NOTES:
Include all in one document or in separate documents (and be clear where and what names)
Use Case Model
Use Case Narratives

At least a UML class diagram with all details. Either include here, link to any kind of pdf or draw.io or github or ...

If you did other design items, documents, sequence diagrams, etc, also give the info on how to find them here.

Objective:
For this assignment we were tasked with creating an updated version of our Tic-Tac-Toe games from the first homework assignment. Collaborating with our new group members and compiling all of our original code to create a new and improved version of our own games.

Project Approach:

- Program will be written in Java 9.0.4
- Review of eachothers code from the previous assignment so we can figure out the direction we would like to take as a group
- Group members will submit their changes to our github repository with proper documentation to go along with each commit.
  - After each submission we will report to our basecamp discussion on the changes made so everyone is on the same page going forward.
- UML Diagram will be made to outline our class definition before we begin to write our code.
  - This will be made on Draw.io and explicates how a user will move through our system
- Class and Function diagram will be made to show the functions we will have for our respective classes.
- Function definitions will be given to explain their own unique purpose within the program

<u>Program Requirements:</u>

- Initial class definition
- Create a ***game board*** for our tic-tac-toe game
    - The board must be updated after each player's respective moves
- System must be able to distinguish between ***player X*** and ***player O*** (two person game)
    - Create an option for players to enter their own names
- Report to the players whose turn it is after a move is made
- Game board should automatically update after a ***legal move*** is made. This will be achieved by efficient data structures that utilize a minimum amount of memory to ensure that the program runs smoothly and efficiently
- System must prohibit players from making an ***illegal move*** in a location that already has a space
    - Report a message to the player that this is an invalid move, and they must try again
    - Gives the players a maximum of 3 retries before skipping the players turn
- System must ***recognize a win*** as three of their moves matching vertical, horizontally, or diagonally
- System must ***recognize if a draw*** is found; meaning that neither play can win.
- After a "win" has been found report to the players if a player has won the game
- Game counts a tally for winners when chosen to replay the game
- When the players are finished playing, prompt them to either ***replay the game*** or ***exit*** the program
    - Should the players choose to continue playing ***keep a tally*** of how many times they have each won
- Create a computer vs. player option
    - Functions the same way as the pvp mode, except the computer places its symbol of X or O by itself
    - Retry and Tally modes are available in the PvC mode

<u>Function Definitions:</u>

- Player Functions
    - nameSet() - Opens a scanner for the users to input their name so it's a personalized game instead of having player X and player O
    - playerSet() - Has one player select their symbol while the other player defaults to the remaining symbol; then reports to the players what symbols they'll be using

    - playerOrder() - prompts the players to decide what order they'd like to play in
- Game Functions
    - play() - runs the game in an infinite loop until it is told to stop
    - clearBoard() - resets the game board
    - clear() - clears the linked lists so it can be empty for a new game
    - draw() - draws the game board
    - player1() - allows the 1st player to make their moves and tells them if their moves are invalid, otherwise it will accept their move
    - player2() - allows the 2nd player to make their moves and tells them if their moves are invalid, otherwise it will accept their move

- PvC Functions
    - playComp() - runs the game in a specific order according to CvP scenarios, different from the normal PvP play method
    - Comp() - allows the computer to make its moves based on the available locations left on the gameboard
- Game Check Functions
    - checkWin() - checks the 9 possibilities of wins for a player; used in their move selections (player1() and player2())
        - Records the tallies for players wins
        - Lets them know if a player won
        - Prompts them to play again or exit