VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI 590018

Project Report on

"Othello Game"

By

MONISHA S (1BM24CS174)
NALLURI SINDHUJA (1BM24CS179)
NANDHANA N (1BM24CS181)

Under the Guidance of

MONISHA H M
Assistant Professor, Department of CSE
BMS College of Engineering

Work carried out at

Department of Computer Science and Engineering
BMS College of Engineering
(Autonomous college under VTU)
P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019
2025-2026

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the OOPS with JAVA project titled "**Othello Game"** has been carried out by Monisha S (1BM24CS174), Nalluri Sindhuja (1BM24CS179), Nandhana N (1BM24CS181) during the academic year 2025-2026.

Signature of the guide
MONISHA H M
Assistant Professor,
Department of Computer Science and Engineering
BMS College of Engineering, Bangalore

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, MONISHA S (1BM24CS174), NALLURI SINDHUJA (1BM24CS179), NANDHANA N (1BM24CS181), students of 3rd Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this project work entitled "Othello Game" has been carried out by us under the guidance of MONISHA H M**,** Assistant Professor, Department of CSE, BMS College of Engineering, Bangalore during the academic semester Sep-Dec 2025. We also declare that to the best of our knowledge and belief, the project reported here is not from part of any other report by any other students.

Signature of the Candidates

MONISHA S (1BM24CS174)

NALLURI SINDHUJA (1BM24CS179)

NANDHANA N (1BM24CS181)

TABLE OF CONTENTS

PROBLEM STATEMENT

The objective of this project is to design and implement a console-based Othello (Reversi) game using Java by applying Object-Oriented Programming (OOP) concepts.

Othello is a two-player strategy board game played on a grid where players take turns placing their pieces. The goal is to capture the opponent's pieces by sandwiching them between the player's own pieces in horizontal, vertical, or diagonal directions.

Existing System:

- Manual board games require physical setup.
- No automatic validation of rules.
- No automatic score calculation.
- Error-prone and time-consuming.

Proposed System:

- A Java-based console game that:
    - Automatically validates moves
    - Supports diagonal and straight captures
    - Switches player turns automatically
    - Calculates scores
    - Declares the winner

Objective:

- To implement Othello using Java
- To demonstrate OOP concepts
- To provide an interactive and logical gameplay system

INTRODUCTION

Othello, also known as Reversi, is a popular board game that requires logical thinking and strategy. The game is played between two players on a square grid. Each player is assigned a color (Black or White), and the goal is to dominate the board by the end of the game.

In this project, the Othello game is implemented using Java with a 6×6 board to simplify gameplay. The game runs in the command-line interface, making it lightweight and easy to execute.

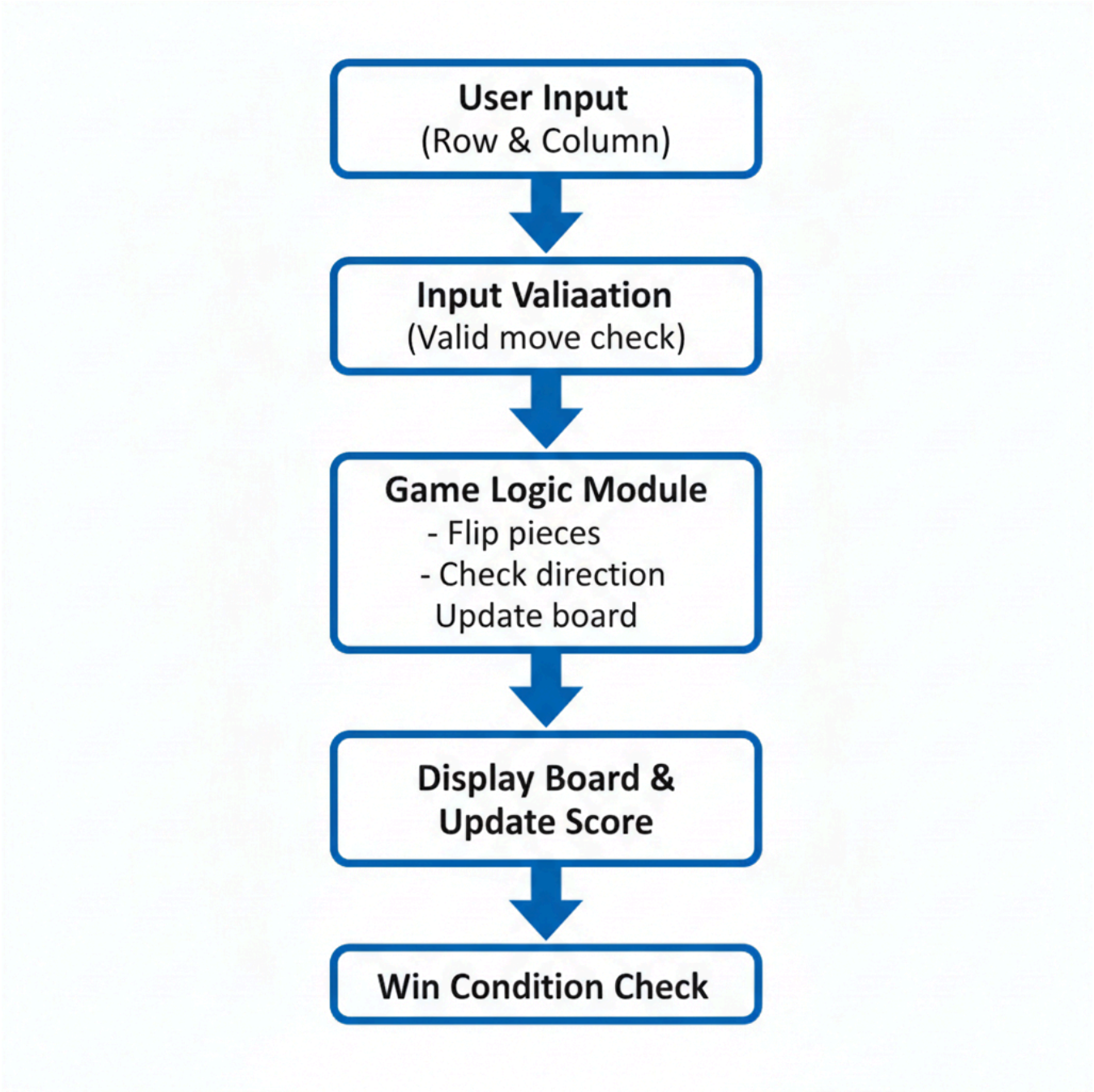
This project demonstrates:

- Proper use of Java programming
- Application of OOP principles
- Game logic implementation
- Input validation
- Decision-making using conditions and loops

The project is suitable for understanding real-world application of Java and object-oriented design.

OVERVIEW OF THE PROJECT

Block Diagram:



Working Explanation:

1. Players enter row and column.
2. The system checks whether the move is valid.
3. If valid, opponent pieces are flipped.
4. The board is updated and displayed.
5. Turn switches to the next player.
6. The game continues until no moves remain.
7. The winner is declared based on score.

TOOLS USED

Tool	Purpose
Java (JDK 17/21)	Core programming language
Command Prompt	Program execution
VS Code / Notepad++	Writing code
Windows OS	Execution environment
JDK Compiler (javac)	Compiling Java files

Why Java?

- Platform independent
- Supports OOP concepts
- Secure and reliable
- Widely used in industry

OOPs CONCEPTS USED

1. Class and Object

- Classes are used to represent game components.
- Objects are created to access game logic.

Example:

- OthelloGame
- Board
- Player

2. Encapsulation

- Variables are kept private.
- Access is controlled using methods.
- Ensures data security.

Example:

private char[][] board;

3. Abstraction

- Complex logic is hidden inside methods.
- The user only interacts with the game interface.

Example:

isValidMove()
makeMove()

4. Inheritance

- Helps reuse code and structure.
- Game classes inherit common behavior if extended.

5. Polymorphism

- The same method behaves differently depending on input.
- Used in move validation and display logic.

6. Control Structures

- Loops → for, while
- Conditions → if-else
- Used to manage turns and board updates

IMPLEMENTATION

Source Code:

```
import java.util.*;
public class Othello6 {

   static final int EMPTY = 0;
   static final int BLACK = 1; // Black moves first
   static final int WHITE = 2;

   static final int SIZE = 6; // 6x6 board

   // Directions (8)
   static final int[][] DIRS = {
      {-1,-1}, {-1,0}, {-1,1},
      {0,-1},         {0,1},
      {1,-1},  {1,0}, {1,1}
   };

   public static class Board {
      int[][] grid = new int[SIZE][SIZE];

      public Board() {
         initialize();
      }

      void initialize() {
         for (int i = 0; i < SIZE; i++)
            Arrays.fill(grid[i], EMPTY);
```

```
        // center start for 6x6: indices 2 and 3
        grid[2][2] = WHITE;
        grid[2][3] = BLACK;
        grid[3][2] = BLACK;
        grid[3][3] = WHITE;
    }

    boolean inBounds(int r, int c) {
        return r >= 0 && r < SIZE && c >= 0 && c < SIZE;
    }

    boolean isValidMove(int player, int r, int c) {
        if (!inBounds(r, c) || grid[r][c] != EMPTY) return false;
        int opp = opponent(player);

        for (int[] d : DIRS) {
            int i = r + d[0], j = c + d[1];
            boolean foundOpp = false;
            while (inBounds(i,j) && grid[i][j] == opp) {
                foundOpp = true;
                i += d[0]; j += d[1];
            }
            if (foundOpp && inBounds(i,j) && grid[i][j] == player) return true;
        }
        return false;
    }

    List<int[]> getValidMoves(int player) {
        List<int[]> moves = new ArrayList<>();
        for (int r = 0; r < SIZE; r++)
            for (int c = 0; c < SIZE; c++)
```

```
            if (isValidMove(player, r, c))
                moves.add(new int[]{r, c});
    return moves;
}

void applyMove(int player, int r, int c) {
    if (!isValidMove(player, r, c)) return;
    grid[r][c] = player;
    int opp = opponent(player);

    for (int[] d : DIRS) {
        int i = r + d[0], j = c + d[1];
        List<int[]> toFlip = new ArrayList<>();
        while (inBounds(i,j) && grid[i][j] == opp) {
            toFlip.add(new int[]{i,j});
            i += d[0]; j += d[1];
        }
        if (!toFlip.isEmpty() && inBounds(i,j) && grid[i][j] == player) {
            for (int[] p : toFlip) grid[p[0]][p[1]] = player;
        }
    }
}

boolean hasAnyValidMove(int player) {
    return !getValidMoves(player).isEmpty();
}

boolean isFull() {
    for (int r = 0; r < SIZE; r++)
        for (int c = 0; c < SIZE; c++)
            if (grid[r][c] == EMPTY) return false;
```

```
        return true;
    }

    boolean isGameOver() {
        return isFull() || (!hasAnyValidMove(BLACK) && !hasAnyValidMove(WHITE));
    }

    int[] score() {
        int b = 0, w = 0;
        for (int r = 0; r < SIZE; r++)
            for (int c = 0; c < SIZE; c++) {
                if (grid[r][c] == BLACK) b++;
                else if (grid[r][c] == WHITE) w++;
            }
        return new int[]{b,w};
    }

    void printBoard() {
        System.out.print("   ");
        for (int c = 0; c < SIZE; c++) System.out.print(c + " ");
        System.out.println();
        System.out.print("   ");
        for (int c = 0; c < SIZE; c++) System.out.print("--");
        System.out.println();
        for (int r = 0; r < SIZE; r++) {
            System.out.print(r + "| ");
            for (int c = 0; c < SIZE; c++) {
                if (grid[r][c] == BLACK) System.out.print("B ");
                else if (grid[r][c] == WHITE) System.out.print("W ");
                else System.out.print(". ");
            }
```

```
            System.out.println();
        }
        int[] s = score();
        System.out.println("\nScore -> Black: " + s[0] + "  White: " + s[1] + "\n");
    }

    int opponent(int p) {
        return p == BLACK ? WHITE : BLACK;
    }
} // end Board

// helper
static int opponent(int p) { return p == BLACK ? WHITE : BLACK; }

// MAIN: simple 2-player console interaction
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    Board board = new Board();
    int current = BLACK; // black moves first

    System.out.println("Welcome to 6x6 Othello (Reversi) — Console Version");
    System.out.println("Enter moves as: row col  (example: 2 3)\n");

    while (!board.isGameOver()) {
        board.printBoard();
        if (!board.hasAnyValidMove(current)) {
            System.out.println((current==BLACK? "Black":"White") + " has no valid moves. Turn
passed.");
            current = opponent(current);
            continue;
        }
```

```
        System.out.println("Current player: " + (current==BLACK? "Black (B)" : "White (W)"));
        List<int[]> moves = board.getValidMoves(current);
        System.out.print("Valid moves: ");
        for (int[] m : moves) System.out.print("[" + m[0] + "," + m[1] + "] ");
        System.out.println();

        System.out.print("Enter move (row col): ");
        int r, c;
        try {
            r = sc.nextInt();
            c = sc.nextInt();
        } catch (InputMismatchException ime) {
            System.out.println("Invalid input. Enter two integers like: 2 3");
            sc.nextLine(); // clear
            continue;
        }

        if (!board.inBounds(r,c)) {
            System.out.println("Out of bounds. Use 0 to " + (SIZE-1));
            continue;
        }
        if (!board.isValidMove(current, r, c)) {
            System.out.println("Invalid move. That position doesn't flip any opponent pieces.");
            continue;
        }

        board.applyMove(current, r, c);
        current = opponent(current);
    } // game loop
```

```
        // final state
        board.printBoard();
        int[] finalScore = board.score();
        System.out.println("Game Over!");
        System.out.println("Final Score -> Black: " + finalScore[0] + "  White: " + finalScore[1]);
        if (finalScore[0] > finalScore[1]) System.out.println("Black wins!");
        else if (finalScore[1] > finalScore[0]) System.out.println("White wins!");
        else System.out.println("It's a draw!");

        sc.close();
    }
}
```

RESULT

```
C:\Users\Sriraksha\OneDrive\Desktop>javac Othello6.java

C:\Users\Sriraksha\OneDrive\Desktop>java Othello6
Welcome to 6x6 Othello (Reversi) ? Console Version
Enter moves as: row col  (example: 2 3)

   0 1 2 3 4 5
   ------------
0| . . . . . .
1| . . . . . .
2| . . W B . .
3| . . B W . .
4| . . . . . .
5| . . . . . .

Score -> Black: 2  White: 2

Current player: Black (B)
Valid moves: [1,2] [2,1] [3,4] [4,3]
Enter move (row col): 3 4
   0 1 2 3 4 5
   ------------
0| . . . . . .
1| . . . . . .
2| . . W B . .
3| . . B B B .
4| . . . . . .
5| . . . . . .

Score -> Black: 4  White: 1
```

```
Current player: White (W)
Valid moves: [1,0] [1,2] [1,4] [2,4] [3,5]
Enter move (row col): 1 4
   0 1 2 3 4 5
   ------------
0| . . . . . .
1| . . . . W .
2| . B B W . .
3| . . W B B .
4| . . W . . .
5| . . . . . .

Score -> Black: 4  White: 4

Current player: Black (B)
Valid moves: [1,2] [1,3] [2,4] [3,1] [4,3] [5,1] [5,2]
Enter move (row col): 2 5
Invalid move. That position doesn't flip any opponent pieces.
   0 1 2 3 4 5
   ------------
0| . . . . . .
1| . . . . W .
2| . B B W . .
3| . . W B B .
4| . . W . . .
5| . . . . . .

Score -> Black: 4  White: 4
```

```
Current player: Black (B)
Valid moves: [0,4] [3,5]
Enter move (row col): 3 5
   0 1 2 3 4 5
   -----------
0| W W W B . B
1| W W W W B B
2| W B W W W B
3| W B W B B B
4| W B B W B B
5| W B B B B B

Score -> Black: 19  White: 16

Current player: White (W)
Valid moves: [0,4]
Enter move (row col): 0 4
   0 1 2 3 4 5
   -----------
0| W W W W W B
1| W W W W W B
2| W B W W W B
3| W B W B B B
4| W B B W B B
5| W B B B B B

Score -> Black: 17  White: 19

Game Over!
Final Score -> Black: 17  White: 19
White wins!
```

REFERENCES

- Java Documentation – Oracle
  https://docs.oracle.com/javase/
- Othello Game Rules
  https://www.worldothello.org
- Java Programming by Herbert Schildt
- Lecture notes provided by college
- Online references:
- GeeksforGeeks (Java concepts)
- Stack Overflow (logic clarification)