A computer network diagram with arrows pointing to the earth

Description automatically generated with medium confidenceA cartoon of a person sitting on a chair

Description automatically generated**Cairo University**

**Faculty of Computers and Artificial**

**Intelligence**

**Object-Oriented-Programming**

**Sent to: Dr. Mohamed El-Ramly**

**CS213**

* **Assignment:** Assignment 2.
* **Task:** Task 2,3,4,5.
* **Section:** S 23.
* **Project Name:** Report of our Game Center.
* **Name, IDs and E-Mails:**

|  |  |  |
| --- | --- | --- |
| Name | ID | E-Mail |
| Aly El- Deen Yasser Ali | 20231109 | ali.el.badry.747@gmail.com |
| Fatema El-Zhraa Ahmed Mohamed | 20230280 | fatmaelfeky922@gmail.com |
| Alaa Tarek Mohammed Salah El-Deen | 20230064 | alaatarek2090@gmail.com |

|  |  |
| --- | --- |
| Name | What does he/she do? |
| Aly El-Deen Yasser Ali | **Game 3, 6, 7 and UI** |
| Fatema El-Zhraa Ahmed Mohamed | **Game 2, 5, 8 and UI** |
| Alaa Tarek Mohammed Salah El-Deen | **Game 1, 4, Ui and Report** |

* **Description for classes created for games:**

 **MisereTicTacToc**  
A modified Tic-Tac-Toe game where players aim to avoid winning. It handles board updates, display, and checks for win, draw, or game-over conditions.

 **MisereTicTacTocPlayer**  
Represents a human player who provides move coordinates. It manages player input using their symbol and name.

 **MisereTicTacTocRandomPlayer**  
A player that makes random moves by selecting empty positions on the board, adding unpredictability to the game.

 **MisereTicTacTacAIPlayer**  
Implements an AI player that uses a recursive algorithm to find the best move, ensuring optimal play to avoid winning.

 **TicTacTocSize4Board**  
A 4x4 version of Tic-Tac-Toe. It manages board updates, display, and game outcomes like wins, draws, or game-over states.

 **TicTacTocSize4Player**  
Represents a human player for the 4x4 game, handling input and validating moves during gameplay.

 **TicTacTocSize4RandomPlayer**  
A random player for the 4x4 board, automatically choosing moves to provide simple, unpredictable play.

 **TicTacToc\_Size5**  
Implements a 5x5 Tic-Tac-Toe game with additional logic for tracking and checking winning patterns using a map structure.

 **TicTacTocPlayer**  
Represents a human player for the 5x5 game, capturing and validating player moves.

 **TicTacTocRandomPlayer**  
A random player for the 5x5 game that automatically selects moves, adding unpredictability to gameplay.

* **Connect\_Board**

It is the main board of the game which holds the data of the board and its methods which relate to the game as display, update, is\_win, is\_draw and game\_is\_over . the Connect\_Board class inherits from the Board class.Also that class contains constructor and destructor

* **Connect\_Player**

It represents the human player (user) which makes the moves dependent on the input of the user and it inherits from the Player class. It contains a constructor and method called getmove which makes the user to play.

* **Connect\_Random\_Player**

It represents the randomized computer all its moves depends on random moves and it inherits from the RandomPlayer class . it contains constructor and getmove function also as that makes the randomized moves.

* **Math\_Tic\_Tac\_Toe\_board**

It represents the board of the game . the class inherits from the Board class .

The class contains methods that runs the game according to the rules of the game .For example , display\_board, update\_board ,is\_draw, game\_is\_over and is\_win . there is a destructor is added . there is also a vector of boolean in the private section to see the valid numbers which is used only once in the board.

* **Math\_Tic\_Tac\_Teo\_player**

It represents the human player (user) . it contains a constructor and function called getmove which takes the moves from the user . Also , it contains a vector of numbers which represents the valid range of numbers to play with weather even or odd .That class inherits from the Player class.

* **Math\_Tic\_Tac\_teo\_random\_player**

It represents the random computer player which plays with random moves .

It contains same attributes and same methods of the Math\_Tic\_Teo\_player, but the implementation differs . it inherits from the RandomPlayer class.

* **X\_O\_Board**

It represents the board of the game and that class is made to be used in the Tic\_Tac\_Teo\_9x9 \_board as association relation. The class contains constructor ,destructor ,update\_board ,display\_board,..etc. similar to most of the boards of the other games but there are three methods are added which are setter,increment and display row.

Setter helps in initialization as ensuring . increment helps in the logic of the n\_moves of the Tic\_Tac\_Teo\_9x9 game. Display\_row helps in the display of the Tic\_Tac\_Toe\_9x9\_board.

* **Tic\_Tac\_Teo\_9x9\_board**

It represents the board of the 9\*9 game actually two boards : one is small to represent each board takes which character and that board is 3\* 3 . the other one is 9\*9 board which already the player plays on . the small one to represent only how is going to win while the other board represents the game to be played by the user. It contains the similar methods of any game. This class contains pointer of pointers of X\_O\_board to represent the 9\*9 . As the main controller here for win , draw or updates of the game is the 3\*3 board which determines the state of the game but the other one for the dynamic moves only and helps the 3\*3 board in judging of the state of the board. The class inherits from the Board class.

* **Tic\_Tac\_Teo\_9\*9\_player**

It represents the human player (user). It inherits from the Player class. It contains constructor and getmove function which takes the location of the cell to play in the game.

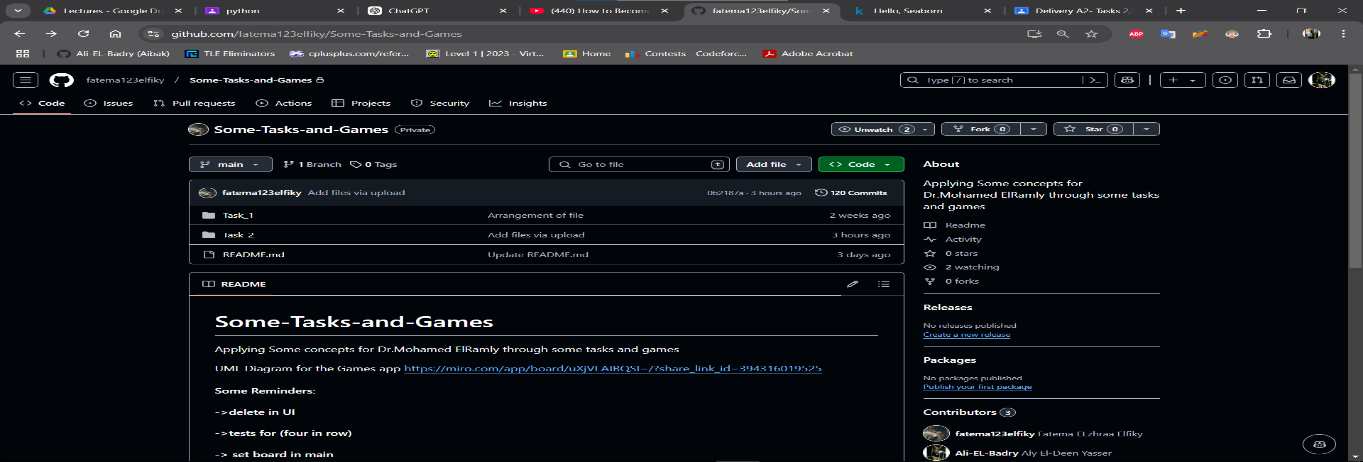
* **Tic\_Tac\_Teo\_9\*9\_random\_player**

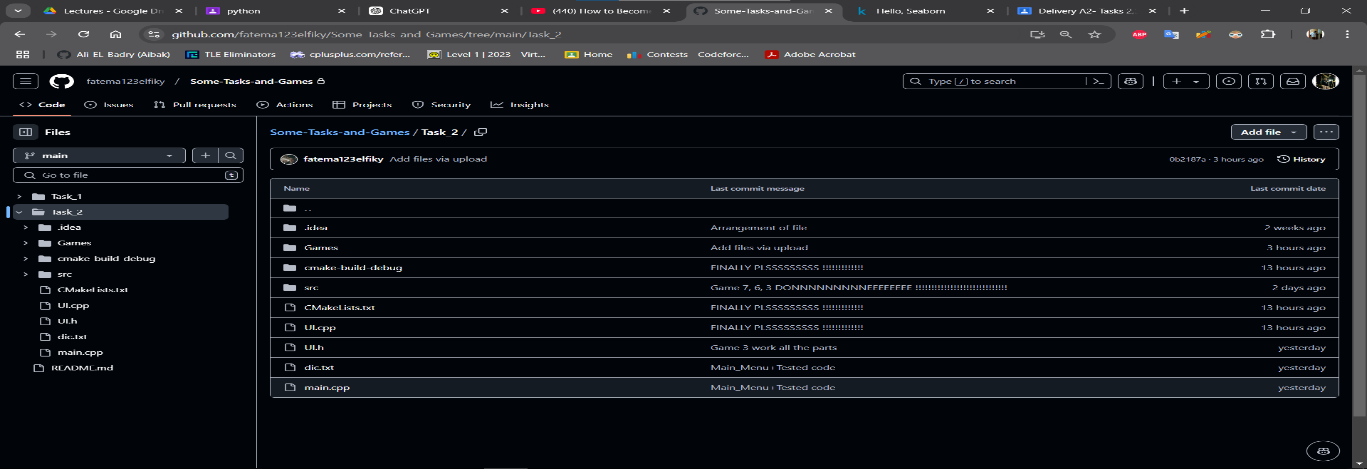
It represents the random computer player . It inherits from the RandomPlayer class. It contains also constructor and getmove function which makes a random move to play.

* **UML Design Digram :**

**Here is the link of the UML Design Diagram :** [**https://miro.com/app/board/uXjVLAIBQSI=/?share\_link\_id=394316019525**](https://miro.com/app/board/uXjVLAIBQSI=/?share_link_id=394316019525)

* **Github Repository:**

****

****

* **AI Bounce Explaination:**

**In Game 6 : The AI player uses the Minimax algorithm to determine the best move in a Misère Tic-Tac-Toe game. The code comprises two main functions:**

1. **calculateMinMax  
   This recursive function evaluates the board state to predict possible outcomes:**
   * **If the game is won (is\_win()), the function returns 1 or -1 based on whether the current player (Maximizing) wins or loses.**
   * **If the game is a draw (is\_draw()), it returns 0.**
   * **For each empty position, the function simulates a move, switches turns to the opponent, and recursively evaluates the next board states.  
     It minimizes the opponent's score while maximizing the AI's score to ensure optimal play.**
2. **getBestMove  
   This function iterates through all available moves to find the optimal one:**
   * **For each empty position, the function simulates a move and uses calculateMinMax to evaluate its outcome.**
   * **The move that yields the highest value (best outcome) is selected.**

**By analyzing all possible future moves and counter-moves, the AI ensures it always plays optimally, making it challenging for an opponent to win.**

* **Aly El-Deen Yasser Ali’s Report:**

My Review of Fatema’s Code:

It is a perfect code that handles many test cases, it is also good as a user interface, but the code has some issues as some of the variable names are inconspicuous and need to be clearer as it is hard to know what is the target from that variable, also, many comments consist of code parts that is commented and not used again.

My Review of Alaa’s Code:

It is a good code that is easy to understand and easy to discover any part of it but there are some issues as it has the same problem as Fatema that many comments consist of code parts that are commented on and not used again, also in the part that takes the name of the user if you entered your name separated with a space it will cause problem and only one part of name will be taken to to the name of player not all the parts.

* **Fatema Elzhraa Ahmed’s Report:**

My review of Aly’s Code :

The code is clean and has good readability. The code passed the tests successfully . The code uses string to represent the player symbols, but since only single characters like 'X' or 'O' are needed, using char would be more memory-efficient and sufficient for this context. string provides additional functionality that isn't needed here, which may add unnecessary overhead. In the Misere ,in the is\_win function, there are two separate loops for checking rows and columns. These can be combined into a single loop to reduce redundant computations and improve efficiency. This would make the code more streamlined and reduce overhead.

My review of Alaa’s Code :

The code passed the tests successfully. The code was readable and understandable. On the other hand , She has inherited from the framework but instead of determination of the template in the inheritance ,she inherited it as a template ,so it was better to determine it as the game goes as char only or numbers only and so on .The code miss the point of security of the data as in the word\_tic game has vector of strings in the public sector and it should be in the private sector as to be secured for the class only. Also the last point , the definition of a global array in the header to be used in a method of the class is not as good practice and also causes a lack of encapsulation . so it was better to make it inside the class and private attribute .