**Tic\_Tac\_Toe game in Go**

This is my **GO** version code for a player board game featuring a human player, **MG**, and a computer player, **AI**. The game uses a 5x5 grid, with each cell initially empty. Here is the design and the explanation of the code.

**Global Variables and Constants**

A screenshot of a computer screen

Description automatically generated

Picture 1: Cells

* **Cells:** a global array representing the 5\_by\_5 game board imported for input and output functions, generating random numbers and string manipulation, and seeding the random number generator.

A screenshot of a computer screen

Description automatically generated

Picture 2: players.

* **playerMG** and **playerAI:** constants representing the human player and the AI.

**Function**

A screenshot of a computer screen

Description automatically generated

Picture 3: initialise board.

* **InitialsBoard:** sets all cells on the board to “—“, indicating they are empty.

A screenshot of a computer screen

Description automatically generated

**Picture 4: display the board.**

* **displayBoard:** prints the current state of the board to the console, displaying the grid in a 5x5 format.

A screenshot of a computer screen

Description automatically generated

Picture 5: checking winner.

* **checkWinner:** determines if the specified player has won. It checks predefined winning combinations in rows, columns, and diagonals. To see if they are filled with the player's markers.

A screenshot of a computer screen

Description automatically generated

Picture 6: add move.

* **AddMove:** places the player’s marker in the specified cell if it's empty. Returns true if successful, false otherwise.

A screenshot of a computer screen

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

Picture 7: two pictures in one: play game.

* **PlayGame:** manages the game loop. Players alternate turns, with the AI choosing a random cell and the human player inputting their choice. The function handles move validation and run management, displays the board, and checks for a winner after each move.

A screenshot of a computer

Description automatically generated

Picture 8: main game.

* **Main:** the entry point of the program. It repeatedly calls **playgame()** to start new games until the player chooses not to continue playing.

**Game Mechanics**

* **Initialisation**: the board is initiated, and the game loop begins.
* **Turns:** the game alternates between humans and AI players. The turns array defines the sequence of turns.
* **Player moves:** The player moves for AI randomly selects a cell, and human by inputs a move through the console.
* **Move validation:** check if the chosen cell is empty before placing the marker.
* **Winning condition:** after each move, the game checks if the current player has achieved a winning combination.
* **End of Game:** the game ends if there’s a winner or if all turns are used. That’s a tie. The replay option asks the player if they want to play again. The game continues until the player types no or press enter.

**Go UML Diagram**

A screenshot of a computer

Description automatically generated

**Picture 9: the drawing of the class diagram.**