**Project report on Othello**

**Submitted to:**

Dr. PSVS Sai Prasad

Assistant Professor

School of Computer and Information Sciences

University of Hyderabad

**Submitted By:**

Yash Agrawal(18mcmc09)

Sourya Kumar Verma(18mcmc24)

**What is Othello?**

There are sixty-four identical game pieces called disks (often spelled "discs"), which are light on one side and dark on the other.

Players take turns placing disks on the board with their assigned color facing up.

During a play, any disks of the opponent's color that are in a straight line and bounded by the disk just placed and another disk of the current player's color are turned over to the current player's color.

The object of the game is to have the majority of disks turned to display your color when the last playable empty square is filled.

**Rules for the game**

The dark player moves first.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f | g | h |  |
| 1 | [a1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 1 |
| 2 | [a2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 2 |
| 3 | [a3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 3 |
| 4 | [a4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d4O](https://en.wikipedia.org/wiki/File:Reversi_Od44.png) | [e4X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [f4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 4 |
| 5 | [a5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d5X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [e5O](https://en.wikipedia.org/wiki/File:Reversi_Od44.png) | [f5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 5 |
| 6 | [a6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 6 |
| 7 | [a7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 7 |
| 8 | [a8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 8 |
|  | a | b | c | d | e | f | g | h |  |

Starting position

Dark must place a piece with the dark side up on the board, in such a position that there exists at least one straight (horizontal, vertical, or diagonal) occupied line between the new piece and another dark piece, with one or more contiguous light pieces between them. In the below situation, dark has the following options indicated by translucent pieces:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f | g | h |  |
| 1 | [a1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 1 |
| 2 | [a2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 2 |
| 3 | [a3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d3x](https://en.wikipedia.org/wiki/File:Reversi_xd44.png) | [e3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 3 |
| 4 | [a4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c4x](https://en.wikipedia.org/wiki/File:Reversi_xd44.png) | [d4O](https://en.wikipedia.org/wiki/File:Reversi_Od44.png) | [e4X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [f4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 4 |
| 5 | [a5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d5X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [e5O](https://en.wikipedia.org/wiki/File:Reversi_Od44.png) | [f5x](https://en.wikipedia.org/wiki/File:Reversi_xd44.png) | [g5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 5 |
| 6 | [a6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e6x](https://en.wikipedia.org/wiki/File:Reversi_xd44.png) | [f6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 6 |
| 7 | [a7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 7 |
| 8 | [a8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 8 |
|  | a | b | c | d | e | f | g | h |  |

Where dark may play

After placing the piece, dark turns over (flips, captures) all light pieces lying on a straight line between the new piece and any anchoring dark pieces. All reversed pieces now show the dark side, and dark can use them in later moves—unless light has reversed them back in the meantime. In other words, a valid move is one where at least one piece is reversed.

If dark decided to put a piece in the topmost location (all choices are strategically equivalent at this time), one piece gets turned over, so that the board appears thus:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f | g | h |  |
| 1 | [a1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 1 |
| 2 | [a2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 2 |
| 3 | [a3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d3X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [e3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 3 |
| 4 | [a4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d4X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [e4X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [f4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 4 |
| 5 | [a5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d5X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [e5O](https://en.wikipedia.org/wiki/File:Reversi_Od44.png) | [f5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 5 |
| 6 | [a6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 6 |
| 7 | [a7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 7 |
| 8 | [a8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 8 |
|  | a | b | c | d | e | f | g | h |  |

After dark play

Now light plays. This player operates under the same rules, with the roles reversed: light lays down a light piece, causing a dark piece to flip. Possibilities at this time appear thus (indicated by transparent pieces):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f | g | h |  |
| 1 | [a1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 1 |
| 2 | [a2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 2 |
| 3 | [a3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c3o](https://en.wikipedia.org/wiki/File:Reversi_od44.png) | [d3X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [e3o](https://en.wikipedia.org/wiki/File:Reversi_od44.png) | [f3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 3 |
| 4 | [a4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d4X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [e4X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [f4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 4 |
| 5 | [a5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c5o](https://en.wikipedia.org/wiki/File:Reversi_od44.png) | [d5X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [e5O](https://en.wikipedia.org/wiki/File:Reversi_Od44.png) | [f5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 5 |
| 6 | [a6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 6 |
| 7 | [a7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 7 |
| 8 | [a8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 8 |
|  | a | b | c | d | e | f | g | h |  |

Where light may play

Light takes the bottom left option and reverses one piece:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f | g | h |  |
| 1 | [a1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h1](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 1 |
| 2 | [a2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h2](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 2 |
| 3 | [a3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d3X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [e3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h3](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 3 |
| 4 | [a4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d4X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [e4X](https://en.wikipedia.org/wiki/File:Reversi_Xd44.png) | [f4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h4](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 4 |
| 5 | [a5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c5O](https://en.wikipedia.org/wiki/File:Reversi_Od44.png) | [d5O](https://en.wikipedia.org/wiki/File:Reversi_Od44.png) | [e5O](https://en.wikipedia.org/wiki/File:Reversi_Od44.png) | [f5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h5](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 5 |
| 6 | [a6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h6](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 6 |
| 7 | [a7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h7](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 7 |
| 8 | [a8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [b8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [c8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [d8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [e8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [f8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [g8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | [h8](https://en.wikipedia.org/wiki/File:Reversi_d44.png) | 8 |
|  | a | b | c | d | e | f | g | h |  |

After light play

Players take alternate turns. If one player can not make a valid move, play passes back to the other player. When neither player can move, the game ends. This occurs when the grid has filled up or when neither player can legally place a piece in any of the remaining squares. This means the game may end before the grid is completely filled. This possibility may occur because one player has no pieces remaining on the board in that player's color. In over-the-board play this is generally scored as if the board were full (64–0).

**Environmental Setup**

So because we are not able to integrate the frontend and the backend part of the project as we are not able to collaborate because of slow internet.

So we have attached a separate file for front end and backend, after reaching the university we will integrate it and send it to you, for now

**For Backend**

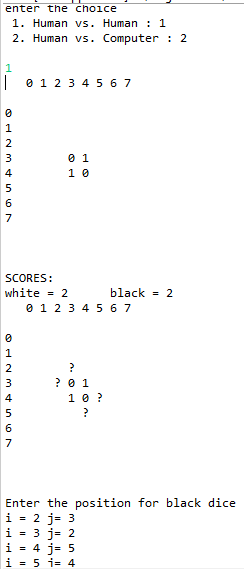
There is a folder in this zip file named backend

After opening to that folder you have to open command line on that folder path.

Compile : javac Front.java

Run: java Front

In command line only you can see the CUI(command user interface) for this game which looks something like this



Where initially you will have to enter 1 for human vs human and 2 for human vs computer

0 represent white dice and 1 represent black dice

? represent possible suggestion moves for respective dice turn

**For Frontend**

You have to open Frontend folder

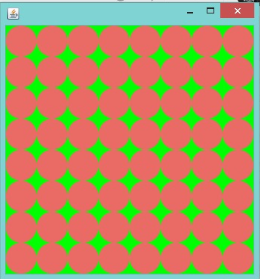
Because we are not able to integrate completely you will not able to run the game in frontend however you can see the gui for the game

All you have to do is to run these

On command line in frontend folder

compile : javac Front.java

run : java Front



It looks like that

**Work Division**

**Sourya kumar verma Role:**

Desing and coding of GUI and integrating GUI with the game (code written by Yash Agarwal)

**Yash Agrawal role:**

Writing code for backend with all functionality and passing that code to Sourya for integration.

**Idea and Working (GUI):**

In the GUI First page is displaying Label with game name (Othello), Selection of play mode (drop down list) and

start button, after clicking start button gameboard will open up. gameboard has 8 rows and 8 coloums. (64 buttonss).

actual idea behind GUI implementation is, displaying the output received in form matrix from the game code (Front.java)

after performing Minimax algorithm. and enabling buttons and changing color of button based on the received matrix

the matrix contains 0,1,2,9

0 ------> White color

1 ------> Black color

9 ------> Pink Color (for Suggestion)

0 1 2 3 4 5 6 7

[

0 [9 ,9 ,9 ,9 ,9 ,9 ,9 ,9 ]

1 [9 ,9 ,9 ,9 ,9 ,9 ,9 ,9 ]

2 [9 ,9 ,9 ,9 ,9 ,9 ,9 ,9 ]

3 [9 ,9 ,9 ,0 ,1 ,9 ,9 ,9 ]

4 [9 ,9 ,9 ,1 ,0 ,9 ,9 ,9 ]

5 [9 ,9 ,9 ,9 ,9 ,9 ,9 ,9 ]

6 [9 ,9 ,9 ,9 ,9 ,9 ,9 ,9 ]

7 [9 ,9 ,9 ,9 ,9 ,9 ,9 ,9 ]

]

Initially (3,3) = 0(White), (3,4) = 1(Black), (4,3) = 1(Black), (4,4) = 0(White)

3 4

here 3 [0, 1]

4 [1, 1]

and after displaying the output based on suggestion user can press where button is pink (other button is disabled can not be clicked).

return back to games actual program and again perform Minimax algorithm and so on

**working of algorithm**

Minimax Algorithm with alpha beta pruning

When we will call minimax algorithm then we will call minimx algo with parameter

**int** minimax(**int** gameBoard[][] , **int** depth , **boolean** isMax , **int** h , **int** alpha , **int** beta)

First we will check whether there is a move for other player or not if not then same player play again for that if negate the isMax value

Then we check for the base condition

1.deapth has reached

2. game ends as there is not that depth this will used when we are at end .

3. when there is no dice of opposition such as (0 – any number) or (any number – 0) score

After that it checks for whether there is move for max player or min player

If the move is for max player then again minimax will call with checking all the possible moves for max player and return utility value and select maximum utility value and insert that move in gameboard

I used alpha beta pruning to reduce time complexity

I used DFS for the minimax algorithm

I can use BFS but because of space complexity I preferred DFS.

All the code are there in this text file and also frontend and backend folder contains all the necessary java files to run the program.

**Code Section**

GUI code

###############################################################################################################################

/\*

AI based game Othello Game created by :

Name : Yash Agarwal (18MCMC09)

Name : Sourya Kumar Verma (18MCMC24)

\*/

import java.awt.Container;

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.EventQueue;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.GridLayout;

import java.awt.BorderLayout;

import java.awt.Color;

import javafx.scene.layout.Border;

import java.util.\*;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class AIOthello { // main class for creating frames , panels and buttons

public JFrame frame; // declaring variables for frame

public JFrame frame1;

//static int arr[];

public JButton btn\_00; // declaring buttons

public JButton btn\_01;

public JButton btn\_02;

public JButton btn\_03;

public JButton btn\_04;

public JButton btn\_05;

public JButton btn\_06;

public JButton btn\_07;

public JButton btn\_10;

public JButton btn\_11;

public JButton btn\_12;

public JButton btn\_13;

public JButton btn\_14;

public JButton btn\_15;

public JButton btn\_16;

public JButton btn\_17;

public JButton btn\_20;

public JButton btn\_21;

public JButton btn\_22;

public JButton btn\_23;

public JButton btn\_24;

public JButton btn\_25;

public JButton btn\_26;

public JButton btn\_27;

public JButton btn\_30;

public JButton btn\_31;

public JButton btn\_32;

public JButton btn\_33;

public JButton btn\_34;

public JButton btn\_35;

public JButton btn\_36;

public JButton btn\_37;

public JButton btn\_40;

public JButton btn\_41;

public JButton btn\_42;

public JButton btn\_43;

public JButton btn\_44;

public JButton btn\_45;

public JButton btn\_46;

public JButton btn\_47;

public JButton btn\_50;

public JButton btn\_51;

public JButton btn\_52;

public JButton btn\_53;

public JButton btn\_54;

public JButton btn\_55;

public JButton btn\_56;

public JButton btn\_57;

public JButton btn\_60;

public JButton btn\_61;

public JButton btn\_62;

public JButton btn\_63;

public JButton btn\_64;

public JButton btn\_65;

public JButton btn\_66;

public JButton btn\_67;

public JButton btn\_70;

public JButton btn\_71;

public JButton btn\_72;

public JButton btn\_73;

public JButton btn\_74;

public JButton btn\_75;

public JButton btn\_76;

public JButton btn\_77;

public JButton button;

public static int xInput; //static interger sending row position

public static int yInput; //static interger sending coloum postion

public static int Pturn; // static integer for sending player turn;

JTextField txtField1; // text field for showing white's score

JTextField txtField2; // text field for showing black'sscore

Dimension size1; // for setting panel postion (optional)

Dimension size2;

JLabel lab1; // Label

JLabel lab2; // Label

ImageIcon pink; // for colourful buttons

ImageIcon blue;

ImageIcon grey;

ImageIcon white;

ImageIcon black;

JPanel pan; //panel

JComboBox jcom; // combobox drop down menu

/\*\*

\* Launch the application.

\*/

public static void Main(AIOthello window)/\*public static void main(String args[])\*/ {

//AIOthello window = new AIOthello();

fun(window);

}

static void fun(AIOthello window) {

window.frame1.setVisible(true); // setting frame visible

}

/\*\*

\* Create the application.

\*/

public AIOthello() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*

\* @return

\*/

public void initialize() {

//enableColor();

//enableKey();

/\*

First frame displaying game name

choosing player

\*/

frame1 = new JFrame(); // creating frame for front page

pan = new JPanel(); // creating panel for front page

pan.setLayout(null); // setting layout so that position can be changed

frame1.setSize(700,500); // setting frame size

JLabel l = new JLabel("Welcome to Othello");

l.setLayout(null);

l.setFont(new Font("Serif", Font.PLAIN, 44)); // setting label font

l.setBounds(160,30,700,200); // setting position and size of label

String s[] = {"Man to Man", "Man to Computer"}; // srting arr for playmode selection

jcom = new JComboBox(s); // creating drop down list for playmode

//jcom.setLayout(null);

jcom.setBounds(250,200,200,30); //setting position and size of dropdown

pan.setBounds(0,0,700,700); // setting position and size of panel

frame1.setResizable(false); // fixing frame resizablity

frame1.getContentPane().add(pan); // adding panel to frame

button = new JButton("Start"); // creating start button

button.setFont(new Font("Serif", Font.PLAIN, 20)); // setting font for button

button.setLayout(null); // setting layout button

button.setBounds(250,300,200,30); // setting position and size of button

frame = new JFrame(); // creating frame for gameboard

frame.setVisible(false); // setting gameboard frame visiblity false

button.addActionListener(new ActionListener() { //event listner for button

public void actionPerformed( ActionEvent e) { // after pressing button following will performed

Pturn = jcom.getSelectedIndex();

Pturn = Pturn + 1;

//JOptionPane.showMessageDialog(frame1, Pturn);

frame1.setVisible(false); // disabling first frame

frame.setVisible(true); // disabling game board frame

enableKey();

}

});

pan.add(button); // adding button to panel

pan.add(l); //adding lebel to panel

pan.add(jcom); // adding dropdown list to panel

frame1.setVisible(true);

JPanel panel = new JPanel(new GridLayout(1, 1)); // panel for showing scores

//panel.setLayout(new FlowLayout());

lab1 = new JLabel("White's Score: "); // label

lab2 = new JLabel("Black's Score: "); //label

txtField1 = new JTextField(); // text field for white's scores

txtField2 = new JTextField(); // text field for black's scores

txtField1.setEnabled(false); // setting uneditable textfield

txtField2.setEnabled(false);

panel.setBounds(10, 10, 100, 100); //setting panel size and postions

size1 = lab1.getPreferredSize(); // getting size of label in size1, size2 variables

size2 = lab2.getPreferredSize();

lab1.setBounds(10, 10, size1.width, size1.height); //setting size and postions

lab2.setBounds(10 + size1.width, 10, size2.width, size2.height);

//lab1.setBounds(0,20 , 50, 50);

//lab2.setBounds(100,20 , 50, 50);

panel.add(lab1);

panel.add(txtField1);

panel.add(lab2);

panel.add(txtField2);

JPanel p1 = new JPanel(new GridLayout(8, 8));// creating 8 by 8 grid for gameboard

pink = new ImageIcon("pink.png"); //creating Icon object

blue = new ImageIcon("blue.png");

black = new ImageIcon("black.png");

white = new ImageIcon("white.png");

btn\_00 = new JButton(pink); // creating button with icon object

btn\_00.setMargin(new Insets(0, 0, 0, 0)); // setting margin of button

btn\_00.setBackground(Color.green);

btn\_00.setBorder(null);

btn\_00.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) { // action listner for button at(0,0)

//btn\_00.setIcon(pink);

xInput = 0; //setting row = 0

yInput = 0; // setting col = 0;

enableKey(); // enabling and disabling buttons based on matrix received by minimix algo

enableColor(); //changing color based on matrix received by minimax algo

}

});

/\*

similarly action listner for all button is done below

\*/

p1.add(btn\_00);

btn\_01 = new JButton(pink);

btn\_01.setMargin(new Insets(0, 0, 0, 0));

btn\_01.setBackground(Color.green);

btn\_01.setBorder(null);

btn\_01.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 0;

yInput = 1;

enableKey();

enableColor();

}

});

p1.add(btn\_01);

btn\_02 = new JButton(pink);

btn\_02.setMargin(new Insets(0, 0, 0, 0));

btn\_02.setBackground(Color.green);

btn\_02.setBorder(null);

btn\_02.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 0;

yInput = 2;

enableKey();

enableColor();

}

});

p1.add(btn\_02);

btn\_03 = new JButton(pink);

btn\_03.setMargin(new Insets(0, 0, 0, 0));

btn\_03.setBackground(Color.green);

btn\_03.setBorder(null);

btn\_03.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 0;

yInput = 3;

enableKey();

enableColor();

}

});

p1.add(btn\_03);

btn\_04 = new JButton(pink);

btn\_04.setMargin(new Insets(0, 0, 0, 0));

btn\_04.setBackground(Color.green);

btn\_04.setBorder(null);

btn\_04.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 0;

yInput = 4;

enableKey();

enableColor();

}

});

p1.add(btn\_04);

btn\_05 = new JButton(pink);

btn\_05.setMargin(new Insets(0, 0, 0, 0));

btn\_05.setBackground(Color.green);

btn\_05.setBorder(null);

btn\_05.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 0;

yInput = 5;

enableColor();

enableKey();

}

});

p1.add(btn\_05);

btn\_06 = new JButton(pink);

btn\_06.setMargin(new Insets(0, 0, 0, 0));

btn\_06.setBackground(Color.green);

btn\_06.setBorder(null);

btn\_06.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 0;yInput = 6;enableColor();

enableKey();

}

});

p1.add(btn\_06);

btn\_07 = new JButton(pink);

btn\_07.setMargin(new Insets(0, 0, 0, 0));

btn\_07.setBackground(Color.green);

btn\_07.setBorder(null);

btn\_07.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 0;yInput = 7;enableColor();

enableKey();

}

});

p1.add(btn\_07);

btn\_10 = new JButton(pink);

btn\_10.setMargin(new Insets(0, 0, 0, 0));

btn\_10.setBackground(Color.green);

btn\_10.setBorder(null);

btn\_10.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 1;yInput = 0;enableColor();

enableKey();

}

});

p1.add(btn\_10);

btn\_11 = new JButton(pink);

btn\_11.setMargin(new Insets(0, 0, 0, 0));

btn\_11.setBackground(Color.green);

btn\_11.setBorder(null);

btn\_11.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 1;xInput = 1;enableColor();

enableKey();

}

});

p1.add(btn\_11);

btn\_12 = new JButton(pink);

btn\_12.setMargin(new Insets(0, 0, 0, 0));

btn\_12.setBackground(Color.green);

btn\_12.setBorder(null);

btn\_12.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 1;yInput = 2;enableColor();

enableKey();

}

});

p1.add(btn\_12);

btn\_13 = new JButton(pink);

btn\_13.setMargin(new Insets(0, 0, 0, 0));

btn\_13.setBackground(Color.green);

btn\_13.setBorder(null);

btn\_13.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 1;yInput = 3;enableColor();

enableKey();

}

});

p1.add(btn\_13);

btn\_14 = new JButton(pink);

btn\_14.setMargin(new Insets(0, 0, 0, 0));

btn\_14.setBackground(Color.green);

btn\_14.setBorder(null);

btn\_14.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 1;yInput = 4;enableColor();

enableKey();

}

});

p1.add(btn\_14);

btn\_15 = new JButton(pink);

btn\_15.setMargin(new Insets(0, 0, 0, 0));

btn\_15.setBackground(Color.green);

btn\_15.setBorder(null);

btn\_15.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 1;yInput = 5;enableColor();

enableKey();

}

});

p1.add(btn\_15);

btn\_16 = new JButton(pink);

btn\_16.setMargin(new Insets(0, 0, 0, 0));

btn\_16.setBackground(Color.green);

btn\_16.setBorder(null);

btn\_16.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 1;yInput = 6;enableColor();

enableKey();

}

});

p1.add(btn\_16);

btn\_17 = new JButton(pink);

btn\_17.setMargin(new Insets(0, 0, 0, 0));

btn\_17.setBackground(Color.green);

btn\_17.setBorder(null);

btn\_17.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 1;yInput = 7;enableColor();

enableKey();

}

});

p1.add(btn\_17);

btn\_20= new JButton(pink);

btn\_20.setMargin(new Insets(0, 0, 0, 0));

btn\_20.setBackground(Color.green);

btn\_20.setBorder(null);

btn\_20.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 2;yInput = 0;enableColor();

enableKey();

}

});

p1.add(btn\_20);

btn\_21= new JButton(pink);

btn\_21.setMargin(new Insets(0, 0, 0, 0));

btn\_21.setBackground(Color.green);

btn\_21.setBorder(null);

btn\_21.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 2;yInput = 1;enableColor();

enableKey();

}

});

p1.add(btn\_21);

btn\_22= new JButton(pink);

btn\_22.setMargin(new Insets(0, 0, 0, 0));

btn\_22.setBackground(Color.green);

btn\_22.setBorder(null);

btn\_22.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 2;yInput = 2;enableColor();

enableKey();

}

});

p1.add(btn\_22);

btn\_23= new JButton(pink);

btn\_23.setMargin(new Insets(0, 0, 0, 0));

btn\_23.setBackground(Color.green);

btn\_23.setBorder(null);

btn\_23.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 2;yInput = 3;enableColor();

enableKey();

}

});

p1.add(btn\_23);

btn\_24= new JButton(pink);

btn\_24.setMargin(new Insets(0, 0, 0, 0));

btn\_24.setBackground(Color.green);

btn\_24.setBorder(null);

btn\_24.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 2;yInput = 4;enableColor();

enableKey();

}

});

p1.add(btn\_24);

btn\_25= new JButton(pink);

btn\_25.setMargin(new Insets(0, 0, 0, 0));

btn\_25.setBackground(Color.green);

btn\_25.setBorder(null);

btn\_25.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 2;yInput = 5;enableColor();

enableKey();

}

});

p1.add(btn\_25);

btn\_26= new JButton(pink);

btn\_26.setMargin(new Insets(0, 0, 0, 0));

btn\_26.setBackground(Color.green);

btn\_26.setBorder(null);

btn\_26.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 2;yInput = 6;enableColor();

enableKey();

}

});

p1.add(btn\_26);

btn\_27= new JButton(pink);

btn\_27.setMargin(new Insets(0, 0, 0, 0));

btn\_27.setBackground(Color.green);

btn\_27.setBorder(null);

btn\_27.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 2;yInput = 7;enableColor();

enableKey();

}

});

p1.add(btn\_27);

btn\_30= new JButton(pink);

btn\_30.setMargin(new Insets(0, 0, 0, 0));

btn\_30.setBackground(Color.green);

btn\_30.setBorder(null);

btn\_30.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 3;yInput = 0;enableColor();

enableKey();

}

});

p1.add(btn\_30);

btn\_31= new JButton(pink);

btn\_31.setMargin(new Insets(0, 0, 0, 0));

btn\_31.setBackground(Color.green);

btn\_31.setBorder(null);

btn\_31.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 3;yInput = 1;enableColor();

enableKey();

}

});

p1.add(btn\_31);

btn\_32= new JButton(pink);

btn\_32.setMargin(new Insets(0, 0, 0, 0));

btn\_32.setBackground(Color.green);

btn\_32.setBorder(null);

btn\_32.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 3;yInput = 2;enableColor();

enableKey();

}

});

p1.add(btn\_32);

btn\_33= new JButton(pink);

btn\_33.setMargin(new Insets(0, 0, 0, 0));

btn\_33.setBackground(Color.green);

btn\_33.setBorder(null);

btn\_33.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 3;yInput = 3;enableColor();

enableKey();

}

});

p1.add(btn\_33);

btn\_34= new JButton(pink);

btn\_34.setMargin(new Insets(0, 0, 0, 0));

btn\_34.setBackground(Color.green);

btn\_34.setBorder(null);

btn\_34.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 3;yInput = 4;enableColor();

enableKey();

}

});

p1.add(btn\_34);

btn\_35= new JButton(pink);

btn\_35.setMargin(new Insets(0, 0, 0, 0));

btn\_35.setBackground(Color.green);

btn\_35.setBorder(null);

btn\_35.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 3;yInput = 5;enableColor();

enableKey();

}

});

p1.add(btn\_35);

btn\_36= new JButton(pink);

btn\_36.setMargin(new Insets(0, 0, 0, 0));

btn\_36.setBackground(Color.green);

btn\_36.setBorder(null);

btn\_36.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 3;yInput = 6;enableColor();

enableKey();

}

});

p1.add(btn\_36);

btn\_37= new JButton(pink);

btn\_37.setMargin(new Insets(0, 0, 0, 0));

btn\_37.setBackground(Color.green);

btn\_37.setBorder(null);

btn\_37.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 3;yInput = 7;enableColor();

enableKey();

}

});

p1.add(btn\_37);

btn\_40= new JButton(pink);

btn\_40.setMargin(new Insets(0, 0, 0, 0));

btn\_40.setBackground(Color.green);

btn\_40.setBorder(null);

btn\_40.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 4;yInput = 0;enableColor();

enableKey();

}

});

p1.add(btn\_40);

btn\_41= new JButton(pink);

btn\_41.setMargin(new Insets(0, 0, 0, 0));

btn\_41.setBackground(Color.green);

btn\_41.setBorder(null);

btn\_41.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 4;yInput = 1;enableColor();

enableKey();

}

});

p1.add(btn\_41);

btn\_42= new JButton(pink);

btn\_42.setMargin(new Insets(0, 0, 0, 0));

btn\_42.setBackground(Color.green);

btn\_42.setBorder(null);

btn\_42.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 4;yInput = 2;enableColor();

enableKey();

}

});

p1.add(btn\_42);

btn\_43= new JButton(pink);

btn\_43.setMargin(new Insets(0, 0, 0, 0));

btn\_43.setBackground(Color.green);

btn\_43.setBorder(null);

btn\_43.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 4;yInput = 3;enableColor();

enableKey();

}

});

p1.add(btn\_43);

btn\_44= new JButton(pink);

btn\_44.setMargin(new Insets(0, 0, 0, 0));

btn\_44.setBackground(Color.green);

btn\_44.setBorder(null);

btn\_44.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 4;yInput = 4;enableColor();

enableKey();

}

});

p1.add(btn\_44);

btn\_45= new JButton(pink);

btn\_45.setMargin(new Insets(0, 0, 0, 0));

btn\_45.setBackground(Color.green);

btn\_45.setBorder(null);

btn\_45.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 4;yInput = 5;enableColor();

enableKey();

}

});

p1.add(btn\_45);

btn\_46= new JButton(pink);

btn\_46.setMargin(new Insets(0, 0, 0, 0));

btn\_46.setBackground(Color.green);

btn\_46.setBorder(null);

btn\_46.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 4;yInput = 6;enableColor();

enableKey();

}

});

p1.add(btn\_46);

btn\_47= new JButton(pink);

btn\_47.setMargin(new Insets(0, 0, 0, 0));

btn\_47.setBackground(Color.green);

btn\_47.setBorder(null);

btn\_47.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 4;yInput = 7;enableColor();

enableKey();

}

});

p1.add(btn\_47);

btn\_50= new JButton(pink);

btn\_50.setMargin(new Insets(0, 0, 0, 0));

btn\_50.setBackground(Color.green);

btn\_50.setBorder(null);

btn\_50.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 5;yInput = 0;enableColor();

enableKey();

}

});

p1.add(btn\_50);

btn\_51= new JButton(pink);

btn\_51.setMargin(new Insets(0, 0, 0, 0));

btn\_51.setBackground(Color.green);

btn\_51.setBorder(null);

btn\_51.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 5;yInput = 1;enableColor();

enableKey();

}

});

p1.add(btn\_51);

btn\_52= new JButton(pink);

btn\_52.setMargin(new Insets(0, 0, 0, 0));

btn\_52.setBackground(Color.green);

btn\_52.setBorder(null);

btn\_52.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 5;yInput = 2;enableColor();

enableKey();

}

});

p1.add(btn\_52);

btn\_53= new JButton(pink);

btn\_53.setMargin(new Insets(0, 0, 0, 0));

btn\_53.setBackground(Color.green);

btn\_53.setBorder(null);

btn\_53.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 5;yInput = 3;enableColor();

enableKey();

}

});

p1.add(btn\_53);

btn\_54= new JButton(pink);

btn\_54.setMargin(new Insets(0, 0, 0, 0));

btn\_54.setBackground(Color.green);

btn\_54.setBorder(null);

btn\_54.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 5;yInput = 4;enableColor();

enableKey();

}

});

p1.add(btn\_54);

btn\_55= new JButton(pink);

btn\_55.setMargin(new Insets(0, 0, 0, 0));

btn\_55.setBackground(Color.green);

btn\_55.setBorder(null);

btn\_55.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 5;yInput = 5;enableColor();

enableKey();

}

});

p1.add(btn\_55);

btn\_56= new JButton(pink);

btn\_56.setMargin(new Insets(0, 0, 0, 0));

btn\_56.setBackground(Color.green);

btn\_56.setBorder(null);

btn\_56.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 5;yInput = 6;enableColor();

enableKey();

}

});

p1.add(btn\_56);

btn\_57= new JButton(pink);

btn\_57.setMargin(new Insets(0, 0, 0, 0));

btn\_57.setBackground(Color.green);

btn\_57.setBorder(null);

btn\_57.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 5;yInput = 7;enableColor();

enableKey();

}

});

p1.add(btn\_57);

btn\_60= new JButton(pink);

btn\_60.setMargin(new Insets(0, 0, 0, 0));

btn\_60.setBackground(Color.green);

btn\_60.setBorder(null);

btn\_60.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 6;yInput = 0;enableColor();

enableKey();

}

});

p1.add(btn\_60);

btn\_61= new JButton(pink);

btn\_61.setMargin(new Insets(0, 0, 0, 0));

btn\_61.setBackground(Color.green);

btn\_61.setBorder(null);

btn\_61.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 6;yInput = 1;enableColor();

enableKey();

}

});

p1.add(btn\_61);

btn\_62= new JButton(pink);

btn\_62.setMargin(new Insets(0, 0, 0, 0));

btn\_62.setBackground(Color.green);

btn\_62.setBorder(null);

btn\_62.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 6;yInput = 2;enableColor();

enableKey();

}

});

p1.add(btn\_62);

btn\_63= new JButton(pink);

btn\_63.setMargin(new Insets(0, 0, 0, 0));

btn\_63.setBackground(Color.green);

btn\_63.setBorder(null);

btn\_63.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 6;yInput = 3;enableColor();

enableKey();

}

});

p1.add(btn\_63);

btn\_64= new JButton(pink);

btn\_64.setMargin(new Insets(0, 0, 0, 0));

btn\_64.setBackground(Color.green);

btn\_64.setBorder(null);

btn\_64.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 6;yInput = 4;enableColor();

enableKey();

}

});

p1.add(btn\_64);

btn\_65= new JButton(pink);

btn\_65.setMargin(new Insets(0, 0, 0, 0));

btn\_65.setBackground(Color.green);

btn\_65.setBorder(null);

btn\_65.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 6;yInput = 5;enableColor();

enableKey();

}

});

p1.add(btn\_65);

btn\_66= new JButton(pink);

btn\_66.setMargin(new Insets(0, 0, 0, 0));

btn\_66.setBackground(Color.green);

btn\_66.setBorder(null);

btn\_66.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 6;yInput = 6;enableColor();

enableKey();

}

});

p1.add(btn\_66);

btn\_67= new JButton(pink);

btn\_67.setMargin(new Insets(0, 0, 0, 0));

btn\_67.setBackground(Color.green);

btn\_67.setBorder(null);

btn\_67.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 6;yInput = 7;enableColor();

enableKey();

}

});

p1.add(btn\_67);

btn\_70= new JButton(pink);

btn\_70.setMargin(new Insets(0, 0, 0, 0));

btn\_70.setBackground(Color.green);

btn\_70.setBorder(null);

btn\_70.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 7;yInput = 0;enableColor();

enableKey();

}

});

p1.add(btn\_70);

btn\_71= new JButton(pink);

btn\_71.setMargin(new Insets(0, 0, 0, 0));

btn\_71.setBackground(Color.green);

btn\_71.setBorder(null);

btn\_71.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 7;yInput = 1;enableColor();

enableKey();

}

});

p1.add(btn\_71);

btn\_72= new JButton(pink);

btn\_72.setMargin(new Insets(0, 0, 0, 0));

btn\_72.setBackground(Color.green);

btn\_72.setBorder(null);

btn\_72.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 7;yInput = 2;enableColor();

enableKey();

}

});

p1.add(btn\_72);

btn\_73= new JButton(pink);

btn\_73.setMargin(new Insets(0, 0, 0, 0));

btn\_73.setBackground(Color.green);

btn\_73.setBorder(null);

btn\_73.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 7;yInput = 3;enableColor();

enableKey();

}

});

p1.add(btn\_73);

btn\_74= new JButton(pink);

btn\_74.setMargin(new Insets(0, 0, 0, 0));

btn\_74.setBackground(Color.green);

btn\_74.setBorder(null);

btn\_74.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 7;yInput = 4;enableColor();

enableKey();

}

});

p1.add(btn\_74);

btn\_75= new JButton(pink);

btn\_75.setMargin(new Insets(0, 0, 0, 0));

btn\_75.setBackground(Color.green);

btn\_75.setBorder(null);

btn\_75.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 7;yInput = 5;enableColor();

enableKey();

}

});

p1.add(btn\_75);

btn\_76= new JButton(pink);

btn\_76.setMargin(new Insets(0, 0, 0, 0));

btn\_76.setBackground(Color.green);

btn\_76.setBorder(null);

btn\_76.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 7;yInput = 6;enableColor();

enableKey();

}

});

p1.add(btn\_76);

btn\_77= new JButton(pink);

btn\_77.setMargin(new Insets(0, 0, 0, 0));

btn\_77.setBackground(Color.green);

btn\_77.setBorder(null);

btn\_77.addActionListener(new ActionListener() {

public void actionPerformed( ActionEvent e) {

xInput = 7;yInput = 7;enableColor();

enableKey();

}

});

p1.add(btn\_77);

//panel.getContentPane().add(l1, BorderLayout.EAST);

//panel.getContentPane().add(l1, BorderLayout.EAST);

frame.getContentPane().add(panel, BorderLayout.NORTH);

frame.getContentPane().add(p1, BorderLayout.SOUTH);

frame.pack();

// frame.setVisible(true);

}

public void enableKey() {

if (Front.gameBoard[0][0] != 2) // Enabling and Disabling Key ---- start

{

btn\_00.setEnabled(false);

} else {

btn\_00.setEnabled(true);

}

if (Front.gameBoard[0][1] != 2) {

btn\_01.setEnabled(false);

} else {

btn\_01.setEnabled(true);

}

if (Front.gameBoard[0][2] != 2) {

btn\_02.setEnabled(false);

} else {

btn\_02.setEnabled(true);

}

if (Front.gameBoard[0][3] != 2) {

btn\_03.setEnabled(false);

} else {

btn\_03.setEnabled(true);

}

if (Front.gameBoard[0][4] != 2) {

btn\_04.setEnabled(false);

} else {

btn\_04.setEnabled(true);

}

if (Front.gameBoard[0][5] != 2) {

btn\_05.setEnabled(false);

} else {

btn\_05.setEnabled(true);

}

if (Front.gameBoard[0][6] != 2) {

btn\_06.setEnabled(false);

} else {

btn\_06.setEnabled(true);

}

if (Front.gameBoard[0][7] != 2) {

btn\_07.setEnabled(false);

} else {

btn\_07.setEnabled(true);

}

if (Front.gameBoard[1][0] != 2) {

btn\_10.setEnabled(false);

} else {

btn\_10.setEnabled(true);

}

if (Front.gameBoard[1][1] != 2) {

btn\_11.setEnabled(false);

} else {

btn\_11.setEnabled(true);

}

if (Front.gameBoard[1][2] != 2) {

btn\_12.setEnabled(false);

} else {

btn\_12.setEnabled(true);

}

if (Front.gameBoard[1][3] != 2) {

btn\_13.setEnabled(false);

} else {

btn\_13.setEnabled(true);

}

if (Front.gameBoard[1][4] != 2) {

btn\_14.setEnabled(false);

} else {

btn\_14.setEnabled(true);

}

if (Front.gameBoard[1][5] != 2) {

btn\_15.setEnabled(false);

} else {

btn\_15.setEnabled(true);

}

if (Front.gameBoard[1][6] != 2) {

btn\_16.setEnabled(false);

} else {

btn\_16.setEnabled(true);

}

if (Front.gameBoard[1][7] != 2) {

btn\_17.setEnabled(false);

} else {

btn\_17.setEnabled(true);

}

if (Front.gameBoard[2][0] != 2) {

btn\_20.setEnabled(false);

} else {

btn\_20.setEnabled(true);

}

if (Front.gameBoard[2][1] != 2) {

btn\_21.setEnabled(false);

} else {

btn\_21.setEnabled(true);

}

if (Front.gameBoard[2][2] != 2) {

btn\_22.setEnabled(false);

} else {

btn\_22.setEnabled(true);

}

if (Front.gameBoard[2][3] != 2) {

btn\_23.setEnabled(false);

} else {

btn\_23.setEnabled(true);

}

if (Front.gameBoard[2][4] != 2) {

btn\_24.setEnabled(false);

} else {

btn\_24.setEnabled(true);

}

if (Front.gameBoard[2][5] != 2) {

btn\_25.setEnabled(false);

} else {

btn\_25.setEnabled(true);

}

if (Front.gameBoard[2][6] != 2) {

btn\_26.setEnabled(false);

} else {

btn\_26.setEnabled(true);

}

if (Front.gameBoard[2][7] != 2) {

btn\_27.setEnabled(false);

} else {

btn\_27.setEnabled(true);

}

if (Front.gameBoard[3][0] != 2) {

btn\_30.setEnabled(false);

} else {

btn\_30.setEnabled(true);

}

if (Front.gameBoard[3][1] != 2) {

btn\_31.setEnabled(false);

} else {

btn\_31.setEnabled(true);

}

if (Front.gameBoard[3][2] != 2) {

btn\_32.setEnabled(false);

} else {

btn\_32.setEnabled(true);

}

if (Front.gameBoard[3][3] != 2) {

btn\_33.setEnabled(false);

} else {

btn\_33.setEnabled(true);

}

if (Front.gameBoard[3][4] != 2) {

btn\_34.setEnabled(false);

} else {

btn\_34.setEnabled(true);

}

if (Front.gameBoard[3][5] != 2) {

btn\_35.setEnabled(false);

} else {

btn\_35.setEnabled(true);

}

if (Front.gameBoard[3][6] != 2) {

btn\_36.setEnabled(false);

} else {

btn\_36.setEnabled(true);

}

if (Front.gameBoard[3][7] != 2) {

btn\_37.setEnabled(false);

} else {

btn\_37.setEnabled(true);

}

if (Front.gameBoard[4][0] != 2) {

btn\_40.setEnabled(false);

} else {

btn\_40.setEnabled(true);

}

if (Front.gameBoard[4][1] != 2) {

btn\_41.setEnabled(false);

} else {

btn\_41.setEnabled(true);

}

if (Front.gameBoard[4][2] != 2) {

btn\_42.setEnabled(false);

} else {

btn\_42.setEnabled(true);

}

if (Front.gameBoard[4][3] != 2) {

btn\_43.setEnabled(false);

} else {

btn\_43.setEnabled(true);

}

if (Front.gameBoard[4][4] != 2) {

btn\_44.setEnabled(false);

} else {

btn\_44.setEnabled(true);

}

if (Front.gameBoard[4][5] != 2) {

btn\_45.setEnabled(false);

} else {

btn\_45.setEnabled(true);

}

if (Front.gameBoard[4][6] != 2) {

btn\_46.setEnabled(false);

} else {

btn\_46.setEnabled(true);

}

if (Front.gameBoard[4][7] != 2) {

btn\_47.setEnabled(false);

} else {

btn\_47.setEnabled(true);

}

if (Front.gameBoard[5][0] != 2) {

btn\_50.setEnabled(false);

} else {

btn\_50.setEnabled(true);

}

if (Front.gameBoard[5][1] != 2) {

btn\_51.setEnabled(false);

} else {

btn\_51.setEnabled(true);

}

if (Front.gameBoard[5][2] != 2) {

btn\_52.setEnabled(false);

} else {

btn\_52.setEnabled(true);

}

if (Front.gameBoard[5][3] != 2) {

btn\_53.setEnabled(false);

} else {

btn\_53.setEnabled(true);

}

if (Front.gameBoard[5][4] != 2) {

btn\_54.setEnabled(false);

} else {

btn\_54.setEnabled(true);

}

if (Front.gameBoard[5][5] != 2) {

btn\_55.setEnabled(false);

} else {

btn\_55.setEnabled(true);

}

if (Front.gameBoard[5][6] != 2) {

btn\_56.setEnabled(false);

} else {

btn\_56.setEnabled(true);

}

if (Front.gameBoard[5][7] != 2) {

btn\_57.setEnabled(false);

} else {

btn\_57.setEnabled(true);

}

if (Front.gameBoard[6][0] != 2) {

btn\_60.setEnabled(false);

} else {

btn\_60.setEnabled(true);

}

if (Front.gameBoard[6][1] != 2) {

btn\_61.setEnabled(false);

} else {

btn\_61.setEnabled(true);

}

if (Front.gameBoard[6][2] != 2) {

btn\_62.setEnabled(false);

} else {

btn\_62.setEnabled(true);

}

if (Front.gameBoard[6][3] != 2) {

btn\_63.setEnabled(false);

} else {

btn\_63.setEnabled(true);

}

if (Front.gameBoard[6][4] != 2) {

btn\_64.setEnabled(false);

} else {

btn\_64.setEnabled(true);

}

if (Front.gameBoard[6][5] != 2) {

btn\_65.setEnabled(false);

} else {

btn\_65.setEnabled(true);

}

if (Front.gameBoard[6][6] != 2) {

btn\_66.setEnabled(false);

} else {

btn\_66.setEnabled(true);

}

if (Front.gameBoard[6][7] != 2) {

btn\_67.setEnabled(false);

} else {

btn\_67.setEnabled(true);

}

if (Front.gameBoard[7][0] != 2) {

btn\_70.setEnabled(false);

} else {

btn\_70.setEnabled(true);

}

if (Front.gameBoard[7][1] != 2) {

btn\_71.setEnabled(false);

} else {

btn\_71.setEnabled(true);

}

if (Front.gameBoard[7][2] != 2) {

btn\_72.setEnabled(false);

} else {

btn\_72.setEnabled(true);

}

if (Front.gameBoard[7][3] != 2) {

btn\_73.setEnabled(false);

} else {

btn\_73.setEnabled(true);

}

if (Front.gameBoard[7][4] != 2) {

btn\_74.setEnabled(false);

} else {

btn\_74.setEnabled(true);

}

if (Front.gameBoard[7][5] != 2) {

btn\_75.setEnabled(false);

} else {

btn\_75.setEnabled(true);

}

if (Front.gameBoard[7][6] != 2) {

btn\_76.setEnabled(false);

} else {

btn\_76.setEnabled(true);

}

if (Front.gameBoard[7][7] != 2) {

btn\_77.setEnabled(false);

} else {

btn\_77.setEnabled(true);

}

} // Enabling and Disabling Key ---- end

public void enableColor() // Changing colour of keys ------ start

{

if(Front.gameBoard[0][0] == 0)

{

btn\_00.setBackground(Color.WHITE);

btn\_00.setIcon(black);

}

else if(Front.gameBoard[0][0] == 1)

{

btn\_00.setBackground(Color.BLACK);

btn\_00.setIcon(white);

}

if(Front.gameBoard[0][1] == 0)

{

btn\_01.setBackground(Color.WHITE);

btn\_01.setIcon(black);

}

else if(Front.gameBoard[0][1] == 1)

{

btn\_01.setBackground(Color.BLACK);

btn\_00.setIcon(white);

}

if(Front.gameBoard[0][2] == 0)

{

btn\_02.setBackground(Color.WHITE);

btn\_02.setIcon(black);

}

else if(Front.gameBoard[0][2] == 1)

{

btn\_02.setBackground(Color.BLACK);

btn\_02.setIcon(white);

}

if(Front.gameBoard[0][3] == 0)

{

btn\_03.setBackground(Color.WHITE);

btn\_03.setIcon(black);

}

else if(Front.gameBoard[0][3] == 1)

{

btn\_03.setBackground(Color.BLACK);

btn\_03.setIcon(white);

}

if(Front.gameBoard[0][4] == 0)

{

btn\_04.setBackground(Color.WHITE);

btn\_04.setIcon(black);

}

else if(Front.gameBoard[0][4] == 1)

{

btn\_04.setBackground(Color.BLACK);

btn\_04.setIcon(white);

}

if(Front.gameBoard[0][5] == 0)

{

btn\_05.setBackground(Color.WHITE);

btn\_05.setIcon(black);

}

else if(Front.gameBoard[0][5] == 1)

{

btn\_05.setBackground(Color.BLACK);

btn\_05.setIcon(white);

}

if(Front.gameBoard[0][6] == 0)

{

btn\_06.setBackground(Color.WHITE);

btn\_06.setIcon(black);

}

else if(Front.gameBoard[0][6] == 1)

{

btn\_06.setBackground(Color.BLACK);

btn\_06.setIcon(white);

}

if(Front.gameBoard[0][7] == 0)

{

btn\_07.setBackground(Color.WHITE);

btn\_07.setIcon(black);

}

else if(Front.gameBoard[0][7] == 1)

{

btn\_07.setBackground(Color.BLACK);

btn\_07.setIcon(white);

}

if(Front.gameBoard[1][0] == 0) // Enabling Disabling Key start point

{

btn\_10.setBackground(Color.WHITE);

btn\_10.setIcon(black);

}

else if(Front.gameBoard[1][0] == 1)

{

btn\_10.setBackground(Color.BLACK);

btn\_10.setIcon(white);

}

if(Front.gameBoard[1][1] == 0)

{

btn\_11.setBackground(Color.WHITE);

btn\_11.setIcon(black);

}

else if(Front.gameBoard[1][1] == 1)

{

btn\_11.setBackground(Color.BLACK);

btn\_11.setIcon(white);

}

if(Front.gameBoard[1][2] == 0)

{

btn\_12.setBackground(Color.WHITE);

btn\_12.setIcon(black);

}

else if(Front.gameBoard[1][2] == 1)

{

btn\_12.setBackground(Color.BLACK);

btn\_12.setIcon(white);

}

if(Front.gameBoard[1][3] == 0)

{

btn\_13.setBackground(Color.WHITE);

btn\_13.setIcon(black);

}

else if(Front.gameBoard[1][3] == 1)

{

btn\_13.setBackground(Color.BLACK);

btn\_13.setIcon(white);

}

if(Front.gameBoard[1][4] == 0)

{

btn\_14.setBackground(Color.WHITE);

btn\_14.setIcon(black);

}

else if(Front.gameBoard[1][4] == 1)

{

btn\_14.setBackground(Color.BLACK);

btn\_14.setIcon(white);

}

if(Front.gameBoard[1][5] == 0)

{

btn\_15.setBackground(Color.WHITE);

btn\_15.setIcon(black);

}

else if(Front.gameBoard[1][5] == 1)

{

btn\_15.setBackground(Color.BLACK);

btn\_15.setIcon(white);

}

if(Front.gameBoard[1][6] == 0)

{

btn\_16.setBackground(Color.WHITE);

btn\_16.setIcon(black);

}

else if(Front.gameBoard[1][6] == 1)

{

btn\_16.setBackground(Color.BLACK);

btn\_16.setIcon(white);

}

if(Front.gameBoard[1][7] == 0)

{

btn\_17.setBackground(Color.WHITE);

btn\_17.setIcon(black);

}

else if(Front.gameBoard[1][7] == 1)

{

btn\_17.setBackground(Color.BLACK);

btn\_17.setIcon(white);

}

if(Front.gameBoard[2][0] == 0)

{

btn\_20.setBackground(Color.WHITE);

btn\_20.setIcon(black);

}

else if(Front.gameBoard[2][0] == 1)

{

btn\_20.setBackground(Color.BLACK);

btn\_20.setIcon(white);

}

if(Front.gameBoard[2][1] == 0)

{

btn\_21.setBackground(Color.WHITE);

btn\_21.setIcon(black);

}

else if(Front.gameBoard[2][1] == 1)

{

btn\_21.setBackground(Color.BLACK);

btn\_21.setIcon(white);

}

if(Front.gameBoard[2][2] == 0)

{

btn\_22.setBackground(Color.WHITE);

btn\_22.setIcon(black);

}

else if(Front.gameBoard[2][2] == 1)

{

btn\_22.setBackground(Color.BLACK);

btn\_22.setIcon(white);

}

if(Front.gameBoard[2][3] == 0)

{

btn\_23.setBackground(Color.WHITE);

btn\_23.setIcon(black);

}

else if(Front.gameBoard[2][3] == 1)

{

btn\_23.setBackground(Color.BLACK);

btn\_23.setIcon(white);

}

if(Front.gameBoard[2][4] == 0)

{

btn\_24.setBackground(Color.WHITE);

btn\_24.setIcon(black);

}

else if(Front.gameBoard[2][4] == 1)

{

btn\_24.setBackground(Color.BLACK);

btn\_24.setIcon(white);

}

if(Front.gameBoard[2][5] == 0)

{

btn\_25.setBackground(Color.WHITE);

btn\_25.setIcon(black);

}

else if(Front.gameBoard[2][5] == 1)

{

btn\_25.setBackground(Color.BLACK);

btn\_25.setIcon(white);

}

if(Front.gameBoard[2][6] == 0)

{

btn\_26.setBackground(Color.WHITE);

btn\_26.setIcon(black);

}

else if(Front.gameBoard[2][6] == 1)

{

btn\_26.setBackground(Color.BLACK);

btn\_26.setIcon(white);

}

if(Front.gameBoard[2][7] == 0)

{

btn\_27.setBackground(Color.WHITE);

btn\_27.setIcon(black);

}

else if(Front.gameBoard[2][7] == 1)

{

btn\_27.setBackground(Color.BLACK);

btn\_27.setIcon(white);

}

if(Front.gameBoard[3][0] == 0)

{

btn\_30.setBackground(Color.WHITE);

btn\_30.setIcon(black);

}

else if(Front.gameBoard[3][0] == 1)

{

btn\_30.setBackground(Color.BLACK);

btn\_30.setIcon(white);

}

if(Front.gameBoard[3][1] == 0)

{

btn\_31.setBackground(Color.WHITE);

btn\_31.setIcon(black);

}

else if(Front.gameBoard[3][1] == 1)

{

btn\_31.setBackground(Color.BLACK);

btn\_31.setIcon(white);

}

if(Front.gameBoard[3][2] == 0)

{

btn\_32.setBackground(Color.WHITE);

btn\_32.setIcon(black);

}

else if(Front.gameBoard[3][2] == 1)

{

btn\_32.setBackground(Color.BLACK);

btn\_32.setIcon(white);

}

if(Front.gameBoard[3][3] == 0)

{

//btn\_33.setIcon(new ImageIcon("white.png"));

btn\_33.setIcon(black);

btn\_33.setBackground(Color.white);

}

else if(Front.gameBoard[3][3] == 1)

{

btn\_33.setIcon(black);

btn\_33.setBackground(Color.WHITE);

//btn\_33.setIcon(new ImageIcon("white.png"));

}

if(Front.gameBoard[3][4] == 0)

{

btn\_34.setBackground(Color.WHITE);

btn\_34.setIcon(black);

}

else if(Front.gameBoard[3][4] == 1)

{

btn\_34.setBackground(Color.BLACK);

btn\_34.setIcon(white);

}

if(Front.gameBoard[3][5] == 0)

{

btn\_35.setBackground(Color.WHITE);

btn\_35.setIcon(black);

}

else if(Front.gameBoard[3][5] == 1)

{

btn\_35.setBackground(Color.BLACK);

btn\_35.setIcon(white);

}

if(Front.gameBoard[3][6] == 0)

{

btn\_36.setBackground(Color.WHITE);

btn\_36.setIcon(black);

}

else if(Front.gameBoard[3][6] == 1)

{

btn\_36.setBackground(Color.BLACK);

btn\_36.setIcon(white);

}

if(Front.gameBoard[3][7] == 0)

{

btn\_37.setBackground(Color.WHITE);

btn\_37.setIcon(black);

}

else if(Front.gameBoard[3][7] == 1)

{

btn\_37.setBackground(Color.BLACK);

btn\_37.setIcon(white);

}

if(Front.gameBoard[4][0] == 0)

{

btn\_40.setBackground(Color.WHITE);

btn\_40.setIcon(black);

}

else if(Front.gameBoard[4][0] == 1)

{

btn\_40.setBackground(Color.BLACK);

btn\_40.setIcon(white);

}

if(Front.gameBoard[4][1] == 0)

{

btn\_41.setBackground(Color.WHITE);

btn\_41.setIcon(black);

}

else if(Front.gameBoard[4][1] == 1)

{

btn\_41.setBackground(Color.BLACK);

btn\_41.setIcon(white);

}

if(Front.gameBoard[4][2] == 0)

{

btn\_42.setBackground(Color.WHITE);

btn\_42.setIcon(black);

}

else if(Front.gameBoard[4][2] == 1)

{

btn\_42.setBackground(Color.BLACK);

btn\_42.setIcon(white);

}

if(Front.gameBoard[4][3] == 0)

{

btn\_43.setBackground(Color.WHITE);

btn\_43.setIcon(black);

}

else if(Front.gameBoard[4][3] == 1)

{

btn\_43.setBackground(Color.BLACK);

btn\_43.setIcon(white);

}

if(Front.gameBoard[4][4] == 0)

{

btn\_44.setBackground(Color.WHITE);

btn\_44.setIcon(black);

}

else if(Front.gameBoard[4][4] == 1)

{

btn\_44.setBackground(Color.BLACK);

btn\_44.setIcon(white);

}

if(Front.gameBoard[4][5] == 0)

{

btn\_45.setBackground(Color.WHITE);

btn\_45.setIcon(black);

}

else if(Front.gameBoard[4][5] == 1)

{

btn\_45.setBackground(Color.BLACK);

btn\_45.setIcon(white);

}

if(Front.gameBoard[4][6] == 0)

{

btn\_46.setBackground(Color.WHITE);

btn\_46.setIcon(black);

}

else if(Front.gameBoard[4][6] == 1)

{

btn\_46.setBackground(Color.BLACK);

btn\_46.setIcon(white);

}

if(Front.gameBoard[4][7] == 0)

{

btn\_47.setBackground(Color.WHITE);

btn\_47.setIcon(black);

}

else if(Front.gameBoard[4][7] == 1)

{

btn\_47.setBackground(Color.BLACK);

btn\_47.setIcon(white);

}

if(Front.gameBoard[5][0] == 0)

{

btn\_50.setBackground(Color.WHITE);

btn\_50.setIcon(black);

}

else if(Front.gameBoard[5][0] == 1)

{

btn\_50.setBackground(Color.BLACK);

btn\_50.setIcon(white);

}

if(Front.gameBoard[5][1] == 0)

{

btn\_51.setBackground(Color.WHITE);

btn\_51.setIcon(black);

}

else if(Front.gameBoard[5][1] == 1)

{

btn\_51.setBackground(Color.BLACK);

btn\_51.setIcon(white);

}

if(Front.gameBoard[5][2] == 0)

{

btn\_52.setBackground(Color.WHITE);

btn\_52.setIcon(black);

}

else if(Front.gameBoard[5][2] == 1)

{

btn\_52.setBackground(Color.BLACK);

btn\_52.setIcon(white);

}

if(Front.gameBoard[5][3] == 0)

{

btn\_53.setBackground(Color.WHITE);

btn\_53.setIcon(black);

}

else if(Front.gameBoard[5][3] == 1)

{

btn\_53.setBackground(Color.BLACK);

btn\_53.setIcon(white);

}

if(Front.gameBoard[5][4] == 0)

{

btn\_54.setBackground(Color.WHITE);

btn\_54.setIcon(black);

}

else if(Front.gameBoard[5][4] == 1)

{

btn\_54.setBackground(Color.BLACK);

btn\_54.setIcon(white);

}

if(Front.gameBoard[5][5] == 0)

{

btn\_55.setBackground(Color.WHITE);

btn\_55.setIcon(black);

}

else if(Front.gameBoard[5][5] == 1)

{

btn\_55.setBackground(Color.BLACK);

btn\_55.setIcon(white);

}

if(Front.gameBoard[5][6] == 0)

{

btn\_56.setBackground(Color.WHITE);

btn\_56.setIcon(black);

}

else if(Front.gameBoard[5][6] == 1)

{

btn\_56.setBackground(Color.BLACK);

btn\_56.setIcon(white);

}

if(Front.gameBoard[5][7] == 0)

{

btn\_57.setBackground(Color.WHITE);

btn\_57.setIcon(black);

}

else if(Front.gameBoard[5][7] == 1)

{

btn\_57.setBackground(Color.BLACK);

btn\_57.setIcon(white);

}

if(Front.gameBoard[6][0] == 0)

{

btn\_60.setBackground(Color.WHITE);

btn\_60.setIcon(black);

}

else if(Front.gameBoard[6][0] == 1)

{

btn\_60.setBackground(Color.BLACK);

btn\_60.setIcon(white);

}

if(Front.gameBoard[6][1] == 0)

{

btn\_61.setBackground(Color.WHITE);

btn\_61.setIcon(black);

}

else if(Front.gameBoard[6][1] == 1)

{

btn\_61.setBackground(Color.BLACK);

btn\_61.setIcon(white);

}

if(Front.gameBoard[6][2] == 0)

{

btn\_62.setBackground(Color.WHITE);

btn\_62.setIcon(black);

}

else if(Front.gameBoard[6][2] == 1)

{

btn\_62.setBackground(Color.BLACK);

btn\_62.setIcon(white);

}

if(Front.gameBoard[6][3] == 0)

{

btn\_63.setBackground(Color.WHITE);

btn\_63.setIcon(black);

}

else if(Front.gameBoard[6][3] == 1)

{

btn\_63.setBackground(Color.BLACK);

btn\_63.setIcon(white);

}

if(Front.gameBoard[6][4] == 0)

{

btn\_64.setBackground(Color.WHITE);

btn\_64.setIcon(black);

}

else if(Front.gameBoard[6][4] == 1)

{

btn\_64.setBackground(Color.BLACK);

btn\_64.setIcon(white);

}

if(Front.gameBoard[6][5] == 0)

{

btn\_65.setBackground(Color.WHITE);

btn\_65.setIcon(black);

}

else if(Front.gameBoard[6][5] == 1)

{

btn\_65.setBackground(Color.BLACK);

btn\_65.setIcon(white);

}

if(Front.gameBoard[6][6] == 0)

{

btn\_66.setBackground(Color.WHITE);

btn\_66.setIcon(black);

}

else if(Front.gameBoard[6][6] == 1)

{

btn\_66.setBackground(Color.BLACK);

btn\_66.setIcon(white);

}

if(Front.gameBoard[6][7] == 0)

{

btn\_67.setBackground(Color.WHITE);

btn\_67.setIcon(black);

}

else if(Front.gameBoard[6][7] == 1)

{

btn\_67.setBackground(Color.BLACK);

btn\_67.setIcon(white);

}

if(Front.gameBoard[7][0] == 0)

{

btn\_70.setBackground(Color.WHITE);

btn\_70.setIcon(black);

}

else if(Front.gameBoard[7][0] == 1)

{

btn\_70.setBackground(Color.BLACK);

btn\_70.setIcon(white);

}

if(Front.gameBoard[7][1] == 0)

{

btn\_71.setBackground(Color.WHITE);

btn\_71.setIcon(black);

}

else if(Front.gameBoard[7][1] == 1)

{

btn\_71.setBackground(Color.BLACK);

btn\_71.setIcon(white);

}

if(Front.gameBoard[7][2] == 0)

{

btn\_72.setBackground(Color.WHITE);

btn\_72.setIcon(black);

}

else if(Front.gameBoard[7][2] == 1)

{

btn\_72.setBackground(Color.BLACK);

btn\_72.setIcon(white);

}

if(Front.gameBoard[7][3] == 0)

{

btn\_73.setBackground(Color.WHITE);

btn\_73.setIcon(black);

}

else if(Front.gameBoard[7][3] == 1)

{

btn\_73.setBackground(Color.BLACK);

btn\_73.setIcon(white);

}

if(Front.gameBoard[7][4] == 0)

{

btn\_74.setBackground(Color.WHITE);

btn\_74.setIcon(black);

}

else if(Front.gameBoard[7][4] == 1)

{

btn\_74.setBackground(Color.BLACK);

btn\_74.setIcon(white);

}

if(Front.gameBoard[7][5] == 0)

{

btn\_75.setBackground(Color.WHITE);

btn\_75.setIcon(black);

}

else if(Front.gameBoard[7][5] == 1)

{

btn\_75.setBackground(Color.BLACK);

btn\_75.setIcon(white);

}

if(Front.gameBoard[7][6] == 0)

{

btn\_76.setBackground(Color.WHITE);

btn\_76.setIcon(black);

}

else if(Front.gameBoard[7][6] == 1)

{

btn\_76.setBackground(Color.BLACK);

btn\_76.setIcon(white);

}

if(Front.gameBoard[7][7] == 0)

{

btn\_77.setBackground(Color.WHITE);

btn\_77.setIcon(black);

}

else if(Front.gameBoard[7][7] == 1)

{

btn\_77.setBackground(Color.BLACK);

btn\_77.setIcon(white);

}

} // Changing colour of keys ------ end

}

###############################################################################################################################

**Backend code**

Front.java

**import** java.util.\*;

**public** **class** Front {

**int** gameBoard[][];

**public** Front() {

gameBoard = **new** **int**[8][8];

init();

}

**void** init() {

**for**(**int** i = 0 ; i < 8 ; i++)

{

**for**(**int** j = 0 ; j < 8 ; j++)

{

gameBoard[i][j] = 9;

}

}

gameBoard[3][3] = 0;

gameBoard[3][4] = 1;

gameBoard[4][3] = 1;

gameBoard[4][4] = 0;

}

**void** print() {

System.***out***.println(" " + "0 1 2 3 4 5 6 7 \n");

//System.out.println("------------------------");

**for**(**int** i = 0 ; i < 8 ; i++) {

System.***out***.print(i + " ");

**for**(**int** j = 0; j < 8; j++) {

**if**(gameBoard[i][j] == 9)

System.***out***.print(" ");

**else** **if**(gameBoard[i][j] == 2)

System.***out***.print("? ");

**else**

System.***out***.print(gameBoard[i][j]+" ");

}

System.***out***.println("");

//System.out.println("--------------------------");

}

System.***out***.println("\n\n");

}

**void** clearSuggestion(**int** [][]gameBoard){

**for**(**int** i =0 ; i < 8 ; i++)

{

**for**(**int** j = 0; j < 8 ; j++)

{

**if**(gameBoard[i][j] == 2)

gameBoard[i][j] = 9;

}

}

}

**void** getUserInput(**int** turn)

{

**int** x,y;

Scanner in = **new** Scanner(System.***in***);

**if**(turn == 1)

System.***out***.println("Enter the position for black dice");

**else**

System.***out***.println("Enter the position for white dice");

**for**(**int** i = 0; i < 8; i++)

{

**for**(**int** j = 0 ; j < 8 ; j++)

{

**if**(gameBoard[i][j] == 2)

{

System.***out***.println("i = "+ i + " j= "+ j);

}

}

}

x = in.nextInt();

y = in.nextInt();

//System.out.println(x+y);

insertPosition(x,y,turn,gameBoard);

//in.close();

}

**void** insertPosition(**int** x , **int** y , **int** turn ,**int** [][] gameBoard)

{

**if**(turn == 1)

{

**if**((x >=0 && x < 8) && (y >=0 && y < 8) && gameBoard[x][y] == 2)

{

gameBoard[x][y] = 1;

//left top diagonal

**int** i = x-1;

**int** j = y-1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 0)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 1)

{

flag = 1;

**break**;

}

i1--;

j1--;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 1)

{

gameBoard[i][j] = 1;

i--;

j--;

}

}

}

//top

i = x-1;

j = y;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 0)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 1)

{

flag = 1;

**break**;

}

i1--;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 1)

{

gameBoard[i][j] = 1;

i--;

}

}

}

//top right diagonal

i = x-1;

j = y+1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 0)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 1)

{

flag = 1;

**break**;

}

i1--;

j1++;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 1)

{

gameBoard[i][j] = 1;

i--;

j++;

}

}

}

//left

i = x;

j = y-1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 0)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 1)

{

flag = 1;

**break**;

}

j1--;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 1)

{

gameBoard[i][j] = 1;

j--;

}

}

}

//right

i = x;

j = y+1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 0)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 1)

{

flag = 1;

**break**;

}

j1++;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 1)

{

gameBoard[i][j] = 1;

j++;

}

}

}

//left buttom diagonal

i = x+1;

j = y-1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 0)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 1)

{

flag = 1;

**break**;

}

i1++;

j1--;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 1)

{

gameBoard[i][j] = 1;

i++;

j--;

}

}

}

//down

i = x+1;

j = y;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 0)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 1)

{

flag = 1;

**break**;

}

i1++;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 1)

{

gameBoard[i][j] = 1;

i++;

}

}

}

//right buttom diagonal

i = x+1;

j = y+1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 0)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 1)

{

flag = 1;

**break**;

}

i1++;

j1++;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 1)

{

gameBoard[i][j] = 1;

i++;

j++;

}

}

}

}

**else**

{

System.***out***.println("Invalid input enter again");

getUserInput(turn);

}

}

**else**

{

**if**((x >=0 && x < 8) && (y >=0 && y < 8) && gameBoard[x][y] == 2)

{

gameBoard[x][y] = 0;

//left top diagonal

**int** i = x-1;

**int** j = y-1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 1)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 0)

{

flag = 1;

**break**;

}

i1--;

j1--;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 0)

{

gameBoard[i][j] = 0;

i--;

j--;

}

}

}

//top

i = x-1;

j = y;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 1)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 0)

{

flag = 1;

**break**;

}

i1--;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 0)

{

gameBoard[i][j] = 0;

i--;

}

}

}

//top right diagonal

i = x-1;

j = y+1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 1)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 0)

{

flag = 1;

**break**;

}

i1--;

j1++;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 0)

{

gameBoard[i][j] = 0;

i--;

j++;

}

}

}

//left

i = x;

j = y-1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 1)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 0)

{

flag = 1;

**break**;

}

j1--;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 0)

{

gameBoard[i][j] = 0;

j--;

}

}

}

//right

i = x;

j = y+1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 1)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 0)

{

flag = 1;

**break**;

}

j1++;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 0)

{

gameBoard[i][j] = 0;

j++;

}

}

}

//left buttom diagonal

i = x+1;

j = y-1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 1)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 0)

{

flag = 1;

**break**;

}

i1++;

j1--;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 0)

{

gameBoard[i][j] = 0;

i++;

j--;

}

}

}

//down

i = x+1;

j = y;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 1)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 0)

{

flag = 1;

**break**;

}

i1++;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 0)

{

gameBoard[i][j] = 0;

i++;

}

}

}

//right buttom diagonal

i = x+1;

j = y+1;

**if**((i >=0 && i < 8) && (j >=0 && j < 8) && gameBoard[i][j] == 1)

{

**int** flag = 0;

**int** i1 = i;

**int** j1 = j;

**while**((i1 >=0 && i1 < 8) && (j1 >=0 && j1 < 8))

{

**if**(gameBoard[i1][j1] == 9 || gameBoard[i1][j1] == 2)

{

**break**;

}

**if**(gameBoard[i1][j1] == 0)

{

flag = 1;

**break**;

}

i1++;

j1++;

}

**if**(flag == 1)

{

**while**(gameBoard[i][j] != 0)

{

gameBoard[i][j] = 0;

i++;

j++;

}

}

}

}

**else**

{

System.***out***.println("Invalid input enter again");

getUserInput(turn);

}

}

}

**int** countScore()

{

**int** black = 0;

**int** white = 0;

**for**(**int** i = 0 ; i < 8 ;i++)

{

**for**(**int** j =0 ; j <8 ; j++)

{

**if**(gameBoard[i][j] == 1)

black++;

**if**(gameBoard[i][j] == 0)

white++;

}

}

System.***out***.println("SCORES: \n" + "white = "+ white + " black = "+ black);

**if**(black == 0 || white == 0)

**return** 1;

**return** 0;

}

**int** countScore1(**int** gameBoard[][])

{

**int** black = 0;

**int** white = 0;

**for**(**int** i = 0 ; i < 8 ;i++)

{

**for**(**int** j =0 ; j <8 ; j++)

{

**if**(gameBoard[i][j] == 1)

black++;

**if**(gameBoard[i][j] == 0)

white++;

}

}

//System.out.println("SCORES: \n" + "white = "+ white + " black = "+ black);

**if**(black == 0 || white == 0)

**return** 1;

**return** 0;

}

**int** checkEnd()

{

**for**(**int** i = 0 ; i < 8 ; i++)

{

**for**(**int** j = 0 ; j < 8 ; j++)

{

**if**(gameBoard[i][j] == 2 || gameBoard[i][j] == 9)

**return** 0;

}

}

**return** 1;

}

**int** checkEnd1(**int** gameBoard[][])

{

**for**(**int** i = 0 ; i < 8 ; i++)

{

**for**(**int** j = 0 ; j < 8 ; j++)

{

**if**(gameBoard[i][j] == 2 || gameBoard[i][j] == 9)

**return** 0;

}

}

**return** 1;

}

**void** winner()

{

**int** white = 0;

**int** black = 0;

**for**(**int** i = 0; i < 8 ; i++)

{

**for**(**int** j = 0 ; j < 8 ; j++)

{

**if**(gameBoard[i][j] == 1)

black++;

**if**(gameBoard[i][j] == 0)

white++;

}

}

**if**(black == white)

System.***out***.println("Match is draw");

**else** **if**(black > white)

System.***out***.println("Black wins");

**else**

System.***out***.println("White wins");

}

/\*int countSuggestion()

{

int count = 0;

for(int i = 0 ; i < 8 ; i++)

{

for(int j =0 ; j < 8 ; j++)

{

if(gameBoard[i][j] == 2)

count++;

}

}

if(count == 0)

return 1;

return 0;

}\*/

**public** **static** **void** main(String args[]) {

//System.out.println("enter the choice \n 1. Human vs. Human : 1 \n 2. Human vs. Computer : 2 \n");

//Scanner x = new Scanner(System.in);

// int choice = x.nextInt();

**int** choice = 2;

**if**(choice == 1)

{

Front f1 = **new** Front();

//f1.print();

**int** turn = 1;

**int** t = 1;

**while**(**true**)

{

**if**(f1.checkEnd() == 1)

**break**;

**if**(f1.countScore() == 1)

**break**;

Suggestion s1 = **new** Suggestion();

f1.gameBoard = s1.getSuggestion(f1.gameBoard , turn);

f1.print();

**if**(s1.countSuggestion(f1.gameBoard) == 0)

{

f1.getUserInput(turn);

t = -t;

**if**(t == -1)

turn = 0;

**else**

turn = 1;

}

**else**

{

System.***out***.println(turn + " dont have move");

t = -t;

**if**(t == -1)

turn = 0;

**else**

turn = 1;

f1.gameBoard = s1.getSuggestion(f1.gameBoard , turn);

f1.print();

f1.getUserInput(turn);

t = -t;

**if**(t == -1)

turn = 0;

**else**

turn = 1;

}

f1.clearSuggestion(f1.gameBoard);

}

**int** get = f1.countScore();

f1.print();

f1.winner();

}//choice 1 closed

**else** **if**(choice == 2)

{

Front f1 = **new** Front();

f1.print();

**int** turn = 1;

**int** t = 1;

**while**(**true**)

{

**if**(f1.checkEnd() == 1)

**break**;

**if**(f1.countScore() == 1)

**break**;

Suggestion s1 = **new** Suggestion();

f1.gameBoard = s1.getSuggestion(f1.gameBoard , turn);

f1.print();

**if**(s1.countSuggestion(f1.gameBoard) == 0)

{

//minimax function call to print utility values

Minimax m1 = **new** Minimax();

**if**(turn == 0)

{

**int** val = m1.minimax(f1.gameBoard,0,**true**,9 , Integer.***MIN\_VALUE*** , Integer.***MAX\_VALUE***);

}

**if**(turn ==1)

f1.getUserInput(turn);

t = -t;

**if**(t == -1)

turn = 0;

**else**

turn = 1;

}

**else**

{

System.***out***.println(turn + " dont have move");

t = -t;

**if**(t == -1)

turn = 0;

**else**

turn = 1;

f1.gameBoard = s1.getSuggestion(f1.gameBoard , turn);

f1.print();

//minimax function call to print utility values

Minimax m1 = **new** Minimax();

**if**(turn ==0)

{

**int** val = m1.minimax(f1.gameBoard,0 , **true** , 9, Integer.***MIN\_VALUE*** , Integer.***MAX\_VALUE***);

}

**if**(turn == 1)

f1.getUserInput(turn);

t = -t;

**if**(t == -1)

turn = 0;

**else**

turn = 1;

}

f1.clearSuggestion(f1.gameBoard);

}

**int** get = f1.countScore();

f1.print();

f1.winner();

}//choice 2 closed

**else**

{

}

}

}

Suggestion.java

//Suggestions will be getting from this class methods

**public** **class** Suggestion

{

**int** countSuggestion(**int** gameBoard[][])

{

**int** count = 0;

**for**(**int** i = 0 ; i < 8 ; i++)

{

**for**(**int** j =0 ; j < 8 ; j++)

{

**if**(gameBoard[i][j] == 2)

count++;

}

}

**if**(count == 0)

**return** 1;

**return** 0;

}

**public** **int** [][] getSuggestion(**int** B[][] , **int** turn)//return array with suggestions

{

**int** search = 0;//intial move is for black , it will help in finding appropriate suggestion position near white

**if**(turn == 0)//if turn is of white

search = 1;//move is for white , it will help in finding appropriate suggestion position near black

**for**(**int** i = 0 ; i< 8 ; i++)//rows of game board

{

**for**(**int** j = 0 ; j < 8 ; j++) // column of game boared

{

**if**(B[i][j] == search)//it check weather to find the suggestion for white or black move

{

//for i-1, j-1 top left diagonal

**if**((i-1 >= 0 && i-1 < 8) &&(j-1 >= 0 && j-1 < 8))

{

**if**(B[i-1][j-1]== 9)//if the position is 9 then only it can be choosen for suggestion

{

**if**(check(i-1 , j-1 , B, turn))//we check if the position contains correct suggestion position means there is opposition die is there or not

{

B[i-1][j-1] = 2;

}

}

}

//for i-1 , j

**if**((i-1 >= 0 && i-1 < 8) &&(j >= 0 && j < 8))

{

**if**(B[i-1][j]== 9)

{

**if**(check(i-1 , j, B,turn))

{

B[i-1][j] = 2;

}

}

}

// for i-1, j+1

**if**((i-1 >= 0 && i-1 < 8) &&(j+1 >= 0 && j+1 < 8))

{

**if**(B[i-1][j+1]== 9)

{

**if**(check(i-1 , j+1, B,turn))

{

B[i-1][j+1] = 2;

}

}

}

//for i , j-1

**if**((i >= 0 && i < 8) &&(j-1 >= 0 && j-1 < 8))

{

**if**(B[i][j-1]== 9)

{

**if**(check(i, j-1, B,turn))

{

B[i][j-1] = 2;

}

}

}

// for i , j+1

**if**((i >= 0 && i < 8) &&(j+1 >= 0 && j+1 < 8))

{

**if**(B[i][j+1]== 9)

{

**if**(check(i , j+1, B,turn))

{

B[i][j+1] = 2;

}

}

}

//for i+1 , j-1

**if**((i+1 >= 0 && i+1 < 8) &&(j-1 >= 0 && j-1 < 8))

{

**if**(B[i+1][j-1]== 9)

{

**if**(check(i+1 , j-1, B,turn))

{

B[i+1][j-1] = 2;

}

}

}

//for i+1 , j

**if**((i+1 >= 0 && i+1 < 8) &&(j >= 0 && j < 8))

{

**if**(B[i+1][j]== 9)

{

**if**(check(i+1 , j, B,turn))

{

B[i+1][j] = 2;

}

}

}

//for i+1 , j+1

**if**((i+1 >= 0 && i+1 < 8) &&(j+1 >= 0 && j+1 < 8))

{

**if**(B[i+1][j+1]== 9)

{

**if**(check(i+1 , j+1, B,turn))

{

B[i+1][j+1] = 2;

}

}

}

}

}//for loop for column ends i.e. j loop

}//for loop for rows ends i.e i loop

//printnew(B);

**return** B;

}//get suggestion function ends

/\*void printnew(int[][] arr) {

for(int i = 0 ; i < 8 ; i++) {

for(int j = 0; j < 8; j++) {

System.out.print(arr[i][j]+" ");

}

System.out.println("");

}

}\*/

**boolean** check( **int** l , **int** m , **int** arr[][], **int** turn)

{

**int** white = 0;

**int** black = 1;

**if**(turn == 0)

{

**int** temp = white;

white = black;

black = temp;

}

**int** i = l-1;

**int** j = m-1;

**int** countZero = 0;

**while**(i >=0 && j >= 0) {

**if**(arr[i][j] == 9)

**break**;

**if**(arr[i][j] == 2)

**break**;

**if**(arr[i][j] == white)

countZero++;

**if**(arr[i][j] == black && countZero != 0)

{

**return** **true**;

}

**if**(arr[i][j] == black && countZero == 0)

**break**;

i--;

j--;

}

i = l+1;

j = m+1;

countZero = 0;

**while**(i < 8 && j < 8) {

**if**(arr[i][j] == 9)

**break**;

**if**(arr[i][j] == 2)

**break**;

**if**(arr[i][j] == white)

countZero++;

**if**(arr[i][j] == black && countZero != 0)

{

**return** **true**;

}

**if**(arr[i][j] == black && countZero == 0)

**break**;

i++;

j++;

}

i = l-1;

j = m+1;

countZero = 0;

**while**(i >= 0 && j < 8) {

**if**(arr[i][j] == 9)

**break**;

**if**(arr[i][j] == 2)

**break**;

**if**(arr[i][j] == white)

countZero++;

**if**(arr[i][j] == black && countZero != 0)

{

**return** **true**;

}

**if**(arr[i][j] == black && countZero == 0)

**break**;

i--;

j++;

}

i = l+1;

j = m-1;

countZero = 0;

**while**(j >= 0 && i < 8) {

**if**(arr[i][j] == 9)

**break**;

**if**(arr[i][j] == 2)

**break**;

**if**(arr[i][j] == white)

countZero++;

**if**(arr[i][j] == black && countZero != 0)

{

**return** **true**;

}

**if**(arr[i][j] == black && countZero == 0)

**break**;

j--;

i++;

}

i = l-1;

j = m;

countZero = 0;

**while**(i >= 0) {

**if**(arr[i][j] == 9)

**break**;

**if**(arr[i][j] == 2)

**break**;

**if**(arr[i][j] == white)

countZero++;

**if**(arr[i][j] == black && countZero != 0)

{

**return** **true**;

}

**if**(arr[i][j] == black && countZero == 0)

**break**;

i--;

}

i = l+1;

j = m;

countZero = 0;

**while**(i < 8) {

**if**(arr[i][j] == 9)

**break**;

**if**(arr[i][j] == 2)

**break**;

**if**(arr[i][j] == white)

countZero++;

**if**(arr[i][j] == black && countZero != 0)

{

**return** **true**;

}

**if**(arr[i][j] == black && countZero == 0)

**break**;

i++;

}

j = m-1;

i = l;

countZero = 0;

**while**(j >= 0) {

**if**(arr[i][j] == 9)

**break**;

**if**(arr[i][j] == 2)

**break**;

**if**(arr[i][j] == white)

countZero++;

**if**(arr[i][j] == black && countZero != 0)

{

**return** **true**;

}

**if**(arr[i][j] == black && countZero == 0)

**break**;

j--;

}

j = m+1;

i = l;

countZero = 0;

**while**(j < 8) {

**if**(arr[i][j] == 9)

**break**;

**if**(arr[i][j] == 2)

**break**;

**if**(arr[i][j] == white)

countZero++;

**if**(arr[i][j] == black && countZero != 0)

{

**return** **true**;

}

**if**(arr[i][j] == black && countZero == 0)

**break**;

j++;

}

**return** **false**;

}

}

Minimax.java

**public** **class** Minimax {

Front fobj = **new** Front();

Suggestion sobj = **new** Suggestion();

**int** [][] copyArray(**int** [][] gameBoard)

{

**int** arr[][] = **new** **int**[8][8];

**for**(**int** i = 0 ; i < 8 ; i++)

{

**for**(**int** j = 0 ; j < 8 ; j++)

{

arr[i][j] = gameBoard[i][j];

}

}

**return** arr;

}

**int** minimax(**int** gameBoard[][] , **int** depth , **boolean** isMax , **int** h , **int** alpha , **int** beta)

{

**if**(sobj.countSuggestion(gameBoard) == 1)

isMax = !isMax;

**if**(depth == h || fobj.checkEnd1(gameBoard) == 1 || fobj.countScore1(gameBoard) == 1 /\*|| sobj.countSuggestion(gameBoard) == 1\*/)//chek if depth match or its terminal state or gamme ends

{

Utility u = **new** Utility();

**int** value = u.utility(gameBoard);

**if**(sobj.countSuggestion(gameBoard) == 1)

value = value +50;

**return** value;

}

**if**(isMax)

{

**int** max = Integer.***MIN\_VALUE***;

**int** posi = -1 , posj = -1;

Front f1 = **new** Front();

Suggestion s1 = **new** Suggestion();

f1.clearSuggestion(gameBoard);

s1.getSuggestion(gameBoard, 0);

**for**(**int** i = 0 ; i < 8 ; i++)

{

**int** flag = 0;

**for**(**int** j = 0 ; j < 8 ; j++)

{

**if**(gameBoard[i][j] == 2)

{

**int** arr[][] = **new** **int**[8][8];

arr = copyArray(gameBoard);

f1.insertPosition(i, j,0, arr);

**int** value = minimax(arr , depth+1 , **false** , h , alpha , beta);

**if**(value > max)

{

max = value;

posi = i;

posj = j;

}

**if**(max > alpha)

{

alpha = max;

}

**if**(beta <= alpha)

{

//System.out.println("break atlevel = " + depth + " position " + posi + " , " + posj);

flag = 1;

**break**;

}

}//end of checking suggestion position

}//end of j loop

**if**(flag == 1)

**break**;

}//end of i loop

//System.out.println("maximum value return by the minimum player is " + max + " at depth = " + depth + "at position =" + posi + ", " + posj);

**if**(depth == 0)

{

//System.out.println("position are " + posi + " , " + posj + " max = " + max);

**if**(posi != -1 && posj != -1)

{

f1.insertPosition(posi, posj, 0, gameBoard);

}

**else**

{

**int** flag = 0;

**for**(**int** i = 0 ; i < 8 ; i++)

{

**for**(**int** j = 0 ; j < 8 ; j++)

{

**if**(gameBoard[i][j] == 2)

{

f1.insertPosition(i, j, 0, gameBoard);

flag = 1;

**break**;

}

}

**if**(flag == 1)

**break**;

}

}

}

**return** max;

}//end of if(isMax)

**else**

{

**int** min = Integer.***MAX\_VALUE***;

//int posi = -1 , posj = -1;

Front f1 = **new** Front();

Suggestion s1 = **new** Suggestion();

f1.clearSuggestion(gameBoard);

s1.getSuggestion(gameBoard, 1);

**for**(**int** i = 0 ; i < 8 ; i++)

{

**int** flag = 0;

**for**(**int** j = 0 ; j < 8 ; j++)

{

**if**(gameBoard[i][j] == 2)

{

**int** arr[][] = **new** **int**[8][8];

arr = copyArray(gameBoard);

f1.insertPosition(i, j,1, arr);

**int** value = minimax(arr , depth+1 , **true** , h , alpha , beta);

**if**(value < min)

{

min = value;

// posi = i;

// posj = j;

}

**if**(min < beta)

{

beta = min;

}

**if**(beta <= alpha)

{

//System.out.println("break atlevel = " + depth + " position " + posi + " , " + posj);

flag = 1;

**break**;

}

}//end of checking suggestion position

}//end of j loop

**if**(flag == 1)

**break**;

}//end of i loop

//System.out.println("minimum value return by the maximum player is " + min + " at depth = " + depth + "at position =" + posi + ", " + posj);

**return** min;

}// end of else part of if(isMax)

}//end minimax function

/\*void tryutility(int gameBoard[][])

{

int max = -1;

int posi = -1,posj = -1;

Front f1 = new Front();

for(int i = 0 ; i < 8 ;i++)

{

for(int j =0 ; j <8 ; j++)

{

if(gameBoard[i][j] == 2)

{

int arr[][] = new int[8][8];

arr = copyArray(gameBoard);

f1.insertPosition(i, j,0, arr);

f1.clearSuggestion(arr);

Suggestion s1 = new Suggestion();

Utility u1 = new Utility();

//if(turn == 0)

// turn = 1;

//else

// turn = 0;

arr = s1.getSuggestion(arr , 0);

int utilityValue = u1.getUtility(i, j, arr, 0);

System.out.println("Utility value for position " + i + "," + j + "=" + utilityValue);

if(utilityValue > max)

{

max= utilityValue;

posi = i;

posj = j;

}

}

}

}

System.out.println("maximum utility get by choosing position" + posi + " , " + posj + " = " + max);

f1.insertPosition(posi, posj, 0, gameBoard);

}\*/

}

Utility.java

//using greedy approach for evaluation

**public** **class** Utility {

**int** black;

**int** white;

**int** utility(**int** gameBoard[][])

{

countScore(gameBoard);

**return** white - black;

}

**int** getUtility(**int** i , **int** j , **int** gameBoard[][] , **int** turn)

{

//first check for the no move condition with and without corner and wall condition

countScore(gameBoard);

Suggestion s1 = **new** Suggestion();

//int score;

//if(turn == 1)//black turn

// score = obj.black - prev\_score\_black;

//else

//score = obj.white - prev\_score\_white;

**if**(s1.countSuggestion(gameBoard) == 1)//no move condition

{

**if**(checkCorner(i,j) == **true**)// check for corner case

{

**return** 12 + (white - black);

}

**if**(checkWall(i,j) == **true**)//check for wall condition

{

**return** 11+ (white - black);

}

**return** 10 + (white - black);//maximum profit is for no move condition

}

// corner cases condition

**if**(checkCorner(i,j) == **true**)

{

**return** 9+(white - black);

}

//wall position with maximum profit

//simple wall position

**if**(checkWall(i,j) == **true**)

{

**return** 8+(white - black);

}

// position with maximum profit

**return** (white - black); //you have to maintain a frontier for minimax algo

}

**boolean** checkCorner(**int** i , **int** j)

{

**if**(i == 0 && j == 0)

**return** **true**;

**else** **if**(i == 0 && j == 7)

**return** **true**;

**else** **if**(i == 7 && j == 0)

**return** **true**;

**else** **if**(i == 7 && j == 7)

**return** **true**;

**return** **false**;

}

**boolean** checkWall(**int** i , **int** j)

{

**if**(i == 0 || i == 7 || j == 0 || j == 7)

**return** **true**;

**return** **false**;

}

**void** countScore(**int** gameBoard[][])

{

**for**(**int** i = 0 ; i < 8 ;i++)

{

**for**(**int** j =0 ; j <8 ; j++)

{

**if**(gameBoard[i][j] == 1)

black++;

**if**(gameBoard[i][j] == 0)

white++;

}

}

//System.out.println("SCORES: \n" + "white = "+ white + " black = "+ black);

//if(black == 0 || white == 0)

//return 1;

//return 0;

}

}