Jason Randazza

Tic-Tac-Toe Project

CSC229

This project implements a two-player Tic-Tact-Toe game using the Java programming language. A singly linked list was used as the data structure. The board is a 3x3 grid, but instead of using a 2D array, each cell in the grid is a node inside of the singly linked list. This program allows the two players to alternately place their marking symbol (‘X’ or ’O’) on the board by choosing a position from 1 to 9 on the grid. The game checks for a player winning or if there is a draw after each move. The players can choose to play again after the game ends.

The singly linked list has been implemented with the attributes: position, symbol, and a next node reference. When a player makes a move, the list is traversed to find the correct node based on the position number that was entered. The symbol is then stored in that position. The display function maps the nodes back into the 3x3 grid format for printing to the console and easy readability.

Some key functions that have been included are Print Board which displays the nodes in the 3x3 grid format after each player move. Update Board which adds the player’s move to the corresponding node and ensures that the move is valid. Win Condition Check which happens after every move, the game checks if the values match sets of three nodes representing columns, rows, and diagonals. A Draw Condition which checks if all the nodes are filled, and no player has won the game will end in a draw. And Reset Board which resets all the values in the nodes back to ‘ ‘ so the game can be replayed without having to restart the program.

Some challenges that were encountered included traversing the linked list. Since the singly linked list does not allow for direct access to elements by index, each traversal required moving from the head node to the node in the requested position. The helper function getNode() was implemented to retrieve the node at the requested position by iterating through the list. Another challenge was the win condition logic. Because the list is linear mapping of positions 1 to 9 being mapped to the grid needed to be ensured by use of the helper function, checkCells(), which compares the values of three positions on the board to see if they map the players symbol placements. The final challenge faced was handling input validation. In order to keep the game flowing properly a try-catch block was added around the player input to catch invalid entries which prevents the program from crashing. Validation logic was also implemented to ensure that players can only select valid or and unoccupied positions on the game board.

Optional enhancements that were included are a game replay option. After the game ends in a win or draw the players are asked if they want to play again. If they choose yes then the list is reset to all unoccupied spaces so the game can start over without the need to recompile the program. Another enhancement is the error input handling. This ensures that non-integer inputs do not crash the program. If an invalid integer is input the player will be prompted to enter another valid value.

This project provided valuable insights into working with dynamic data structures in a practical application. Using a singly linked list for Tic-Tac-Toe might not be as efficient as a 2D array, but it did force me to think more creatively about how to manage the game’s state and operations in a linear structure. This project also helped me to improve my skills and gain further understanding in handling user inputs, checking win conditions, and maintaining a clear and modular design.