1. Objectives:

Our project name is Tic-Tac-Toe game. This game is very popular and is fairly simple by itself. It is actually a two player game. In this game, there is a board with *n* x *n* squares. In our game, it is 3 x 3 squares.
The goal of Tic-Tac-Toe is to be one of the players to get three same symbols in a row - horizontally, vertically or diagonally - on a 3 x 3 grid.

2. Overview:

This game can be played in a 3x3 grid (shown in the figure 2.1) .The game can be played by two players. There are two options for players:

(a) Human (b) Computer

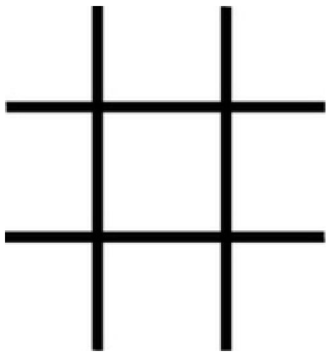


Figure: 2.1

2.1 Players:

For the option human, both the players are human and for the option computer, the first player is human and the second player is computer.

2.2 Theory of Game:

A player can choose between two symbols with his opponent, usual games use “X”and “O”. If first player choose “X” then the second player have to play with “O” and vice versa.

A player marks any of the 3x3 squares with his symbol (may be “X” or “O”) and his aim is to create a straight line horizontally or vertically or diagonally with two intensions:

a) Create a straight line before his opponent to win the game.

b) Restrict his opponent from creating a straight line first.

In case logically no one can create a straight line with his own symbol, the game results a tie.

Hence there are only three possible results – a player wins, his opponent (human or computer) wins or it’s a tie.

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

Figure: 2.2

If any player is able to draw three Xs or three Os in the following combinations then that player wins. The combinations are:

a) 1, 2, 3 b) 4, 5, 6

c) 7, 8, 9 d) 1, 4, 7

e) 2, 5, 8 f) 3, 6, 9

h) 1, 5, 9 i) 3, 5, 7

3. Core Logic - AI:

There are two core logics in this game – when both players are human, and when one is computer. Suppose the player use X and the computer use O . The logic used for the AI is as follows:

3.1 First move:

a) If the center is free, get the center. (Figure: 3.1)
b) Otherwise, get any of the corners. (Figure: 3.2)

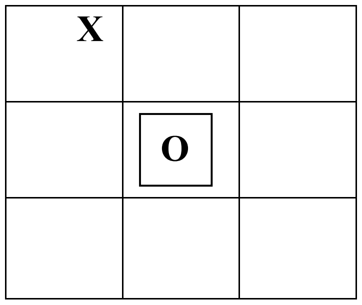


Figure: 3.1

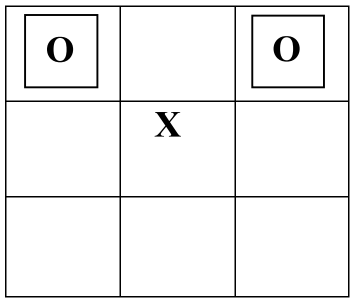


Figure: 3.2

3.2 Second move:

$\boxed{O}$ $\boxed{O}$

a) Block user from winning. (Figure: 3.3)
b) Option for winning by applying the following logic:
If the center is occupied by user, get any of the corners. (Figure: 3.4)

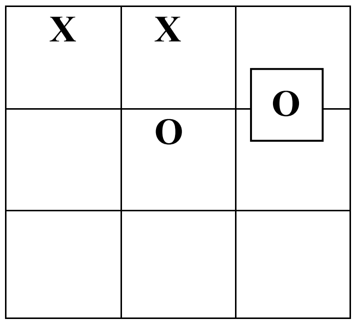


Figure: 3.3

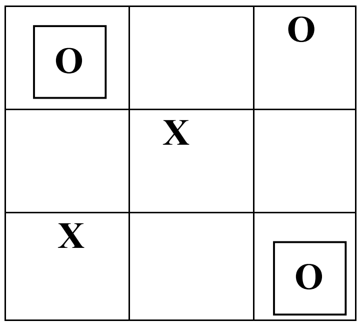


Figure: 3.4

Otherwise, the following cases happen:

Case 1:

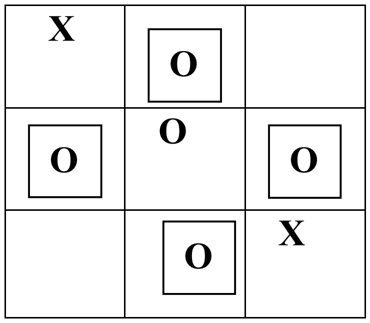


Figure: 3.5

If any situation arises like the figure 3.5 then the computer sets its symbol any one of the position among 2, 4, 6 and 8.

Case 2:

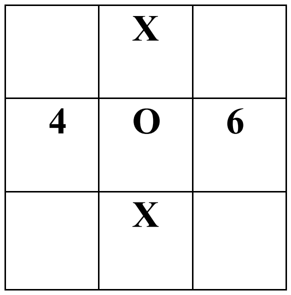


Figure: 3.6

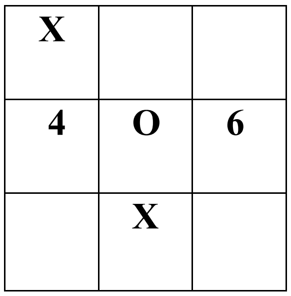


Figure: 3.7

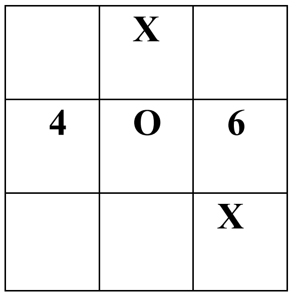


Figure: 3.8

If any situation arises like the figure 3.6 or figure 3.7 or figure 3.8 then the computer sets its symbol at any position among 4 and 6.

Case 3:

	2	
	O	X
X	8	

Figure: 3.9

	2	
X	O	X
	8	

Figure: 3.10

	2	
X	O	
	8	X

Figure: 3.11

If any situation arises like the figure 3.9 or figure 3.10 or figure 3.11 then the computer sets its symbol at any position among 2 and 8.

Case 4:

1	X	3
	O	X
7		9

Figure: 3.12

1	X	3
X	O	
7		9

Figure: 3.13

1		3
X	O	
7	X	9

Figure: 3.14

1		3
	O	X
7	X	9

Figure: 3.15

If any situation arises like the figure 3.12 or figure 3.13 or figure 3.14 or 3.15 then the computer sets its symbol at any position among 1, 3, 7 and 9.

3.3 Third and fourth move:

a) Option for winning. (Figure: 3.16)

b) Block user from winning. (Figure: 3.17)

c) Randomly play a move. (Figure: 3.18)

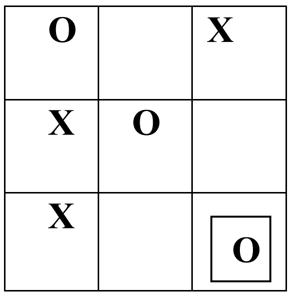


Figure: 3.16

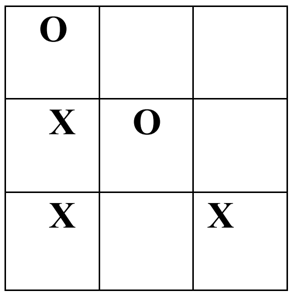


Figure: 3.17

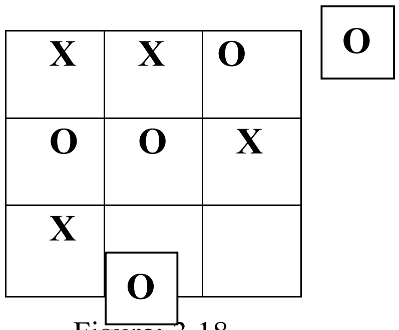


Figure: 3.18

4. Core Logic - Humans:

For each move, check whether any 3 combination is occupied by any player and display the winner accordingly.

5. Classes:

There are two classes in our program .One class is Main.java and another is NewGame.java.

5.1 Class Main:

In this class the main task of the game is done.

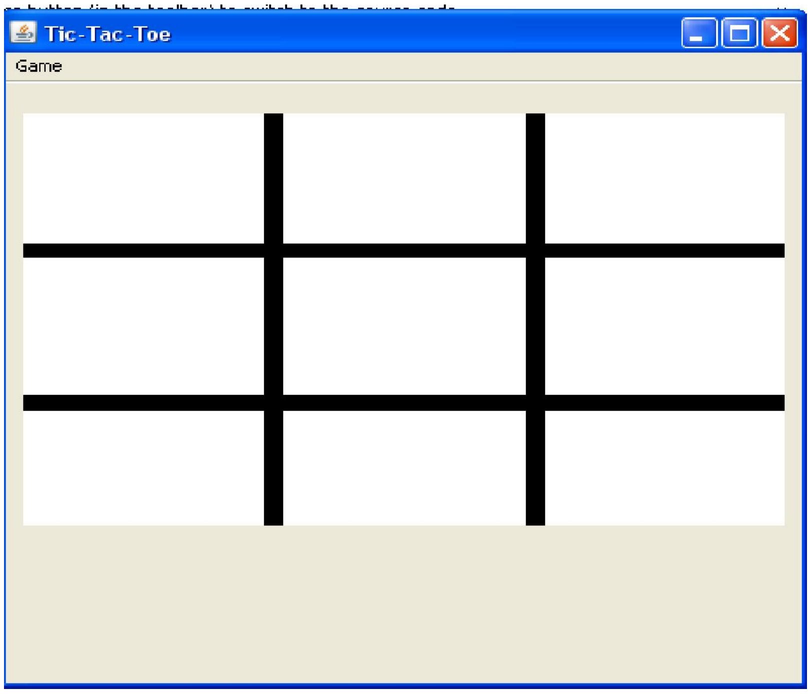
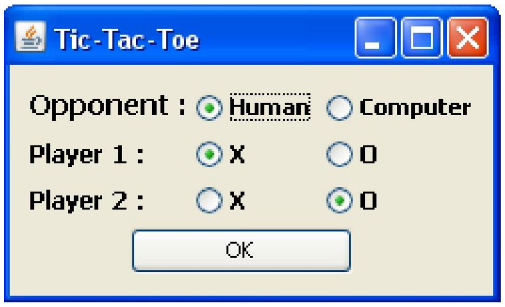


Figure: 5.1

5.2 Class NewGame:

This class is used to show the dialog for choosing options.



Opponent : [✓ Human] Computer
Player 1 : [✓ X] O
Player 2 : X [✓ O]

Figure: 5.2

6. Methods:

The methods we used in our program are as follows:

private void strongLevel ():
To apply artificial intelligence in our program.

private int checkRow (int rownum, String rowname):
To check the combination whether any two symbols(X or O) are same for winning or blocking .

private int checkColumn(int columnnum, String columnname):
To check the combination whether any two symbols(X or O) are same for winning or blocking .

private Boolean checkCorner(String cornername, int count1):
To check the combination whether any two symbols(X or O) are same for winning or blocking.

private void startNewGame():
To display the options for player.

private void setValue(int number, int position):
To set the symbols in the right place for winning and blocking .

private void setElse():
To set the symbols in an empty place randomly.

private void setCorner():
To create scope for computer to win in the third move.

7. Limitations:

1. GUI is not so attractive.

2. Only mouse interface is implemented, keyboard is not activated in the game.

8. Future plan:

1. Keyboard functions will be added.

2. We want to design more complex boards for the game in future.

Reference:

Books:

H.M.Deitel and P.J.Deital, Java How to program: Sixth Edition
Herbert Schildt, The Complete Reference: Fifth edition