Practical project

D04 Java I
Tic Tac Toe game

Evaluation

The project is worth 20% of your final grade.

Submission

The project is due on the last day of class. It should be submitted on Dropbox.

Your practical project should include:

- Your Eclipse project.
- A readme file. This should describe the point of entry, the optional functionalities implemented and any other improvements made to the game, plus any other relevant information concerning the game.
  If any parts of the game are incomplete or do not function correctly when you submit, then you must also describe these issues in the readme file.

Description

Design and develop a Tic Tac Toe (Noughts and Crosses) game using a Java Swing graphical user interface.

Rules

The board is a 3 × 3 grid.

The X player plays first.

The X player and the O player take turns playing.

A player's turn consists of playing on an unmarked square on the grid, which marks it as theirs.

Win condition: A player wins if they acquire three consecutive squares in a row, column, or diagonal line.

Lose condition: A player loses if the other player wins.

Draw condition: Both players draw if all squares have been played with no winner.

General evaluation [2 points]

- [2] Design and code quality
  The application is structured well, and the code is efficient and clear.

Required functionalities [10 points]

- [1] Choose mode
  When starting the game application, the user must be able to choose the mode:

  a) play a local game against a friend
  b) play a local game against an AI player [if applicable]
  c) launch a network game [if applicable]
  d) join a network game [if applicable]

- [2] Local game against friend
  The game runs entirely on one computer. There is one user interface window, and both players play on it, passing the mouse back and forth.

- [2] Game logic/flow
  The game logic and flow must be implemented correctly. For instance: only one player can play at a time, the turn sequence is correct, players cannot play on already marked squares, etc.

- [2] Win/draw algorithm
  Whenever the game could possibly end, the game must check for a win/draw.
  This algorithm should be implemented correctly and efficiently, and should use any available information to narrow down the search as much as possible.

- [1] Play again
  When a round has ended, the players should be able to play another round, continually.

- [1] User interface: Display information
  The game must accurately display the game state, and display any important messages.

- [1] User interface: Controls
  The game must offer game controls that behave correctly and are clear and easy to use.

Optional functionalities [8 points, 18 max]

- [2] Local game against AI player
  The game runs entirely on one computer. There is one user interface window, and both the human player and AI player play on it, taking turns.

- [6] Network game against friend
  Player A launches the network game on computer A. This creates the single central game model and server, plus a client window for player A, all running on computer A. The game waits for player B to connect.
  Player B then joins the game on computer B. This connects with the server on computer A, creates a client window for player B on computer B, and allows the game to begin.

- [1] Random first player
  Each new round, the first player is chosen randomly.

- [1] Player scores
  The game displays each player's score, representing how many rounds they have won.

- [2] Game logs
  The game saves players' scores / high score to a log file after each round, or saves players' moves and wins/draws to a log file after each move.

- [2] Variable board size
  The game allows the user to choose the size of a board: 3 × 3 grid, 4 × 4 grid, or 5 × 5 grid.

- [4] Other features / improvements
  Any other features or improvements you might want to add. For example:

  - Implement advanced AI player algorithm, so that it plays intelligently
  - Add images, add sounds, or otherwise refine the user interface appearance
  - Etc.