As a result, it returns ostream reference.