1. **Formulating the Problem**

**1.1 Problem Description**

Design and implement a TicTacToe game using Java. The program should store the game board in a two dimensional array of char type. The players should be able to input their moves at the same keyboard.

**1.2** **Verbalization**

What is the goal?

Build a Tic-tac-toe game using Java.

What are the givens?

The move (location) inputs from the players.

What are the unknowns?

If there is a winner or not. If there is a winner then who is the winner?

**1.3 Information Elicitation**

Goal Tic-Tac-Toe game

Givens Location

Unknowns Winner

Condition Three X's or O's in a row - horizontally- vertically - diagonally

**2. Planning the Solution**

**2.1. Solution Strategy**

Get a move from the user and then place the char (X or O depending on the current player) in a two dimensional array. Compare the adjacent index values to determine whether there are three X's or O's in a row.

**2.2 Goal Decomposition**

Sub-goal 1

Create the game board.

Sub-goal 2

Get move from user.

Sub-goal 3

Store the input.

Sub-goal 4

Compare the adjacent values.

Sub-goal 5

Display the winner.

**2.4 Data Organization and Description**

**Inputs**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Origin** | **Used in Sub-goal #** |
| xMove | Player X's move | User | 2 |
| oMove | Player O's move | User | 2 |

**Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Origin** | **Used in Sub-goal #** |
| Winner | The winner (X or O) of the game | Screen | 4 |

**3. Designing the Solution**

**3.1 Structure Chart**

First Level Decomposition

Goal Refinement

**Sub-goal 1**

Get player names.

**Sub-goal 2**

Display the game board.

**Sub-goal 3**

Get input from the user.

**Sub-goal 3.1**

Store the input.

**Sub-goal 3.2**

Display the updated board.

**Sub-goal 4**

Compare the values in the 2D array to determine if there are three X's or O's in a row.

**Sub-goal 4.1**

Determine the winner.

**Sub-goal 5**

Display the winner.

**Sub-goal 6**

Clear the game board.

Second Level Decomposition

**3.2 Module and Data Specifications**

**Name:** displayBoard - Display the board on the screen.

**Input:** None.

**Output:** None.

**Logic:** Print out the board using the values stored in the array.

**Name:** clearBoard - clears the board.

**Input:** None.

**Output:** None.

**Logic:** Reinitialize the array to its original state.

**Name:** addMove - store the user move .

**Input:** x - the user's move.

player - the player who made the move.

**Output:** If the location is already occupied with a X or and O request the user to make a valid move.

**Logic:** Determine the location of where the user wants to make the move, if that location is empty store the move in that location. Otherwise, request the user to make a valid move.

**Name:** playerTurn - Determine which player's turn it is .

**Input:** None .

**Output:** None .

**Logic:** If it is player X's turn return 'X', otherwise return 'O'.

**Name:** changeTurn - Alternate the players turn.

**Input:** None.

**Output:** None.

**Logic:** If the current player is X, change the Boolean value of player X to false and change player O's Boolean value to true. If the current player is O, change the Boolean value of player O to false and change player X's Boolean value to true.

**Name:** winner - Display the winner.

**Input:** None.

**Output:** Ex: "John is the winner".

**Logic:** If there are three X's in a row display a message stating player X is the winner. If there are three O's in a row display a message stating player O is the winner.

**Name:** gameOver - Determine if there is a winner.

**Input:** None.

**Output:** None.

**Logic:** Compare the adjacent values in the array and check if there are three X's or O's in a row.

**Name:** main- Instantiate the name of Tic-Tac-Toe.

**Input:** None.

**Output:** None.

**Logic:** Play a single game of Tic-Tac-Toe.

**Data:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Structure** |
| **gameBoard** | **Char** | **Two-Dimensional Array** |
| **playerX** | **String** | **Variable** |
| **PlayerO** | **String** | **Variable** |
| **playerXTurn** | **Boolean** | **Variable** |
| **playerOTurn** | **Boolean** | **Variable** |
| **xMove** | **Integer** | **Variable** |
| **oMove** | **Integer** | **Variable** |
| **numMove** | **Integer** | **Variable** |

**3.3 Algorithm**

Logic

1.0 Get player's names.

1.1 Get player X's name.

1.2 Get player O's name .

2.0 Display instructions for the user.

2.1 Display the game board.

3.0 Get current player's move.

4.0 Store the player's move.

4.1 Display the updated board.

5.0 Compare the values in the array.

5.1 Determine the winner.

6.0 Repeat steps 2.1 to 5.1 until there is a winner or the total number of moves exceeds 9.

7.0 Display Result .

7.1 Display Game Over.

7.2 Display the winner name.

Algorithm Description

The program starts off by getting the player names. The player names are used to create a new game. Once the game is created instructions for the user are displayed on the screen followed by the initial game board. Then the current player, by default X is the first player, will be asked to make a move by entering the number of the location where the player wants to add his or her move.

1 | 2 | 3

--------------------

4 | 5 | 6

--------------------

7 | 8 | 9

Once the move is collected from the user, it is stored in the corresponding index of the array. The updated board is displayed before the other player is asked to make a move. Each time a move is added, the array is checked to determine if there is a winner. The users are asked to make moves until either there is a winner or if the game exceeds nine moves. If there is a winner, a message containing the winner's name is displayed on the screen. If there is not winner then a message is displayed informing the players that the game is over.

**4 Translation**

**4.1 Source Code**

//===================================================

// Name : Nidhi Patel

// SID : 31379144

// Course : IT114

// Section : 452

// Instructor : Maura Deek

// Assignment # : Programming Assignment 1

// Date : 02/19/2016

// Description : This program is a TicTacToe game

//===================================================

//for keyboard input

import java.util.Scanner;

public class TicTacToe

{

//Two-Dimensional array to store the moves

char[][] gameBoard = new char[3][3];

static String playerX, //playerX stores playerX's name

playerO; //playerO stores playerO's name

Boolean playerXTurn = true, //playerXTurn is true when it is playerX's turn

playerOTurn = false; //playerOTurn is true when it is playerO's turn

/\*\*

Constructor with parameters

pX player X's name

pO player O's name

\*/

public TicTacToe(String pX,String pO)

{

playerX = pX;

playerO = pO;

gameBoard[0][0] = '1';

gameBoard[0][1] = '2';

gameBoard[0][2] = '3';

gameBoard[1][0] = '4';

gameBoard[1][1] = '5';

gameBoard[1][2] = '6';

gameBoard[2][0] = '7';

gameBoard[2][1] = '8';

gameBoard[2][2] = '9';

}

//Prints the game board on the screen

public void displayBoard()

{

System.out.println(" " +gameBoard[0][0]+" " + "|" + " " +gameBoard[0][1]+" " + "|" + " " +gameBoard[0][2]);

System.out.println(" " + "|" + " " + "|" + " ");

System.out.println("--------------------");

System.out.println(" " +gameBoard[1][0]+" " + "|" + " " +gameBoard[1][1]+" " + "|" + " " +gameBoard[1][2]);

System.out.println(" " + "|" + " " + "|" + " ");

System.out.println("--------------------");

System.out.println(" " +gameBoard[2][0]+" " + "|" + " " +gameBoard[2][1]+" " + "|" + " " +gameBoard[2][2]);

System.out.println(" " + "|" + " " + "|" + " ");

}

//Reinitialize the array

public void clearBoard()

{

gameBoard[0][0] = '1';

gameBoard[0][1] = '2';

gameBoard[0][2] = '3';

gameBoard[1][0] = '4';

gameBoard[1][1] = '5';

gameBoard[1][2] = '6';

gameBoard[2][0] = '7';

gameBoard[2][1] = '8';

gameBoard[2][2] = '9';

}

/\*\*

Add the move

x location of the move

player player(X or O)

\*\*/

public void addMove(int x, char player)

{

switch(x)

{

case 1:

if(gameBoard[0][0] == 'X' || gameBoard[0][0] == 'O' )

{

System.out.println("Please enter a valid choice\n");

break;

}

else

{

gameBoard[0][0] = player;

break;

}

case 2:

if(gameBoard[0][1] == 'X' || gameBoard[0][1] == 'O' )

{

System.out.println("Please enter a valid choice\n");

break;

}

else

{

gameBoard[0][1] = player;

break;

}

case 3:

if(gameBoard[0][2] == 'X' || gameBoard[0][2] == 'O' )

{

System.out.println("Please enter a valid choice\n");

break;

}

else

{

gameBoard[0][2] = player;

break;

}

case 4:

if(gameBoard[1][0] == 'X' || gameBoard[1][0] == 'O' )

{

System.out.println("Please enter a valid choice\n");

break;

}

else

{

gameBoard[1][0] = player;

break;

}

case 5:

if(gameBoard[1][1] == 'X' || gameBoard[1][1] == 'O' )

{

System.out.println("Please enter a valid choice\n");

break;

}

else

{

gameBoard[1][1] = player;

break;

}

case 6:

if(gameBoard[1][2] == 'X' || gameBoard[1][2] == 'O' )

{

System.out.println("Please enter a valid choice\n");

break;

}

else

{

gameBoard[1][2] = player;

break;

}

case 7:

if(gameBoard[2][0] == 'X' || gameBoard[2][0] == 'O' )

{

System.out.println("Please enter a valid choice\n");

break;

}

else

{

gameBoard[2][0] = player;

break;

}

case 8:

if(gameBoard[2][1] == 'X' || gameBoard[2][1] == 'O' )

{

System.out.println("Please enter a valid choice\n");

break;

}

else

{

gameBoard[2][1] = player;

break;

}

case 9:

if(gameBoard[2][2] == 'X' || gameBoard[2][2] == 'O' )

{

System.out.println("Please enter a valid choice\n");

break;

}

else

{

gameBoard[2][2] = player;

break;

}

default:

System.out.println("Please enter a valid choice\n");

}

}

//Return the char that represents which player's turn it is

public char playerTurn()

{

if(playerXTurn)

{

return 'X';

}

else

{

return 'O';

}

}

//Alternate the players turn

public void changeTurn()

{

if(playerXTurn)

{

playerOTurn = true;

playerXTurn = false;

}

else

{

playerXTurn = true;

playerOTurn = false;

}

}

//Display the winner

//The player who made the last move is the winner

public void winner()

{

if(gameOver()){

if(playerTurn() == 'X'){

System.out.println(playerO + " is the winner!");

}

else

{

System.out.println(playerX + " is the winner!");

}

}

}

/\*\*

Checks if there is a winner

If there is a winner, true is returned

Otherwise false is returned

\*\*/

public boolean gameOver()

{

if(gameBoard[0][0] == gameBoard[1][0] && gameBoard[0][0] == gameBoard[2][0])

{

return true;

}

else if(gameBoard[0][0] == gameBoard[0][1] && gameBoard[0][0] == gameBoard[0][2])

{

return true;

}

else if(gameBoard[0][0] == gameBoard[1][1] && gameBoard[0][0] == gameBoard[2][2])

{

return true;

}

else if(gameBoard[0][1] == gameBoard[1][1] && gameBoard[0][1] == gameBoard[2][1])

{

return true;

}

else if(gameBoard[1][0] == gameBoard[1][1] && gameBoard[1][0] == gameBoard[1][2])

{

return true;

}

else if (gameBoard[0][2] == gameBoard[1][2] && gameBoard[0][2] == gameBoard[2][2])

{

return true;

}

else if (gameBoard[2][0] == gameBoard[2][1] && gameBoard[2][0] == gameBoard[2][2])

{

return true;

}

else if (gameBoard[2][0] == gameBoard[1][1] && gameBoard[2][0] == gameBoard[0][2])

{

return true;

}

else

{

return false;

}

}

/\*\*

Main method that creates a new TicTacToe object and gets input from the user

\*\*/

public static void main (String[]args)

{

int xMove, //Player X's move

oMove, //Player O's move

numMove = 0; //Number of total moves made

//Create a Scanner object to recieve input from the user

Scanner keyboard = new Scanner (System.in);

//Prompt the player X to enter their name and store the input in playerXName

System.out.println("Player X Please Enter Your Name: ");

playerX = keyboard.nextLine();

//Prompt the player O to enter their name and store the input in playerOName

System.out.println("\nPlayer O Please Enter Your Name: ");

playerO = keyboard.nextLine();

System.out.println("\nLets play Tic-Tac-Toe!\n");

//Declare and instantiate a TicTacToe object using the player names

TicTacToe newGame = new TicTacToe(playerX,playerO);

//Give instructions of how to play this game to the user

System.out.println("To make a move please enter the number corresponding to that location\n" +

"For example, if you want to make your move in the center box enter the number 5\n");

//Diplay the game board

newGame.displayBoard();

System.out.println();

/\*\*

While loop iterates until the game is not over and until the number of moves is less than or equal

to 9.(Maximum 9 moves allowed per game)

\*\*/

while(!newGame.gameOver())

{

//If it is player X's turn prompt player X to enter their move and store the move in the gameBoard array

if(newGame.playerTurn() == 'X')

{

System.out.print(playerX + " make your move: ");

xMove = keyboard.nextInt();

newGame.addMove(xMove,'X');

}

//If it is player O's turn prompt player O to enter their move and store the move in the gameBoard array

else

{

System.out.print(playerO + " make your move: ");

oMove = keyboard.nextInt();

newGame.addMove(oMove,'O');

}

//Display the board after every move

System.out.println();

newGame.displayBoard();

System.out.println();

//Increase the number of moves after each move is made

numMove++;

//If the game exceeds the maximum number of moves

//Inform the players that the game is over

if(numMove >= 9)

{

System.out.println("Game Over!");

break;

}

//If the gameOver method returns true, display the winner name

//and clear the board.

else if(newGame.gameOver()){

System.out.println("Game Over!");

newGame.winner();

}

//Switch turn

newGame.changeTurn();

}

newGame.clearBoard();

}

}

**4.2 Program and Module Description**

**displayBoard**

The board is displayed on the screen containing the same values as the array.

clearBoard

Reinitializes the board to how it was at the beginning by assigning numbers 1 through 9 in each index of the array.

addMove

Takes the number of the location where the user wants to make the move and the player who is making the move as inputs. Using the number of the location, the move is added to the corresponding array index. If the user wants to make a move which has already been made an error message is displayed.

playerTurn

Tells whose turn it is.

changeTurn

Alternated the turns of the player based on who the current player is.

winner

Displays the winner if there is a winner.

gameOver

Determines whether there is a winner or not by comparing the values in the array.

main

First of all the main method asks the users to enter their names in. With the names collected from the user a new game is created. Instructions are displayed on the screen for the user and initial game board is displayed. Then the player is asked to input a move. The move is stored in the two- dimensional array. Each time a move is added the board is displayed to the user and the values in the array are compared to check if there is a winner. The steps of asking the players to input their move, storing the move, displaying the updated version of the board and comparing the array values are repeated until there is a winner or the game runs out of moves using a while loop. Lastly, the results of the game are displayed to the user. If there is a winner then the winner name is displayed otherwise "Game Over! " is displayed.

**5. Solution Testing**

**Test Case 1**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Player X** | Sam | | | | | | | | |
| **Player O** | John | | | | | | | | |
| **Player** | Sam | John | Sam | John | Sam | John | Sam | John | Sam |
| **Move** | 1 | 5 | 2 | 3 | 7 | 4 | 9 | 6 |  |
| **Output** |  |  |  |  |  |  |  | Game Over!  John is the winner! |  |

**Test Case 2**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Player X** | David | | | | | | | | |
| **Player O** | Sally | | | | | | | | |
| **Player** | David | Sally | David | Sally | David | Sally | David | Sally | David |
| **Move** | 3 | 6 | 5 | 7 | 2 | 1 | 8 |  |  |
| **Output** |  |  |  |  |  |  | Game Over!  David is the winner! |  |  |

**Test Case 3**

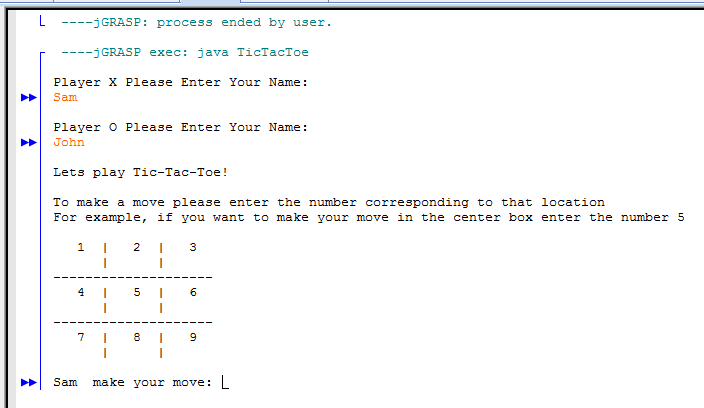
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Player X** | Amy | | | | | | | | |
| **Player O** | Cady | | | | | | | | |
| **Player** | Amy | Cady | Amy | Cady | Amy | Cady | Amy | Cady | Amy |
| **Move** | 7 | 3 | 1 | 4 | 5 | 9 | 6 | 2 | 8 |
| **Output** |  |  |  |  |  |  |  |  | Game Over! |

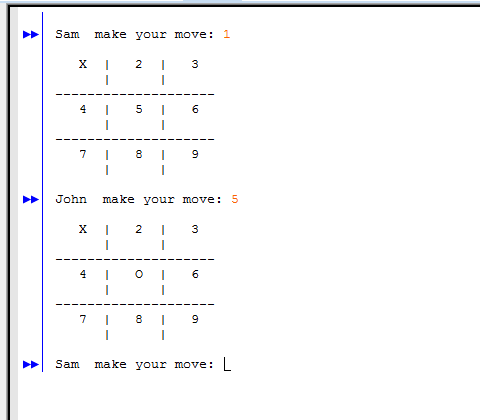
**Test Case 4**

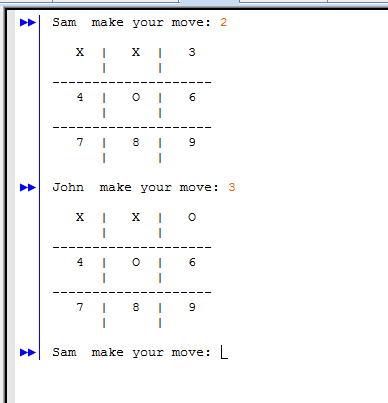
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Player X** | Sara | | | | | | | | |
| **Player O** | Jimmy | | | | | | | | |
| **Player** | Sara | Jimmy | Sara | Jimmy | Sara | Jimmy | Sara | Jimmy | Sara |
| **Move** | 1 | 3 | 1 |  |  |  |  |  |  |
| **Output** |  |  | Please enter a valid choice |  |  |  |  |  |  |

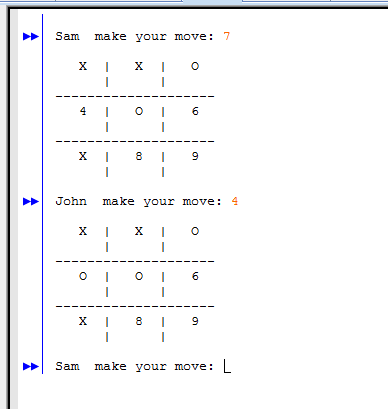
**6 Testing Output**

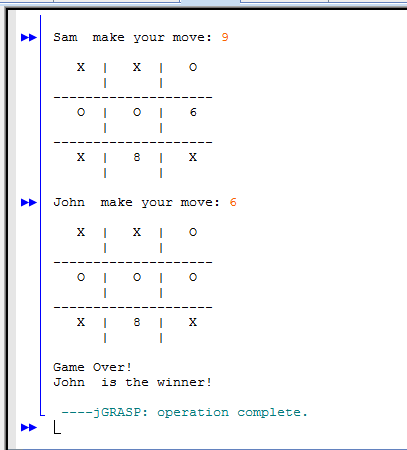
**Test Case 1**

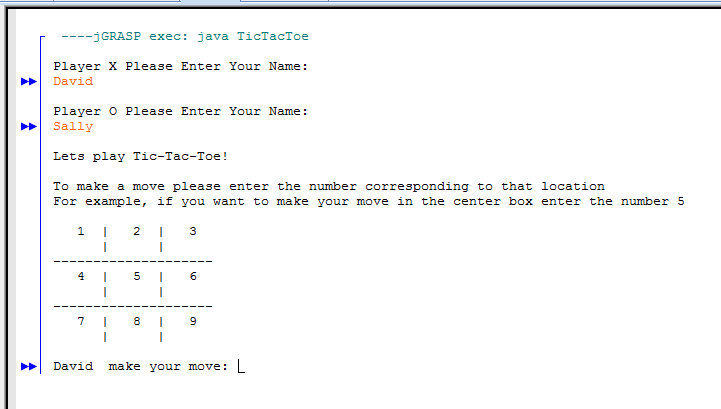
****

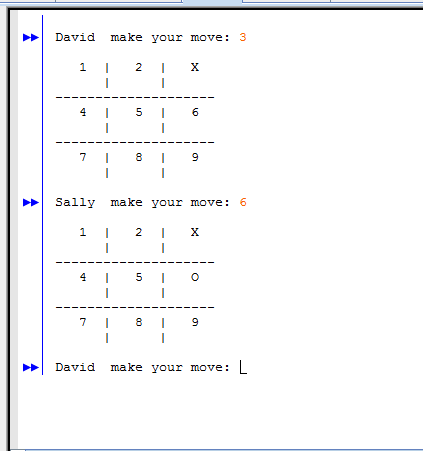
****

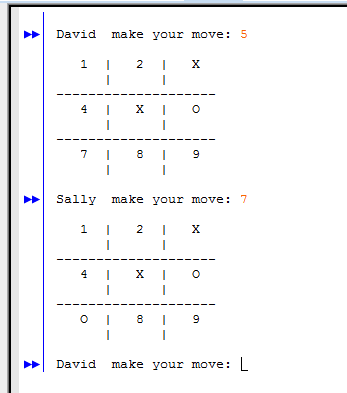
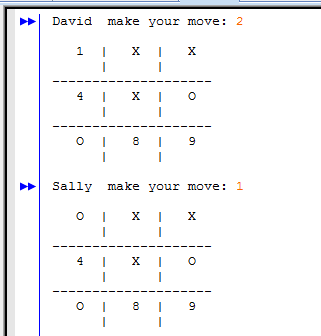
****

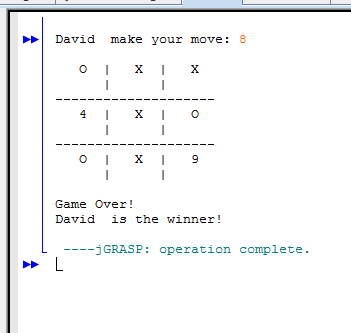
****

****

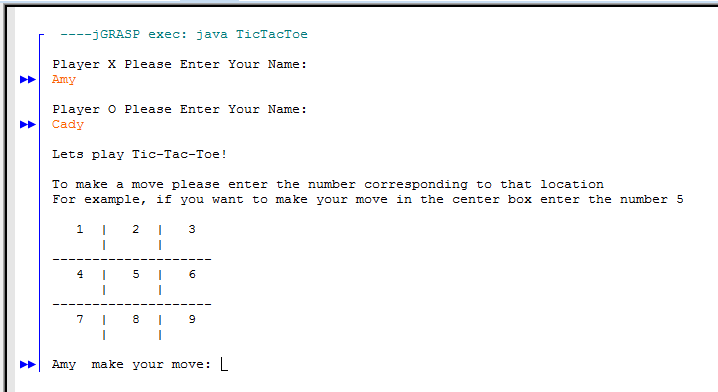
**Test Case 2 **

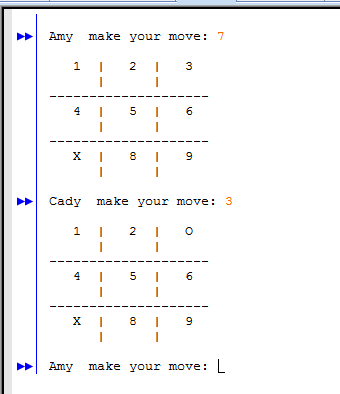
****

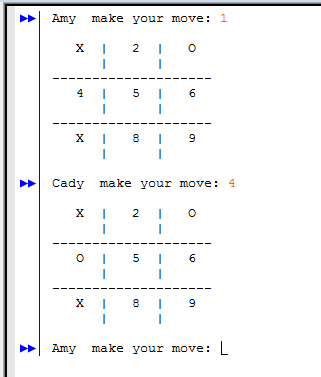
** **

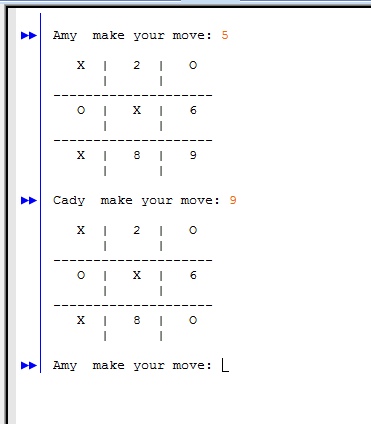
****

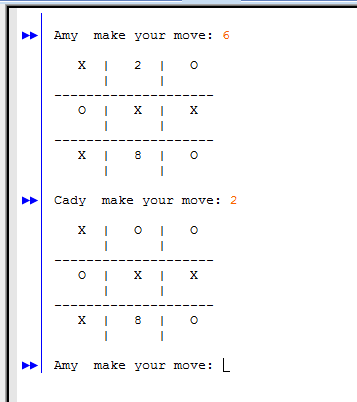
**Test Case 3**

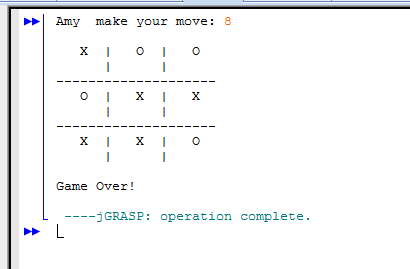
****

****

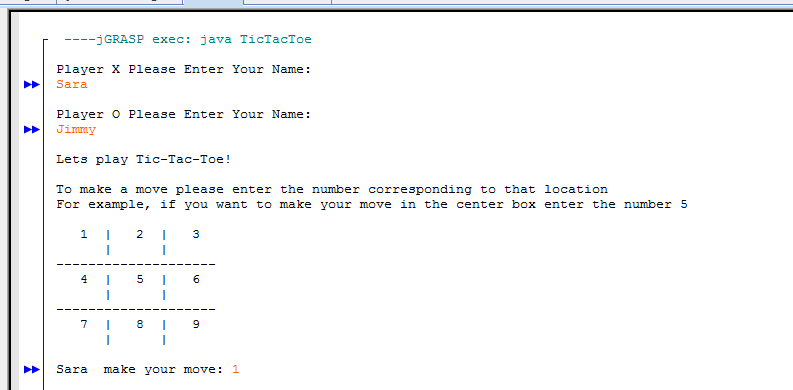
****

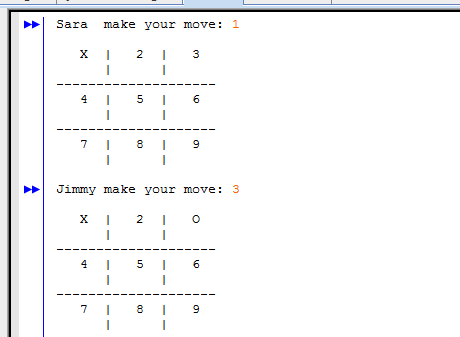
****

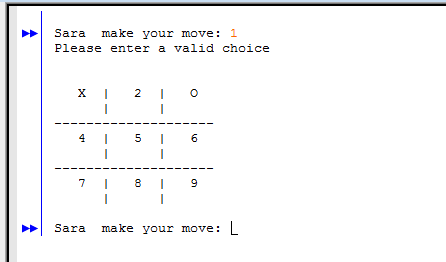
****

****

**Test Case 4**

****

****

****