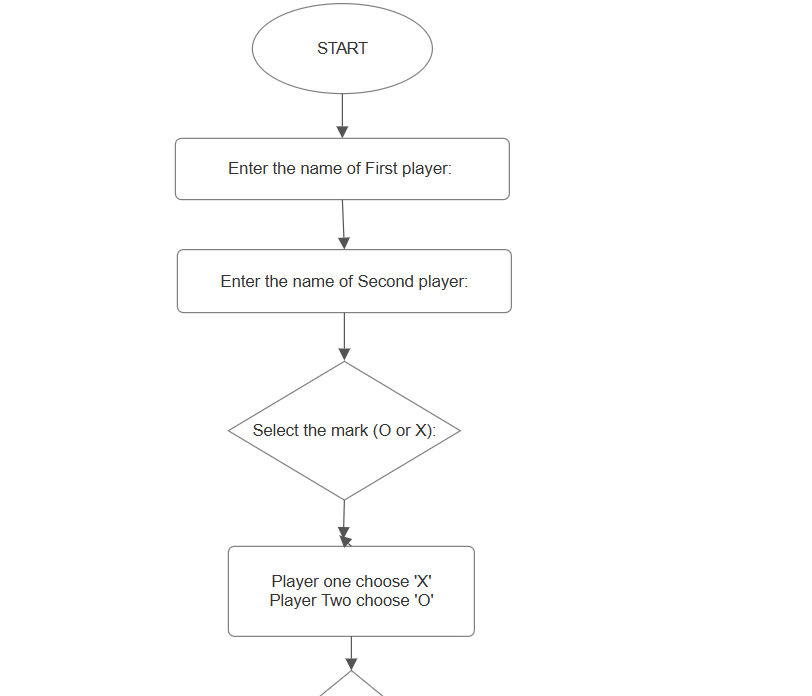
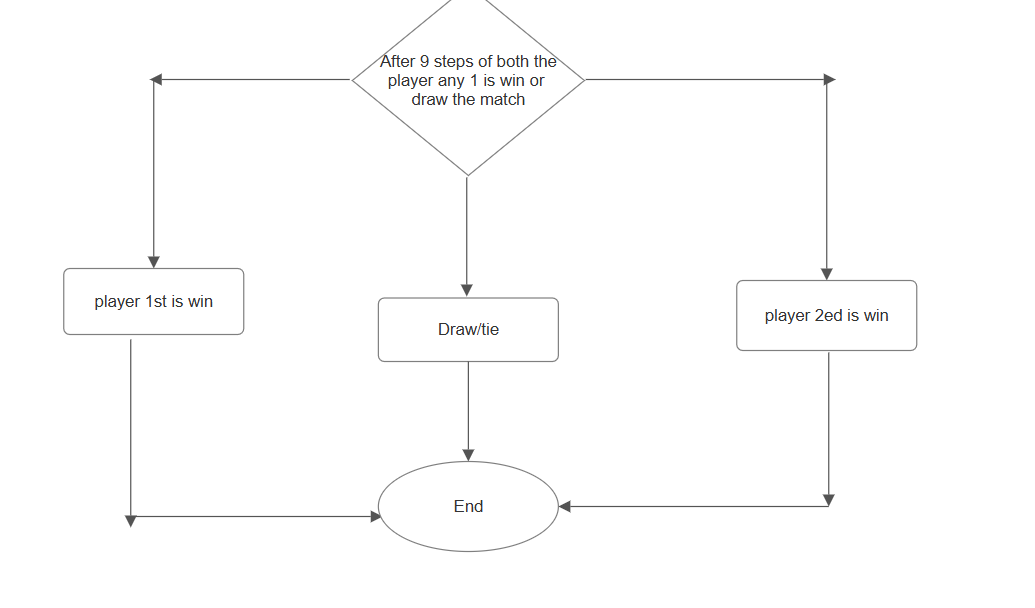
1.UML Diagram

|  |
| --- |
| Tic tac Toe Game |
| - SIZE : int  -table : int[][] |
| + main(args : String[])  - players: Player[2]  - initialize Table()  - display Table()  - update Board(row: int, col: int, | | symbol: char): boolean  - is Cell Empty (row: int, col: int): Boolean  - name: string  - symbol: char  - make Move(row: int, col: int): Move  - row: int  - col: int  - player: Player  - validate Move(): boolean |

**2.Flowchart**





3.Code of World\_Search Game

import java.util.Scanner;

public class TicTocgame{

    char tictoc[][]=new char[3][3];

    public static void Display(char tictoc[][]){

        for(int i=0;i<3;i++){

            for(int j=0;j<3;j++){

                System.out.print(tictoc[i][j]+" ");

            }

            System.out.println();

        }

    }

    static void Replace(char arr[][], char find, char replace){

        for(int i=0;i<3;i++){

            for(int j=0;j<3;j++){

                if(arr[i][j] == find){

                    arr[i][j] = replace;

                    return;

                }

            }

        }

    }

    public boolean CheckforWin(){

        return(CheckforRow() || CheckforColumn() || CheckforDiagonal());

    }

    public boolean Check(char c1, char c2, char c3){

        return (c1 == c2 && c2 == c3);

    }

    public boolean CheckforRow(){

        for(int i=0;i<3;i++){

            if(Check(tictoc[i][0], tictoc[i][1], tictoc[i][2])){

                return true;

            }

        }

        return false;

    }

    public boolean CheckforColumn(){

        for(int i=0;i<3;i++){

            if(Check(tictoc[0][i], tictoc[1][i], tictoc[2][i])){  // Fixed here

                return true;

            }

        }

        return false;

    }

    public boolean CheckforDiagonal(){

        return(Check(tictoc[0][0], tictoc[1][1], tictoc[2][2]) || Check(tictoc[0][2], tictoc[1][1], tictoc[2][0]));

    }

    public static void main(String[] args) {

        TicTocgame game = new TicTocgame();

        Scanner sc = new Scanner(System.in);

        String player1, player2;

        char player1mark, player2mark;

        System.out.print("Enter the name of First player: ");

        player1 = sc.nextLine();

        System.out.print("Enter the name of Second player: ");

        player2 = sc.nextLine();

        System.out.println(player1 + " Select the mark (O or X): ");

        player1mark = sc.next().charAt(0);

        while(player1mark != 'X' && player1mark != 'x' && player1mark != 'O' && player1mark != 'o'){

            System.out.print("Invalid input, please enter valid input (O or X): ");

            player1mark = sc.next().charAt(0);

        }

        if(player1mark == 'X' || player1mark == 'x'){

            player2mark = 'O';

        } else {

            player2mark = 'X';

        }

        // Initialize the board with 1 to 9

        int counter = 0;

        for(int i=0;i<3;i++){

            for(int j=0;j<3;j++){

                game.tictoc[i][j] = Character.forDigit(++counter, 10);

            }

        }

        Display(game.tictoc);

        char input;

        for(int i=0;i<9;i++) {

            if(i % 2 == 0){

                System.out.print(player1 + " Turn: ");

                input = sc.next().charAt(0);

                Replace(game.tictoc, input, player1mark);

            } else {

                System.out.print(player2 + " Turn: ");

                input = sc.next().charAt(0);

                Replace(game.tictoc, input, player2mark);

            }

            Display(game.tictoc);

            // Check for a winner after every move

            if(game.CheckforWin()){

                if(i % 2 == 0){

                    System.out.println(player1 + " wins!");

                } else {

                    System.out.println(player2 + " wins!");

                }

                return;  // End the game

            }

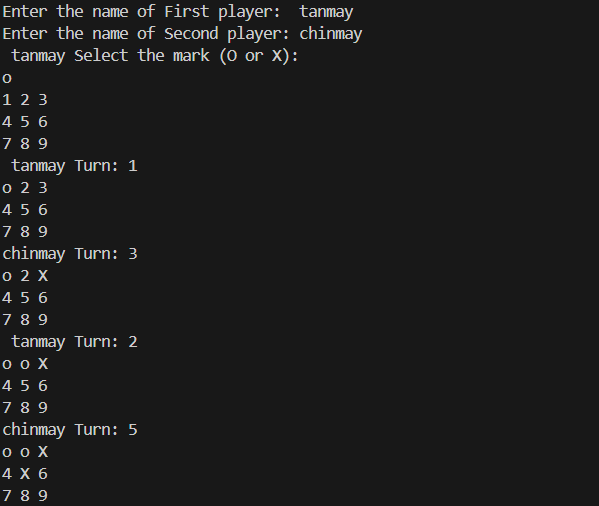
        }

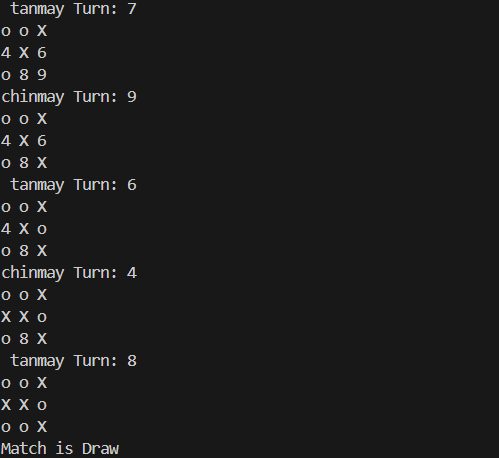
        System.out.println("Match is Draw");

    }

}

**4.OUTPUT**

****



**5.Explanation of code**

* **Class and Attributes:**
  + A Tic Toc game class is created with a 3x3 character array tic toc to represent the Tic-Tac-Toe board.
* **Display Method:**
  + The static Display method prints the Tic-Tac-Toe board's current state.
* t iterates through the tic toc array and prints each element in a 3x3 grid format**.**
* **Replace Method:**
  + The static Replace method searches the arr array for a specific character (find) and replaces it with another character (replace).
  + This is used to place the player's mark on the board in the desired cell.
  + It stops after finding and replacing the first occurrence to ensure a single replacement per turn.
* **Check for Winning Conditions:**
  + Three methods are used to check for a win condition:
    - Check for Row: Checks if any row has three matching characters.
    - Check for Column: Checks if any column has three matching characters.
    - Check for Diagonal: Checks both diagonals for three matching characters.
  + Check for Win combines these methods to return true if any of the winning conditions is met.
* **Check Method:**
  + Check is a helper method that verifies if three characters are equal. It’s used by the row, column, and diagonal checks.
* **Main Method and Game Setup:**
  + main method is the entry point of the program.
  + A Scanner object sc is used to capture player inputs.
  + It prompts the players to enter their names and lets the first player choose their mark (either 'O' or 'X').
* The mark chosen by the first player (player1mark) determines the second player's mark (player2mark).
* **Board Initialization:**
  + The board is initialized with numbers 1 through 9, which represent cell positions for easier selection during gameplay.
* **Gameplay Loop:**
  + A for-loop runs for up to 9 turns, allowing each player to take turns.
  + Players are prompted to enter a position (1-9) on their turn.
  + The Replace method is called to mark the selected position on the board with the current player’s mark.
  + The Display method is called after each move to show the updated board.
* **Win Check and Game End:**
  + After each move, Check for Win is called to check if the current player has won.
  + If a player wins, it prints the winner's name and ends the game.
  + If no winner is found after 9 moves, it declares the game a draw.

**CONCLUSION**

This Java code implements a simple two-player Tic-Tac-Toe game. Players take turns choosing a mark ('X' or 'O') and a cell on the 3x3 grid, with the first player selecting their mark and the second player receiving the opposite one. The game board is displayed after each move, and the program checks for a winner after each turn by verifying rows, columns, and diagonals. If a player wins, the game ends and announces the winner; if no winner is found after 9 moves, the game results in a draw. The program ensures valid inputs and manages the flow of the game turn by turn.