Faculty of Computers and Artificial Intelligence

Cairo University

CS213: Object Oriented Programming
Under Supervision of Dr. Mohammed El-Ramly

Assignment: 2 Section Group: 7 Task: 2 & 3 & 5

Board Games – Assignment 2 Report

Name	Malak Amr Ahmed	Abdelrhman Yasser Ali	Yassin Ahmed Ali
ID’s	20230416	20230220	20230465
Emails	malakamrahmdd@gmail.com	abdelrahmanyr05@gmail.com	yassinsawy@outlook.com
Phones	01156903124	01288217119	01033822107

Classes Description

4x4_XO.h

Class: FourByFour_XO_Board

Description: Represents a 4x4 tic-tac-toe board. Manages board updates, move validation, and win/draw conditions.

Attributes:

- int rows
- int columns
- T** board
- int n_moves

Methods:

- FourByFour_XO_Board()
- ~FourByFour_XO_Board()
- bool validate_token(int& x, int& y, T symbol)
- bool validate_move(int x, int y, T symbol)
- bool update_board(int x, int y, T symbol) override
- void delete_old(int x, int y)
- void display_board() override
- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

Class: FourByFour_XO_Player

Description: Represents a player who makes moves in the 4x4 tic-tac-toe game.

Attributes:

- string name
- T symbol
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

Class: FourByFour_XO_Random_Player

Description: Represents a random AI player for the 4x4 tic-tac-toe game.

Attributes:

- string name
- T symbol
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

5x5_XO.h

Class: FiveByFiveBoard

Description: Represents a 5x5 tic-tac-toe board. Counts sequences of three consecutive symbols and checks for win/draw conditions.

Attributes:

- int rows
- int columns
- T** board
- int n_moves
- int win

Methods:

- FiveByFiveBoard()
- ~FiveByFiveBoard()
- bool update_board(int x, int y, T symbol) override
- void display_board() override
- int countSequences(T player)
- int getMoves()
- void assign_move(int &x, int &y)
- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

Class: FiveByFivePlayer

Description: Represents a player in the 5x5 tic-tac-toe game.

Attributes:

- string name
- T symbol
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

Class: FiveByFiveRandomPlayer

Description: Represents a random AI player in the 5x5 tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

FourInRow.h

Class: FourInRow

Description: Represents a Connect Four game board with 6 rows and 7 columns.

Attributes:

- int rows
- int columns
- T** board
- int n_moves

Methods:

- FourInRow()
- ~FourInRow()
- bool update_board(int x, int y, T symbol) override
- void display_board() override
- bool is_win() override
- bool is_draw() override
- bool game_is_over() override
- void computer_move(T symbol)

Class: FourInRow_Player

Description: Represents a player in the Connect Four game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Class: FourInRow_Random_Player

Description: Represents a random AI player for the Connect Four game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Misere_XO.h

Class: Misere_XO

Description: Represents a 3x3 tic-tac-toe game where the objective is to force the opponent to complete a line.

Attributes:

- int rows
- int columns
- T** board
- int n_moves
- int win

Methods:

- Misere_XO()
- ~Misere_XO()
- bool update_board(int x, int y, T symbol) override
- void display_board() override
- bool is_win() override
- bool is_draw() override
- bool game_is_over() override
- bool getWin()

- int getMoves()
- void assign_move(int& x, int& y)

Class: Misere_XO_Player

Description: Represents a player in the Misere tic-tac-toe game.

Attributes:

- string name
- T symbol
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

Class: Misere_XO_Random_Player

Description: Represents a random AI player for the Misere tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Numerical_Tic.h

Class: NumericalTicTacToe

Description: Represents a numerical tic-tac-toe game where players use numbers to sum up to 15.

Attributes:

- int rows
- int columns
- T** board
- int n_moves

Methods:

- NumericalTicTacToe()
- ~NumericalTicTacToe()
- bool update_board(int x, int y, T symbol) override
- void display_board() override
- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

- bool validate_move(int x, int y, T symbol)

Class: NumericalTicTacToe_Player

Description: Represents a player who makes moves in the numerical tic-tac-toe game.

Attributes:

- string name
- T symbol
- set<int> used

Methods:

- void get_value(T &symbol)
- void getmove(int& x, int& y) override

Class: NumericalTicTacToe_Random_Player

Description: Represents a random AI player in the numerical tic-tac-toe game.

Attributes:

- string name
- T symbol
- set<int> used
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

pyramid_X_O.h

Class: Pyramid_X_O_Board

Description: Represents a pyramid-shaped tic-tac-toe board with 3 rows and a varying number of columns.

Attributes:

- int rows
- int columns
- T** board
- int n_moves

Methods:

- Pyramid_X_O_Board()
- ~Pyramid_X_O_Board()
- bool update_board(int x, int y, T symbol) override
- void display_board() override

- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

Class: Pyramid_X_O_Player

Description: Represents a player in the pyramid tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Class: Pyramid_X_O_Random_Player

Description: Represents a random AI player in the pyramid tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Ultimate_XO.h

Class: Ultimate_XO_Board

Description: Represents a 9x9 ultimate tic-tac-toe board consisting of smaller 3x3 boards.

Attributes:

- int rows
- int columns
- char** board
- int n_moves
- char small_board[3][3]

Methods:

- Ultimate_XO_Board()
- ~Ultimate_XO_Board()
- bool validate_move(int x, int y, char symbol)
- bool update_board(int x, int y, char symbol) override
- void display_board() override
- char check_win(int r, int c)

- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

Class: Ultimate_XO_Player

Description: Represents a player in the ultimate tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Class: Ultimate_XO_Random_Player

Description: Represents a random AI player in the ultimate tic-tac-toe game.

Attributes:

- string name
- T symbol
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

Word_TicTac.h

Class: Word_TicTac

Description: Represents a word-based 3x3 tic-tac-toe game where players form words from a dictionary.

Attributes:

- int rows
- int columns
- T** board
- int n_moves
- set<string> dic

Methods:

- Word_TicTac()
- ~Word_TicTac()
- bool update_board(int x, int y, T symbol) override
- void display_board() override

- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

Class: Word_TicTac_Player

Description: Represents a player in the word-based tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void get_value(T &symbol)
- void getmove(int& x, int& y) override

Class: Word_TicTac_Random_Player

Description: Represents a random AI player in the word-based tic-tac-toe game.

Attributes:

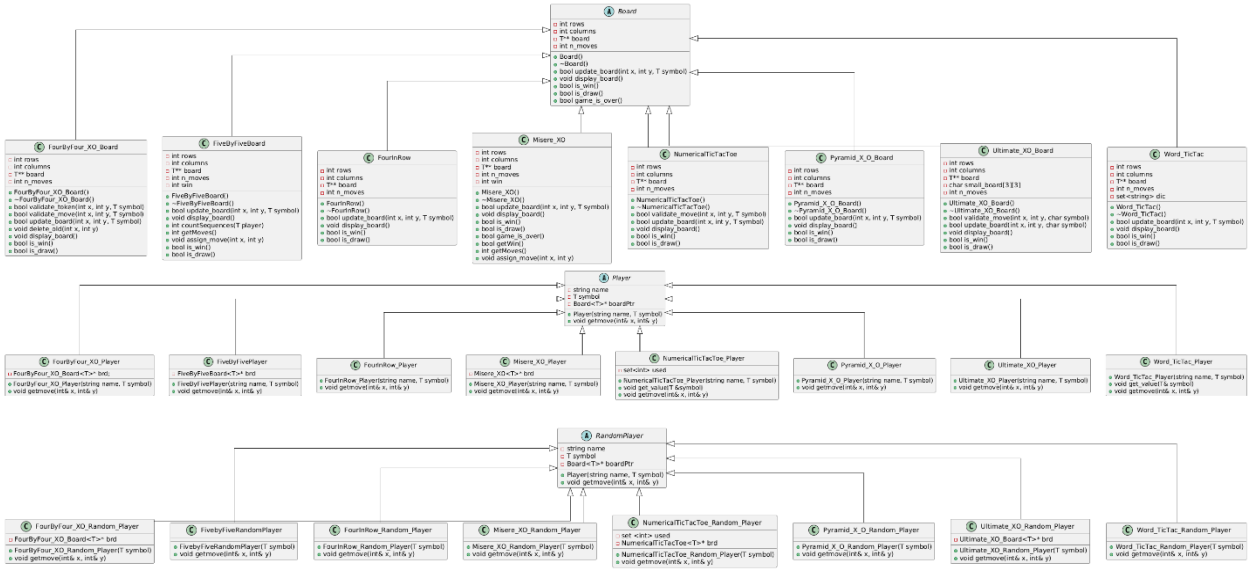
- string name
- T symbol

Methods:

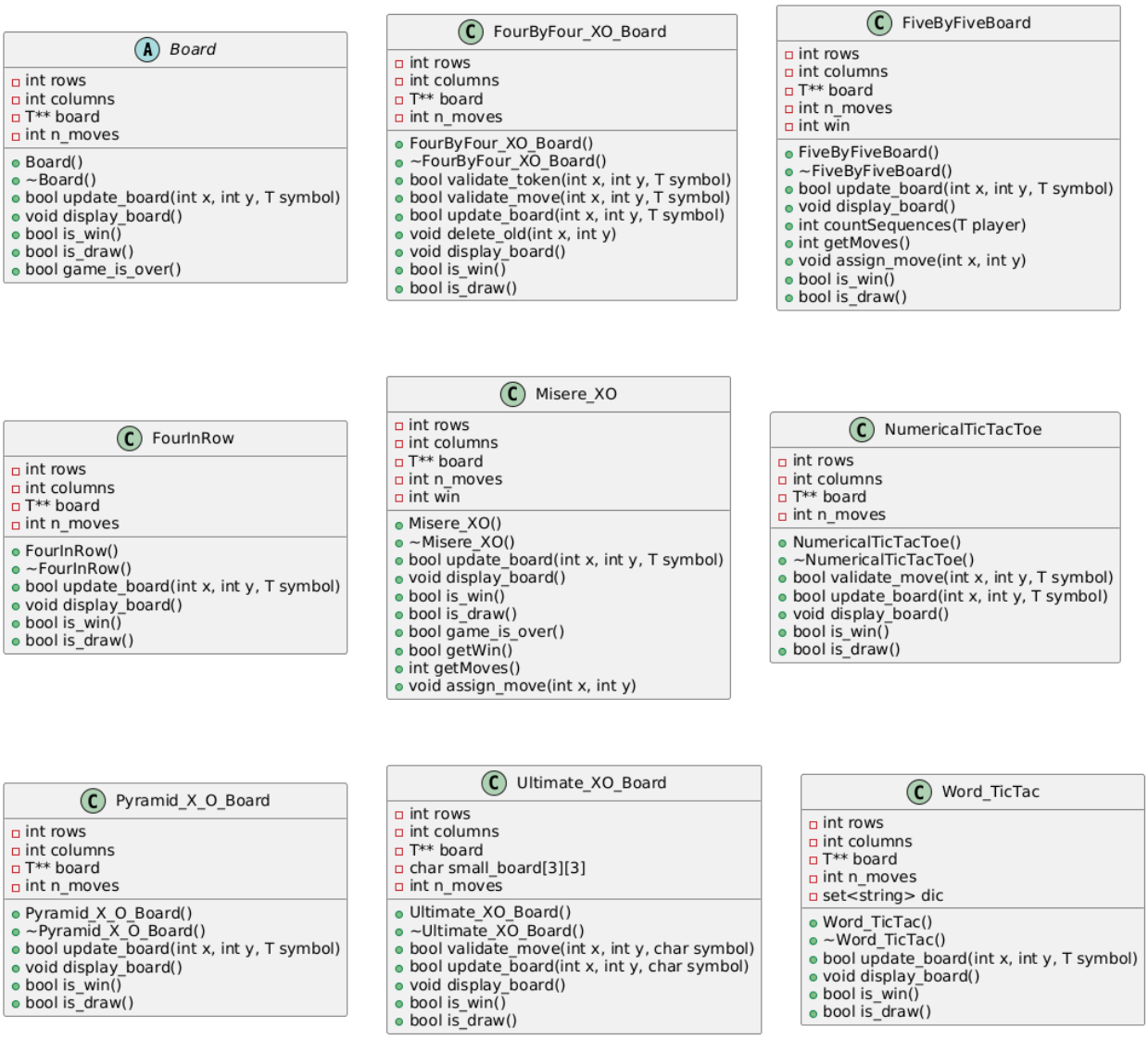
- void getmove(int& x, int& y) override

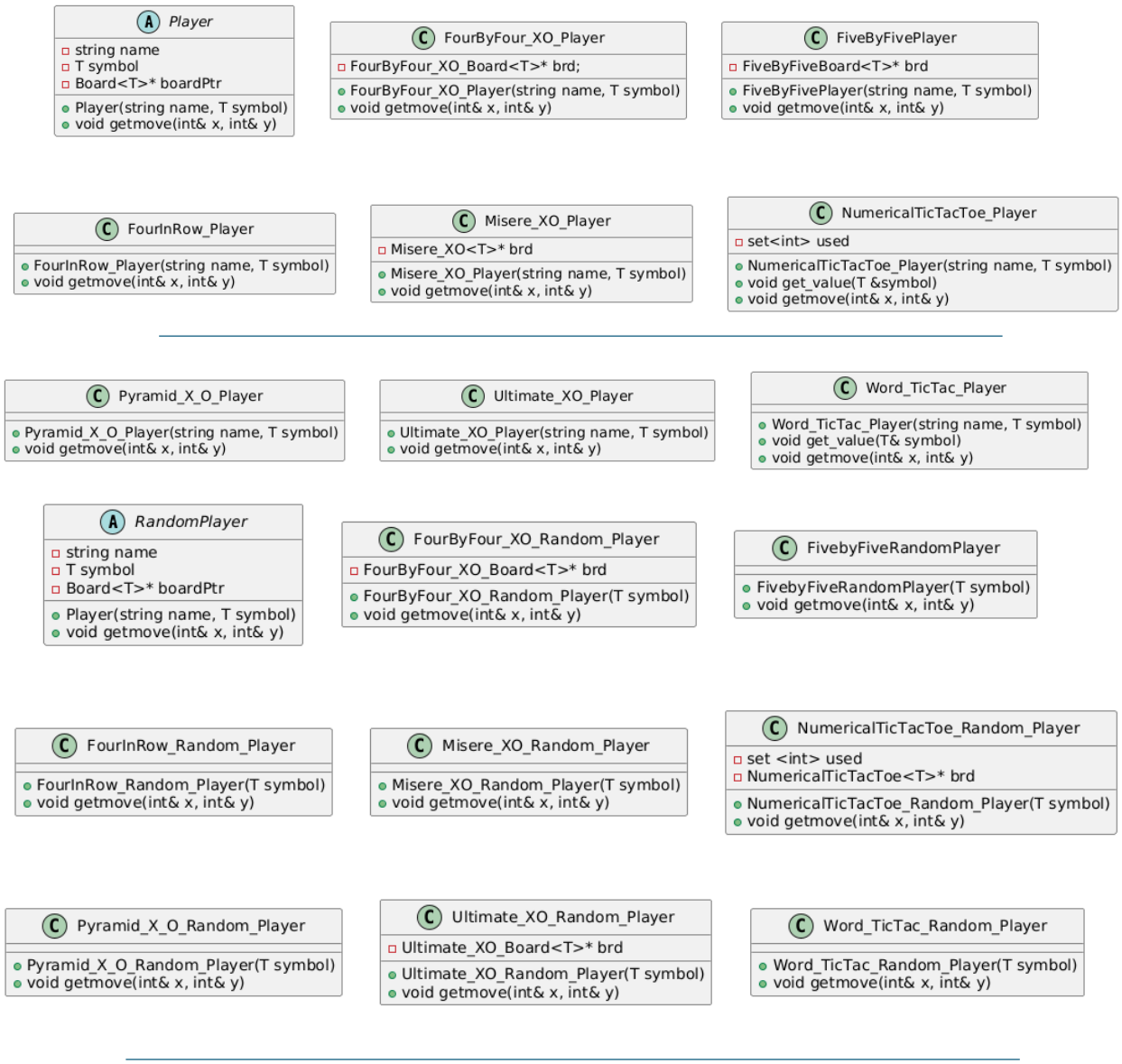
Classes UML Diagram

This is a generic class diagram for the games



The detailed diagram for each game is provided here:




Player
- string name
- T symbol
- Board<T>* boardPtr
+ Player(string name, T symbol)
+ void getmove(int& x, int& y)
FourByFour_XO_Player
- FourByFour_XO_Board<T>* brd;
+ FourByFour_XO_Player(string name, T symbol)
+ void getmove(int& x, int& y)
FiveByFivePlayer
- FiveByFiveBoard<T>* brd
+ FiveByFivePlayer(string name, T symbol)
+ void getmove(int& x, int& y)
FourInRow_Player
+ FourInRow_Player(string name, T symbol)
+ void getmove(int& x, int& y)
Misere_XO_Player
- Misere_XO<T>* brd
+ Misere_XO_Player(string name, T symbol)
+ void getmove(int& x, int& y)
NumericalTicTacToe_Player
- set<int> used
+ NumericalTicTacToe_Player(string name, T symbol)
+ void get_value(T &symbol)
+ void getmove(int& x, int& y)
Pyramid_X_O_Player
+ Pyramid_X_O_Player(string name, T symbol)
+ void getmove(int& x, int& y)
Ultimate_XO_Player
+ Ultimate_XO_Player(string name, T symbol)
+ void getmove(int& x, int& y)
Word_TicTac_Player
+ Word_TicTac_Player(string name, T symbol)
+ void get_value(T& symbol)
+ void getmove(int& x, int& y)
RandomPlayer
- string name
- T symbol
- Board<T>* boardPtr
+ Player(string name, T symbol)
+ void getmove(int& x, int& y)
FourByFour_XO_Random_Player
- FourByFour_XO_Board<T>* brd
+ FourByFour_XO_Random_Player(T symbol)
+ void getmove(int& x, int& y)
FivebyFiveRandomPlayer
+ FivebyFiveRandomPlayer(T symbol)
+ void getmove(int& x, int& y)
FourInRow_Random_Player
+ FourInRow_Random_Player(T symbol)
+ void getmove(int& x, int& y)
Misere_XO_Random_Player
+ Misere_XO_Random_Player(T symbol)
+ void getmove(int& x, int& y)
NumericalTicTacToe_Random_Player
- set <int> used
- NumericalTicTacToe<T>* brd
+ NumericalTicTacToe_Random_Player(T symbol)
+ void getmove(int& x, int& y)
Pyramid_X_O_Random_Player
+ Pyramid_X_O_Random_Player(T symbol)
+ void getmove(int& x, int& y)
Ultimate_XO_Random_Player
- Ultimate_XO_Board<T>* brd
+ Ultimate_XO_Random_Player(T symbol)
+ void getmove(int& x, int& y)
Word_TicTac_Random_Player
+ Word_TicTac_Random_Player(T symbol)
+ void getmove(int& x, int& y)

Work Breakdown

Basic Assignment

- Abdelrahman: Games 1 and 4
- Malak: Games 2 and 5
- Yassin: Games 3 and 6
- Teamwork: Games 7 and 8 and menu

Code Reviews

Yassin

- Found that *Word Tic-Tac-Toe* wasn't searching for valid words in normal vector efficiently so we used *set()* and *count()* that does binary search on the valid words.
- Used *srand()* to seed time so the random pattern becomes unpredictable in random player.

Abdelrahman

- Adjusted *win* attribute and *is_win()* functions in *5x5* and *Misere Tic-Tac-Toe* to make sure that the right player wins the game
- Added some checking in *4x4's RandomPlayer* so that it makes a valid moves and chooses a valid token.

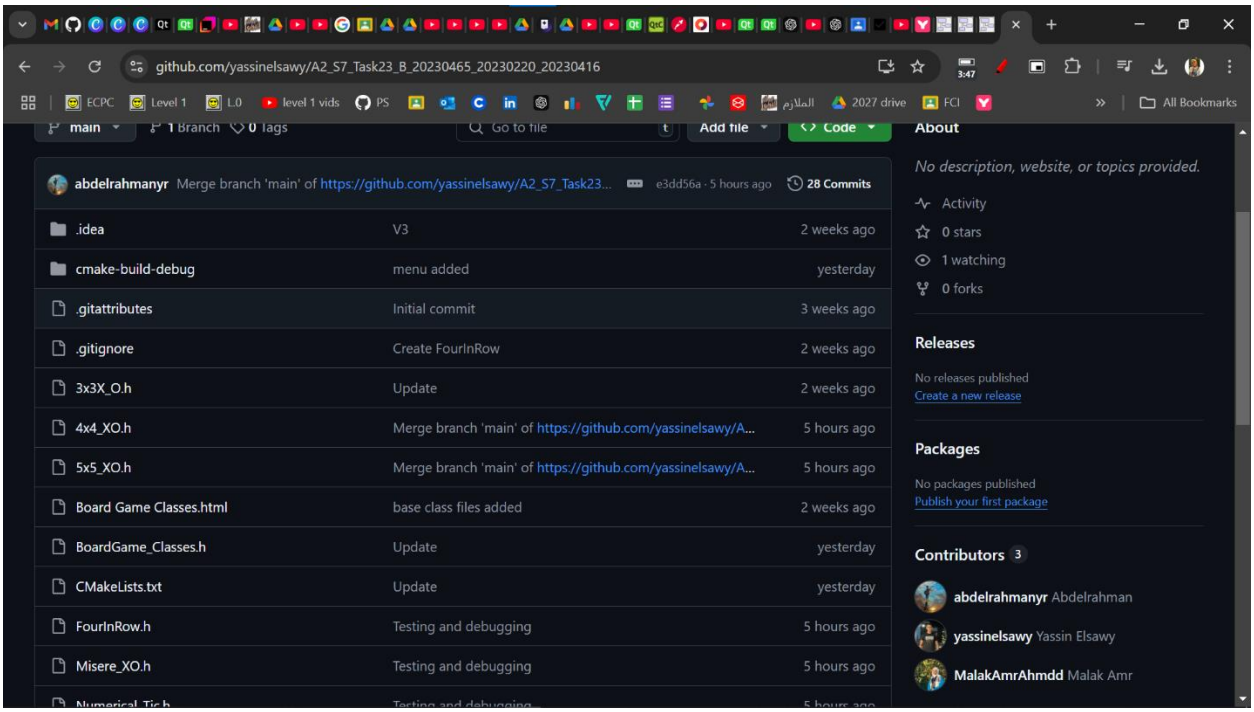
Malak

- In *Numerical Tic-Tac-Toe* the game lets a player win even if he only got two numbers with sum of 15 instead of 3, so I added a condition to check if an empty cell is in the winning sequence in *is_win()* function.

Bonus Description

- We implemented a GUI application that shows the group games 4x4 Tic-Tac-Toe and Ultimate Tic-Tac-Toe
- Qt library was used for implementation
- Work breakdown:
    - Abdelrahman: Designed the UI
    - Yassin: Implemented the UI and enhanced its functionality
    - Malak: Implemented the logic of the games

GitHub Repo



GUI Video Link

https://youtu.be/MM9XJlXEb50