

Programming Lab Report

Name: Seif Eldin Mahmoud

ID: 6773

Group 2

Section 1

Tic-tac-toe Game

1-Introduction:

The program is a simulation of the Tic-tac-toe game using Java Programming language with dynamic ability to choose the dimensions where it is either square or rectangle.

The Game starts by asking the user to enter the names of the two players then it told them who is “X” and who is “O”.

After entering the names of the players, the game starts by displaying the Tic-tac-toe game board with the available places to play.

The program consist of two classes:

1. Board Class
2. TestPlaying Class

2-Discussion:

2.1 Board Class:

The Board class contain the main argument and behavior of the Dashboard, which is the following methods:

- Board (constructor)
- DisplayBoard (Displaying of the Board)
- EmptySpace (Check Empty spaces)
- Insert
- CheckHorizontal (Check winning in Rows)
- CheckVertical (Check winning in Column)
- CheckRighDiagonal (Check winning in Diagonal)
- CheckLeftDiagonal (Check winning in Diagonal)
- Play (Running the game)

2.1.1 Board Method:

The Method is a constructor used to locate a place in the memory for the class creating an object of data type `char [][]`, which is a 2D array of characters.

The constructor locates numbers of characters which is an equal number of rows multiplied by the number of columns and filled by “ ” by space which means that this place is Empty.

```
public Board() {
    for(int i=0; i<play.length; i++)
    {
        for (int j = 0; j < play[0].length; j++) {
            this.play[i][j]=' ';
        }
    }
}
```

2.1.2 Display Board Method

This method is used to display the dashboard to the players after each round to tell the available places to play in and to tell them if one of them is going to the next move.

The method is based on looping on each element in the row and print it by spaces between them then after the row ends it creates a new line filled with “---” to give the game a realistic shape that is familiar to the player.

For easy use of the game, the method also creates one Horizontal line at the top of the board, which is the coordinates of the

```
public void DisplayBoard()
{
    for (int i = 0; i < play[0].length; i++) {
        System.out.print("\t" + (i + 1) + "\t");
    }
    System.out.print("\n");
    for (int i = 0; i < play.length; i++) {
        System.out.print("  " + (i + 1) + "  ");
        for (int j = 0; j < play[0].length; j++) {
            System.out.print("\t" + play[i][j] + "\t");
            if (j % (play[0].length - 1) != 0 || j == 0)
                System.out.print('|');
        }
        if (i % (play.length - 1) != 0 || i == 0)
        {
            System.out.print("\n");
            for (int j = 0; j < play[0].length; j++) {
                System.out.print("-----");
            }
            System.out.print("\n");
        }
    }
    System.out.print("\n\n\n");
}
```

columns and writes a number at the beginning of the row, which is the coordinates of the row.

2.1.3 Empty Space Method:

This method is mainly used to check if the coordinates the player entered are empty or field.

The method checks if the entered coordinates are character equal to “ ” space or not if the answer is not it means that this move is played before by one of them.

```
private boolean EmptySpace(int x,int y)
{
    boolean z=true;
    if(play[x][y]!=' ')
        z=false;
    return z;
}
```

2.1.4 Insert Method

The Insert method is responsible to take the coordinates from the players using GUI by creating a floating window asking it the player to enter the row coordinate and the column coordinate.

The method starts by asking the player to enter the coordinates then it checks if the entered coordinates are found or not by return to the size of the array which is represented by “play.lenght” representing the index of rows and “play[0].lenght” representing the index of columns.

```
private void Insert(char z)
{
    int x,y;
    String M=JOptionPane.showInputDialog("Enter Row Coordinates for "+z+": ");
    String N=JOptionPane.showInputDialog("Enter Column Coordinates "+z+": ");
    if(M!=null&&N!=null)
    {
        x=Integer.parseInt(M)-1;
        y=Integer.parseInt(N)-1;
        if(x<play.length&&y<play[0].length)
        {
            if(EmptySpace(x, y))
                play[x][y]=z;
            else
            {
                System.out.println("Error,This move is played before by "+play[x][y]);
                Insert(z);
            }
        }
        else
        {
            System.out.println("The Entered Coordanites is not found");
            Insert(z);
        }
    }
    else
    {
        JOptionPane.showMessageDialog(null,
            "Please,Enter Coordinates","Error",
            JOptionPane.ERROR_MESSAGE);
        Insert(z);
    }
}
```

After checking that the entered coordinates are found the method checks if the entered coordinates are empty or not then assign the entered character.

If any of the previous condition is false, the method recalls itself again after printing to the player that the entered coordinates either are not found or are played before.

2.1.5 Check Horizontal Method:

The method is responsible for check if there are three same characters following each other in the same row or not.

The method takes parameter z of type char that is going to be the method comparing character to all elements in the row.

If the condition is true the method returns true which means that the value in character Z wins.

```
private boolean CheckHorizontal(char z)
{
    boolean win=false;
    for (int i = 0; i < play.length; i++) {
        for (int j = 0; j < play[0].length; j++) {
            if(play[i][j]==z)
            {
                if(j+1<play[0].length&& j+2<play[0].length)
                {
                    if(play[i][j+1]==z&&play[i][j+2]==z)
                        win=true;
                }
            }
        }
    }
    return win;
}
```

2.1.6 Check Vertical Method:

The method is responsible for check if there are three same characters following each other in the same column or not.

```
private boolean CheckVertical(char z)
{
    boolean win=false;
    for (int j = 0; j < play[0].length; j++) {
        for (int i = 0; i < play.length; i++) {
            if(play[i][j]==z)
            {
                if(i+1<play.length&& i+2<play.length)
                {
                    if(play[i+1][j]==z&&play[i+2][j]==z)
                        win=true;
                }
            }
        }
    }
    return win;
}
```

The method takes parameter z of type char that is going to be the method comparing character to all elements in the column.

If the condition is true the method returns true which means that the value in character Z wins.

2.1.7 Check Right Diagonal Method:

This method is responsible for checking if there are three characters following each other diagonally.

```
private boolean CheckRighDiagonal(char z)
{
    boolean win=false;
    for (int i = 0; i < play.length; i++) {
        for (int j = 0; j < play[0].length; j++) {
            if(play[i][j]==z)
            {
                if(i+1<play.length&&i+2<play.length&&j+1<play[0].length&&j+2<play[0].length)
                {
                    if(play[i+1][j+1]==z&&play[i+2][j+2]==z)
                        win=true;
                }
            }
        }
    }
    return win;
}
```

The method takes character Z that is value is either X or O then the function starts checking each row until it finds equivalence then the method compares the element of the second and the third row and column relative to the previous element if they are equivalence the method return true either it returns false.

2.1.7 Check Left Diagonal Method:

This method is responsible for checking if there are three characters following each other diagonally.

```
private boolean CheckLeftDiagonal(char z)
{
    boolean win=false;
    for (int i = 0; i < play.length; i++){
        for (int j = play[0].length-1; j>=0 ; j--) {
            if(play[i][j]==z)
            {
                if(i+1<play.length&&i+2<play.length&&j-1>=0&&j-2>=0)
                {
                    if(play[i+1][j-1]==z&&play[i+2][j-2]==z)
                        win=true;
                }
            }
        }
    }
    return win;
}
```

The method takes character Z that is value is either X or O then the function starts checking each row until it finds equivalence then the method compares the element of the second and the third row and the two previous columns relative to the previous element if they are equivalence the method return true either it returns false.

2.1.8 Play Method:

This Function is responsible for running the game after the object of type Board is created the method starts the game.

First, the method asks the players to enter their names then it starts showing them the dashboard and asks the first player to enter the coordinates of its move according to the loop as the loop rounds until the

max number of rounds which is equal to the multiplication of the number of rows and columns.

The round of the first player takes the even round as it is divisible by 2 (Round%2=0) and the second player takes the odd rounds as it is not divisible by 2 (Round%2!=0).

The methods used all the previous methods, which will help it to identify the winner according to the algorithm if the win variable is false that means no one win and it is true if it changes to win the method will print the winner's name and break the loop ending the game.

```
public void Play()
{
    String player1,player2;
    player1=JOptionPane.showInputDialog("Enter Player 1 Name:");
    player2=JOptionPane.showInputDialog("Enter Player 2 Name:");
    if(player1!=null&&player2!=null)
    {
        System.out.println(player1+" is O\n"+player2+" is X\n");
        DisplayBoard();
        boolean win=false;
        for (int i = 0; i < play.length*play[0].length; i++) {
            if(i%2!=0)
            {
                char x='X';
                Insert(x);
                if(i>3)
                {
                    if(CheckHorizontal(x)||CheckVertical(x)||CheckRighDiagonal(x)||CheckLeftDiagonal(x))
                    {
                        DisplayBoard();
                        win=true;
                        JOptionPane.showMessageDialog(null,player2+" Win!!");
                        break;
                    }
                }
            }
        }
    }
}
```

```
        else if (i%2==0)
        {
            char x='O';
            Insert(x);
            if(i>3)
            {
                if(CheckHorizontal(x)||CheckVertical(x)||CheckRighDiagonal(x)||CheckLeftDiagonal(x))
                {
                    DisplayBoard();
                    win=true;
                    JOptionPane.showMessageDialog(null,player1+" Win!!");
                    break;
                }
            }
        }
        DisplayBoard();
    }
    if(!win)
    {
        JOptionPane.showMessageDialog(null,"Draw");
        return;
    }
    return;
}
else
    Play();
}
```

2.2 TestPlaying Class:

This class is the main class that uses to run the previous class “Board”.

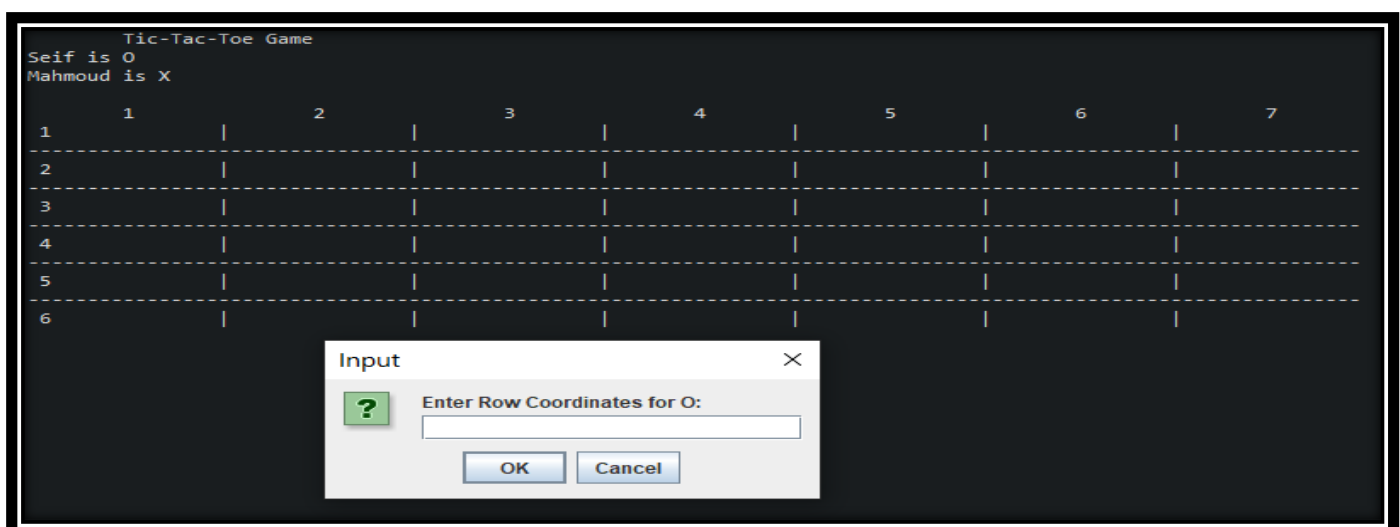
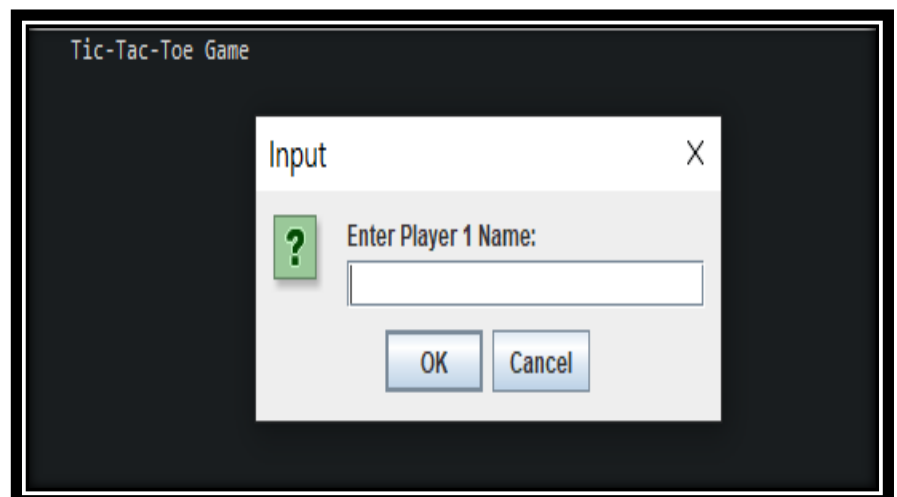
```
public class TestPlaying {  
    public static void main(String[] args) {  
        System.out.println("\tTic-Tac-Toe Game");  
        Board play= new Board();  
        play.Play();  
    }  
}
```

The class starts by allocating place in the memory and run the constructor of the board creating an object then it uses the object to call method play that will start the game.

3-Conclusion:

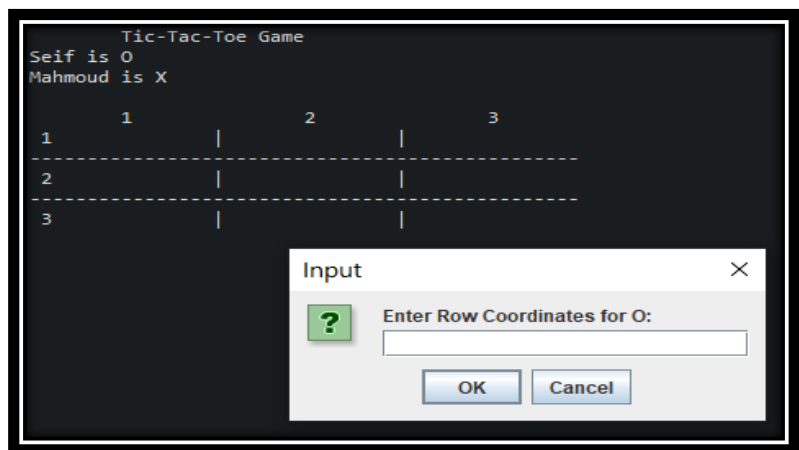
The conclusion of the program is simple to run of the program and its algorithm, methods working and do their functions.

After entering the

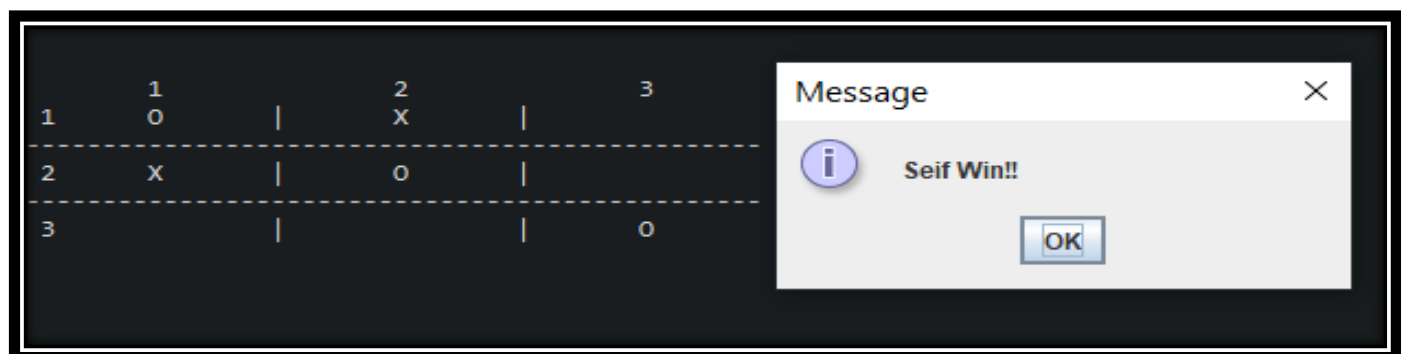


names of the players

Besides, the code is dynamic could be at any size the players could choose any dimension to play with.



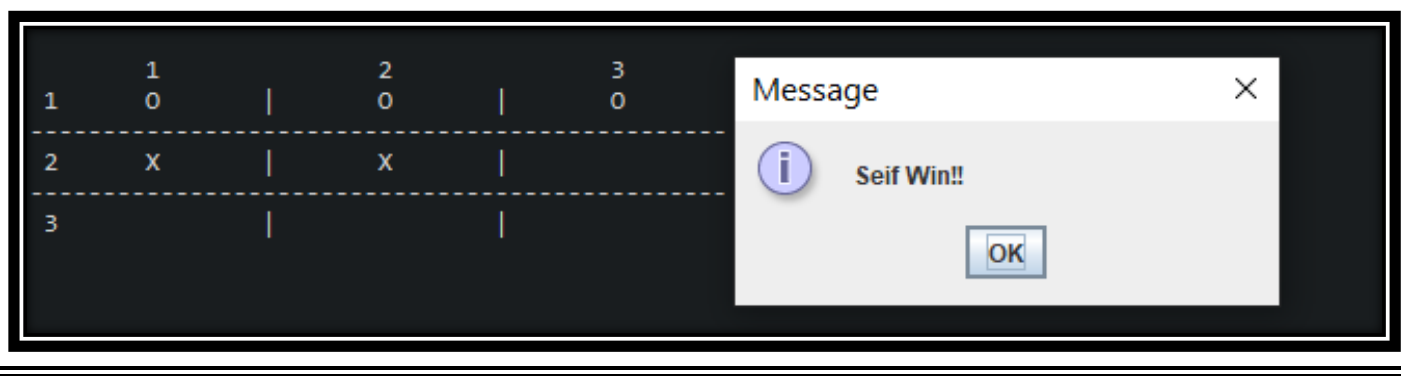
The Right Diagonal Algorithm.



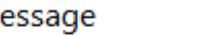
The Left Diagonal Algorithm.



The Horizontal Algorithm.





The Vertical Algorithm.



The Draw Algorithm.

Message

 Draw



The algorithm also works for any dimension as the array is dynamic could be changed by changing the number of elements as the shown figure.

