ET721 – Software development practicum

**Unit test project: Drop chip game**

*Part 1: Analyzing the Connect4 Class in main.py*

Student’s name: Aseel Rahman

Date: OCT 24, 2025

**Learning outcome:** The objective of this project is to thoroughly analyze the Connect4 class defined in the main.py file. This process will involve examining the functionality of each method within the class, running them independently, and understanding how each behaves in isolation. The analysis will include modifying class values, observing the impact of these changes, and documenting findings.

**Steps to Complete the Analysis:**

1. **Access the main.py file**: Open and review the contents of the file to familiarize yourself with the class structure.
2. **Download the file unit\_test\_project\_part1.docx**: This document will be used to record your observations, changes made to the class, and results obtained from running the class.
3. **Analyze each class**:
   * Run the class independently.
   * For each method, understand how it works by testing its functionality.
   * Make modifications to some values in the class and observe how these changes affect the behavior of the class and the game logic.
4. **Document the analysis**: In the Word document (unit\_test\_project\_part1.docx):
   * Record how each method behaves when run.
   * Note the modifications made (e.g., changing player symbols, adjusting the board size, etc.).
   * Describe the outcome of these changes and any other observations regarding the class’s behavior.
5. **Conclude the analysis**: Summarize the key takeaways from the modifications and their effects on the game’s mechanics.

**Python class analysis:**

1. **\_\_init\_\_(self)**

Description: This method is the constructor of the Connect4 class. It runs initially when a new game object is created. It sets up the board as ROWS and COLS. Also it sets ‘X’ as the first player.

Changes and observations: When I changed the self.current\_player = 'Y' to self.current\_player = 'x'. The game starts with player Y’s turn.

1. **switch\_player(self)**

Description: This method switches the current player after each turn. If the current player is ‘X’ then it changes to ‘O’, and if it’s ‘O’ then it switches it back to ‘X’.

Changes and observations: I modified the method to switch between ‘Y’ and ‘N’ instead of ‘X’ and ‘O’. When I run the program, it just shows Player ‘Y’’s turn and don’t get switched.

1. **print\_board(self)**

Description: It displays the game board with columns divided by ‘|’ and the method shows ‘X’ as red and ‘O’ as blue in the board.

Changes and observations: I replaced ‘|’ with ‘\*’ from everywhere in the method and now the game board is made with ‘\*’.

1. **drop\_chip(self, column)**

Description: This method adds the current player’s chip to the selected column. It fills from the bottom up and return true if successful or false if the column is full or invalid.

Changes and observations: I changed the starting row to drop chips from the top instead of the bottom. Now the chips stacked upward instead of downward.

1. **check\_win(self, player)**

Description: This method checks if a player has four chips in a row horizontally and vertically or diagonally. It returns true if the player wins.

Changes and observations: I changed the counts from 4 to 3 to win. Now if a player stacks 3 chips in a row they wins.

1. **play\_game(self)**

Description: this is the main method and game loop. It prints the board, asks for a column number, drops the chip, checks for a win or tie, and switches players each turn.

Changes and observations: I added red color code to print the text ‘Welcome to connect 3!’ in red. Now the message prints in red in terminal.

**Conclude the analysis: After testing and changing different parts of the Connect4 class, I got better understanding of how each method works and connects to the game. By modifying things like the board size, player symbols, and colors, I could see how small changes affect the overall gameplay.**