NISANTASI UNIVERSITY

DEPARTMENT OF SOFTWARE ENGINEERING

Object Oriented Programming

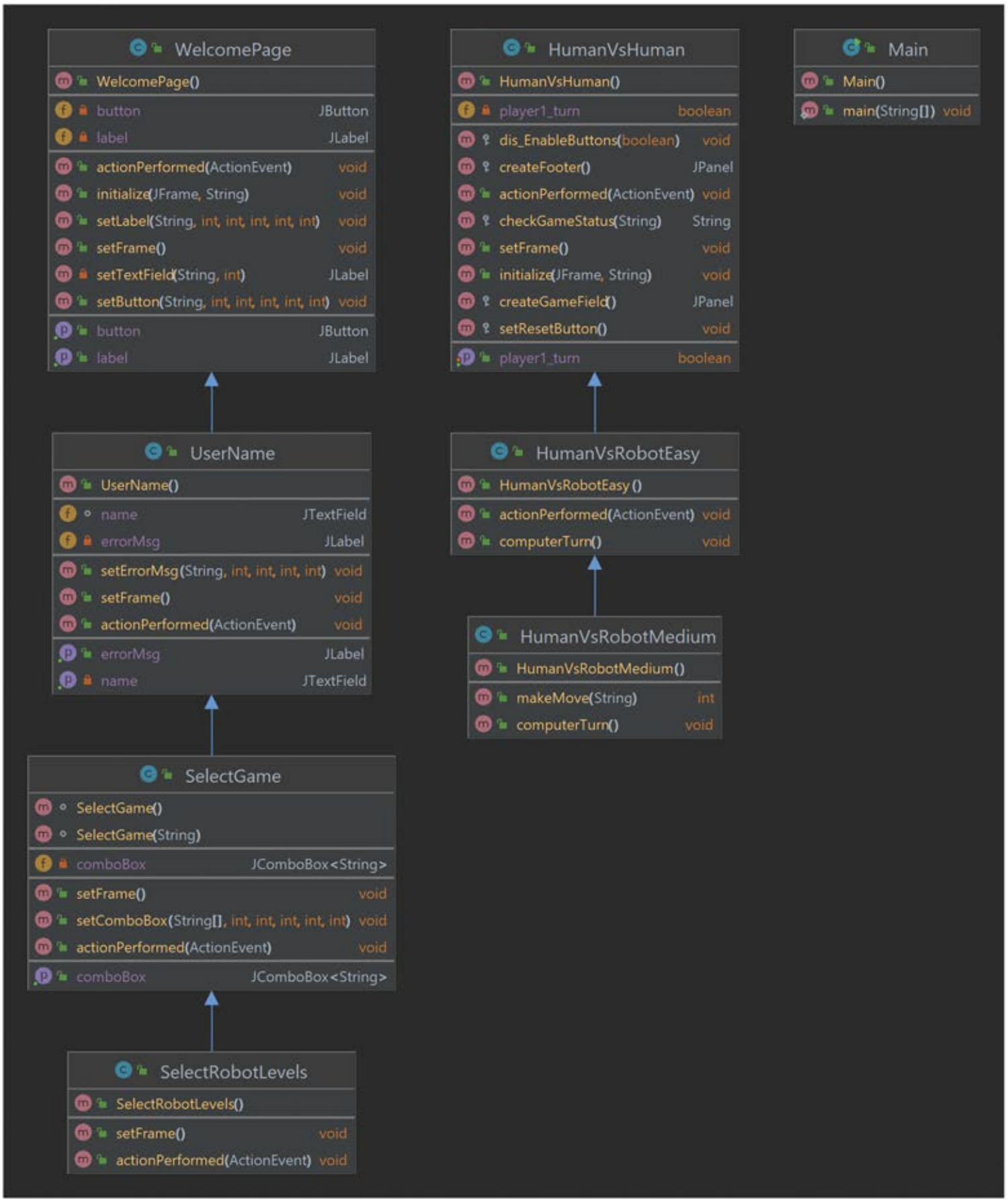
Instructor: Dr. Jawad Rasheed

PROJECT NO: 1
PROJECT NAME: Desktop Tic-Tac-Toe

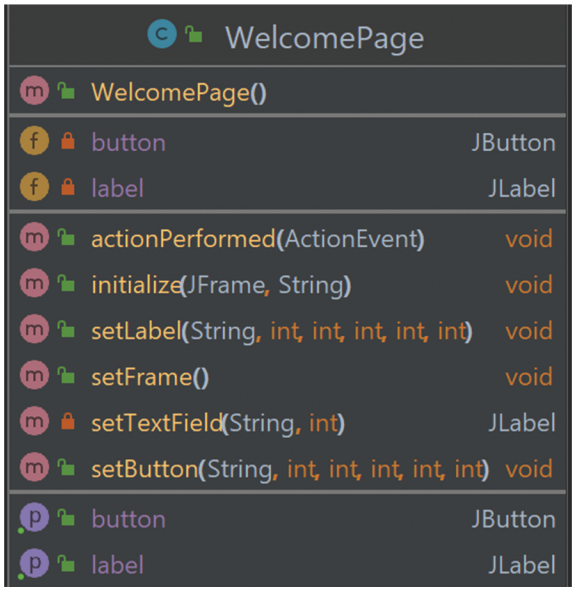
DELIVERY DATE: 08/01/2022

Submitted by:
Name: Abdihakim Ismail Mohamed
Student NO: 20212022160

CLASS DIAGRAM



WELCOMEPAGE CLASS



This Java class represents a welcome page for a Tic-Tac-Toe game. The class contains a main frame and two subcomponents: a JLabel for displaying text, and a JButton for allowing the user to start the game.

The setFrame method is responsible for setting up and displaying the welcome page. It initializes the main frame using the initialize method and adds three text labels to it using the setTextField method. It then adds a button to the frame using the setButton method and sets the frame to be visible.

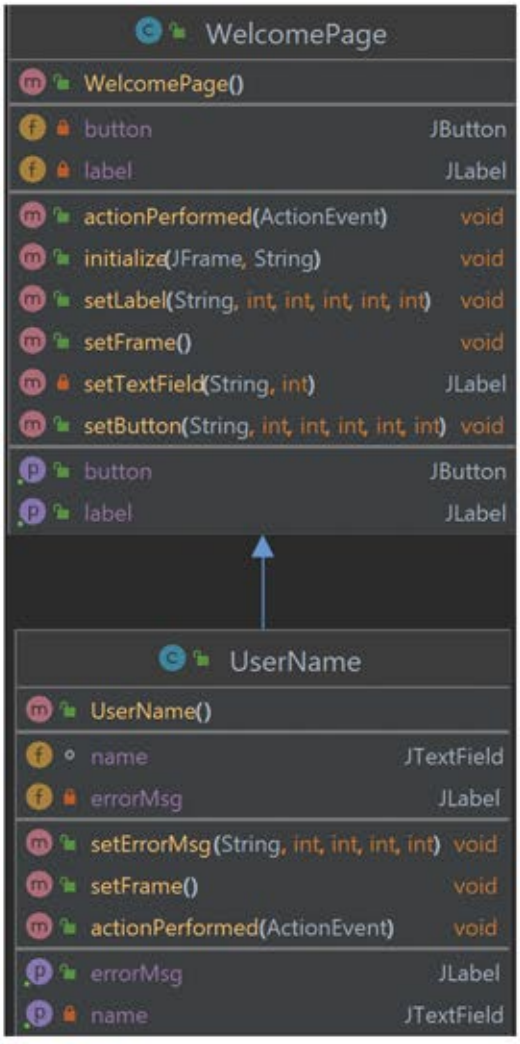
The initialize method sets the default close operation and title of the frame, sets the frame to be non-resizable, sets the size and location of the frame, sets the background color of the frame, and sets the layout of the frame to be null.

The setTextField method creates and returns a JLabel object with the specified text and font.

The setLabel and setButton methods set the text, font, and size of the label and button objects, respectively.

The actionPerformed method is called when an action is performed on the button object (e.g. when the button is clicked). If the source of the event is the button object, the current frame is disposed of and a new UserName object is created and its user interface is displayed.

USERNAME CLASS



The userName class represents a user interface for entering a username in a Tic-Tac-Toe game. The class extends the WelcomePage class and contains a main frame, a JLabel for displaying error messages, a JTextField for allowing the user to enter their username, and a JButton for allowing the user to proceed to the next step.

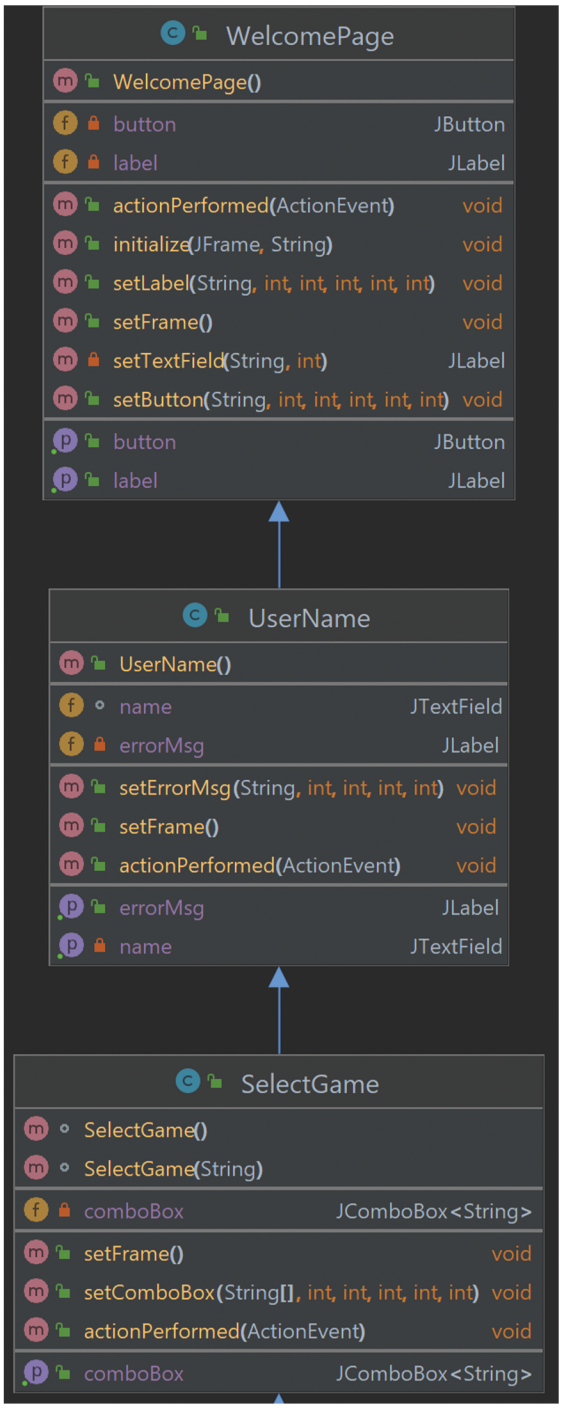
The setFrame method is responsible for setting up and displaying the user interface. It initializes the main frame using the initialize method and adds a label, text field, and button to it using the setLabel, getName, and setButton methods, respectively. It also adds a label for displaying error messages using the setErrorMsg method. The frame is then set to be visible.

The setErrorMsg method sets the text, color, and size of the error message label.

The getName method creates and returns a JTextField object for the user to enter their username.

The actionPerformed method is called when an action is performed on the button object (e.g. when the button is clicked). If the source of the event is the button object, the program checks if the username text field is blank or has less than four characters. If either condition is true, an error message is displayed. If the username is valid, the current

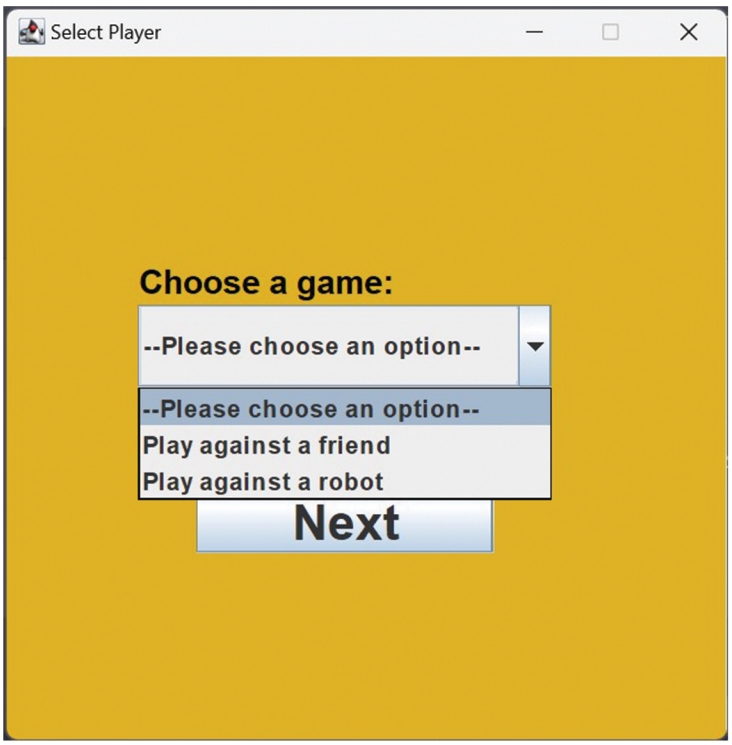
SELECTGAME CLASS



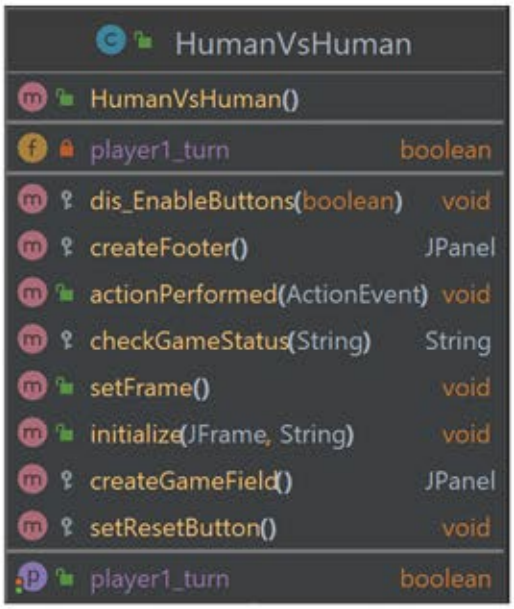
This is a Java class that represents a user interface for selecting the type of game in a Tic-Tac-Toe game. The class extends the UserName class and contains a main frame and two subcomponents: a JComboBox for allowing the user to select the game type, and a JButton for allowing the user to proceed to the next step.

The setComboBox method creates and returns a JComboBox object with the specified list of options and font.

The actionPerformed method is called when an action is performed on the button object (e.g. when the button is clicked). If the source of the event is the button object, the program checks if the combo box is set to the default option. If this is the case, an error message is displayed. If the user has selected either "Play against a friend" or "Play against a robot", the current frame is disposed of and a new HumanVsHuman or SelectRobotLevels object is created and its user interface is displayed. The game type and the current time are also written to the "file.txt" file.



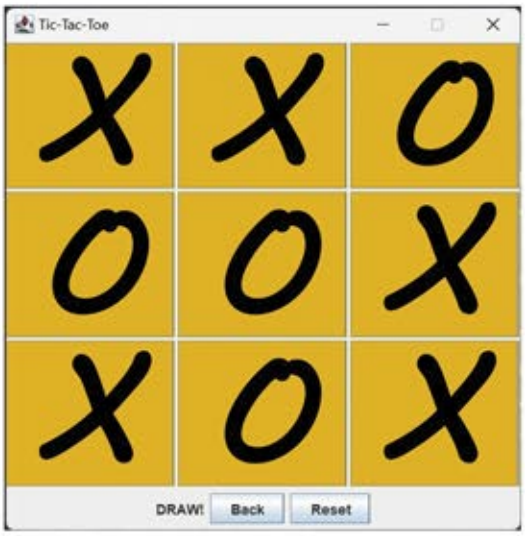
HUMAN VS HUMAN CLASS

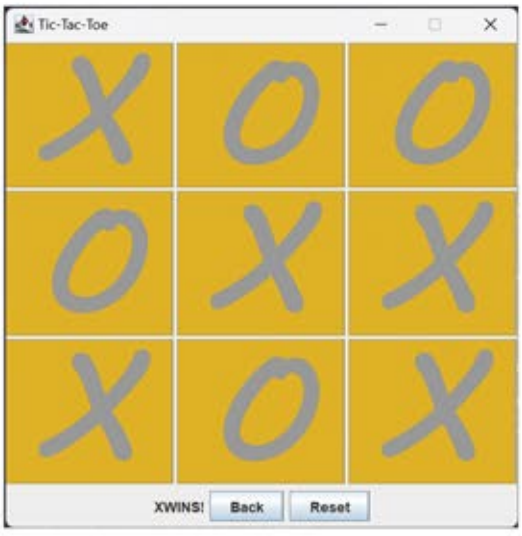


The HumanVsHuman class represents a Tic-Tac-Toe game where two human players take turns to place their symbols (either 'X' or 'O') on a 3x3 grid. The game can be started by clicking on one of the buttons on the grid, and the player whose turn it is will place their symbol on that button. The game ends when one of the players gets three of their symbols in a row, either horizontally, vertically, or diagonally, or when all the buttons on the grid are filled and no player has won. In that case, the game is a draw.

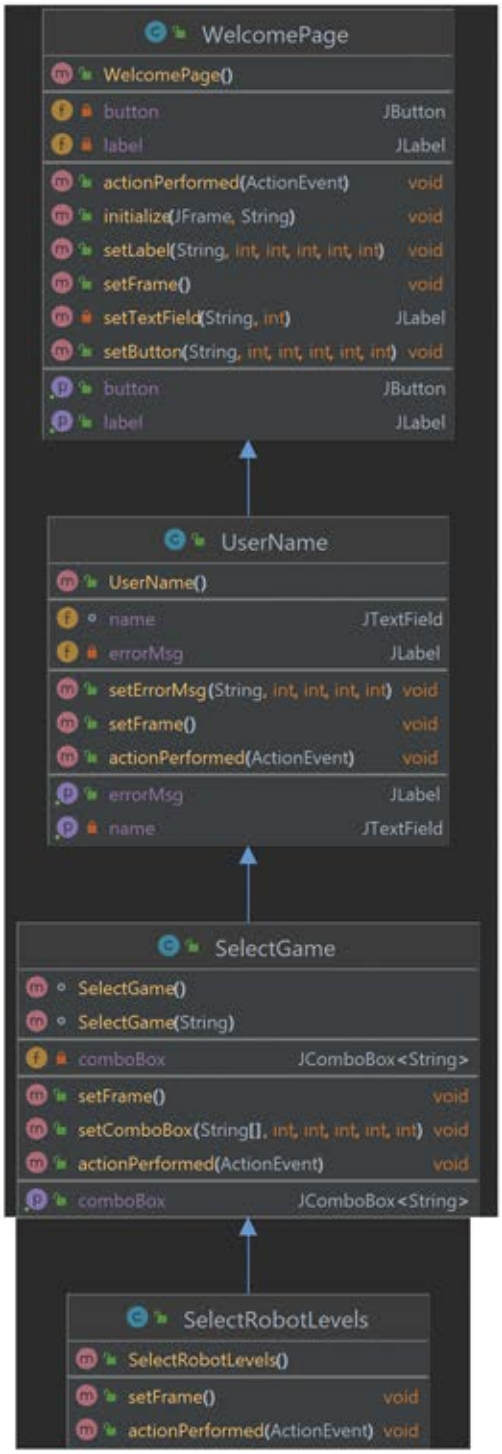
The class has a number of methods that are used to set up and display the game interface, such as setFrame, initialize, createGameField, and createFooter. It also has methods that handle the game logic, such as setResetButton, which resets the game and enables all the buttons on the grid, and checkGameStatus, which checks if any of the players has won or if the game is a draw.

The class implements the ActionListener interface and overrides the actionPerformed method to handle events triggered by the buttons on the grid and the reset and back buttons. When a button on the grid is clicked, the actionPerformed method is called and the symbol of the current player is placed on that button. The checkGameStatus method is then called to determine if the game has ended, and if so, the dis_EnableButtons method is called to disable all the buttons on the grid. If the game is not over, the player1_turn variable is toggled to switch to the other player's turn. If the reset button is clicked, the setResetButton method is called to reset the game, and if the back button is clicked, the current frame is disposed and





SELECTROBOTLEVELS CLASS



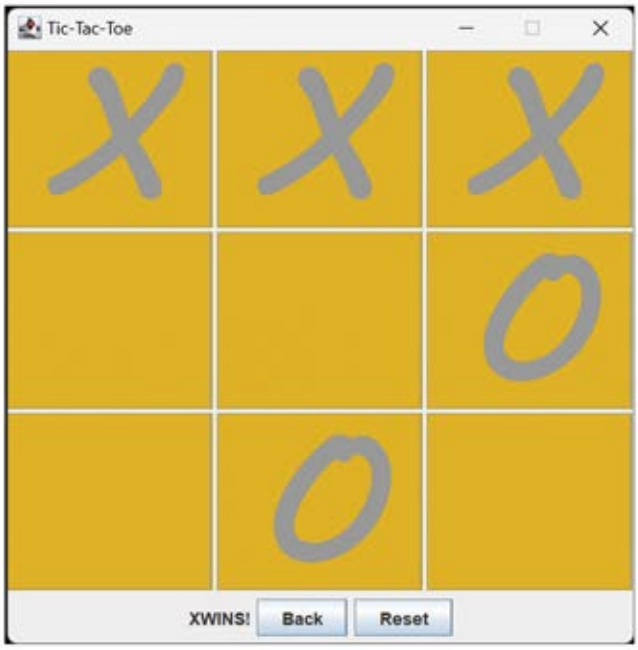
This is a class for selecting the difficulty level of a Tic-Tac-Toe game where one player is a human and the other is a computer. The class extends the SelectGame class, which provides a frame with a label, a drop-down menu, and a button for selecting a game level.

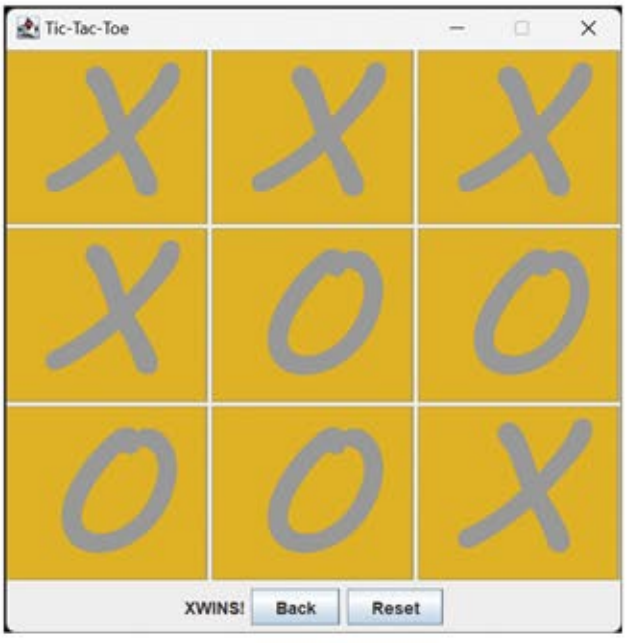
The setFrame method sets up the game selection frame, including the label and drop-down menu for selecting the difficulty level, and the button for starting the game. The actionPerformed method is called when an action is performed on the frame, such as clicking the button or selecting a level from the drop-down menu. If the action is a click on the button, the selected difficulty level is checked. If the level is not "--Please choose an level--", a new game frame is opened based on the selected level. If the level is "Easy", a new HumanVsRobotEasy game is opened. If the level is "Medium", a new HumanVsRobotMedium game is opened.

HUMANVSROBOTEASY CLASS

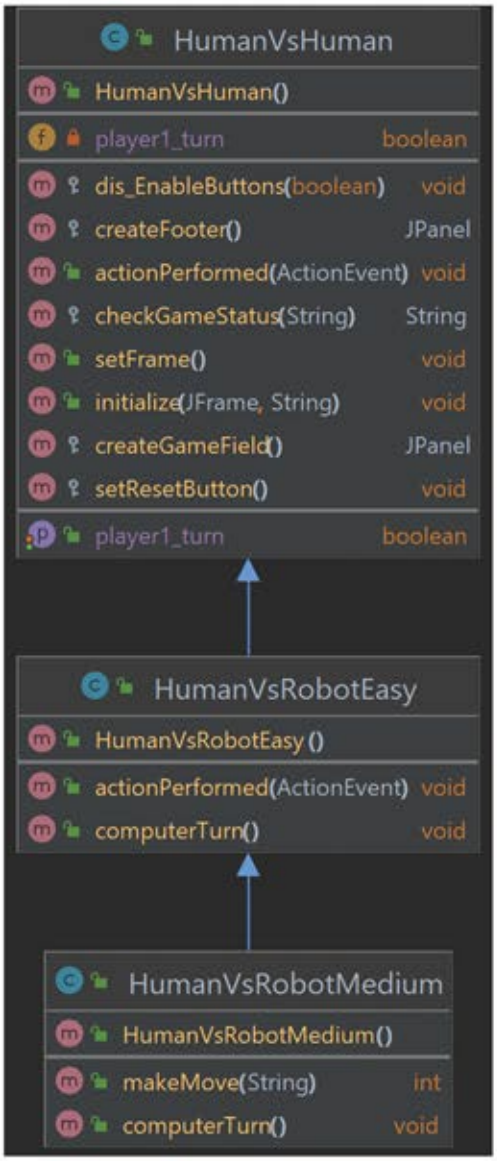
This is a class for a game of Tic-Tac-Toe where one player is a human and the other is a computer. The computer makes a move by selecting a random empty cell on the game board and placing an "O" in it. The human player makes a move by clicking on an empty cell and placing an "X" in it.

The actionPerformed method is called when an action is performed on the game, such as clicking a button or resetting the game. If the action is a click on the reset button, the setResetButton method is called to reset the game. If the action is a click on the back button, the current game frame is closed and a new game selection frame is opened. If the action is a click on one of the game cells, the cell is marked with an "X" if it's the human player's turn, and the checkGameStatus method is called to check if the game has been won or drawn. If the game is not finished, the computer's turn is triggered by calling the computerTurn method.





HUMANVSROBOTMEDIUM CLASS



This is a class for a game of Tic-Tac-Toe where one player is a human and the other is a computer. The computer's difficulty is set to medium, which means it will try to win or block the human player's win if it can, before falling back to making a random move.

The computerTurn method is called when it is the computer's turn to make a move. The computer first checks if it can win with one further move by calling the makeMove method with "O" as the parameter. If it can't win, it checks if it can block the human player's win by calling the makeMove method with "X" as the parameter. If it can't do either, it falls back to making a random move by calling the makeRandomMove method inherited from the HumanVsRobotEasy class.

The makeMove method checks if the computer has two in a row, a column, or a diagonal and can win with one further move. It does this by checking each row, column, and diagonal of the game board for two consecutive cells containing the specified player's symbol and an empty cell. If it finds such a combination, it returns the index of the empty cell as the best move. If it doesn't find any, it returns -1.