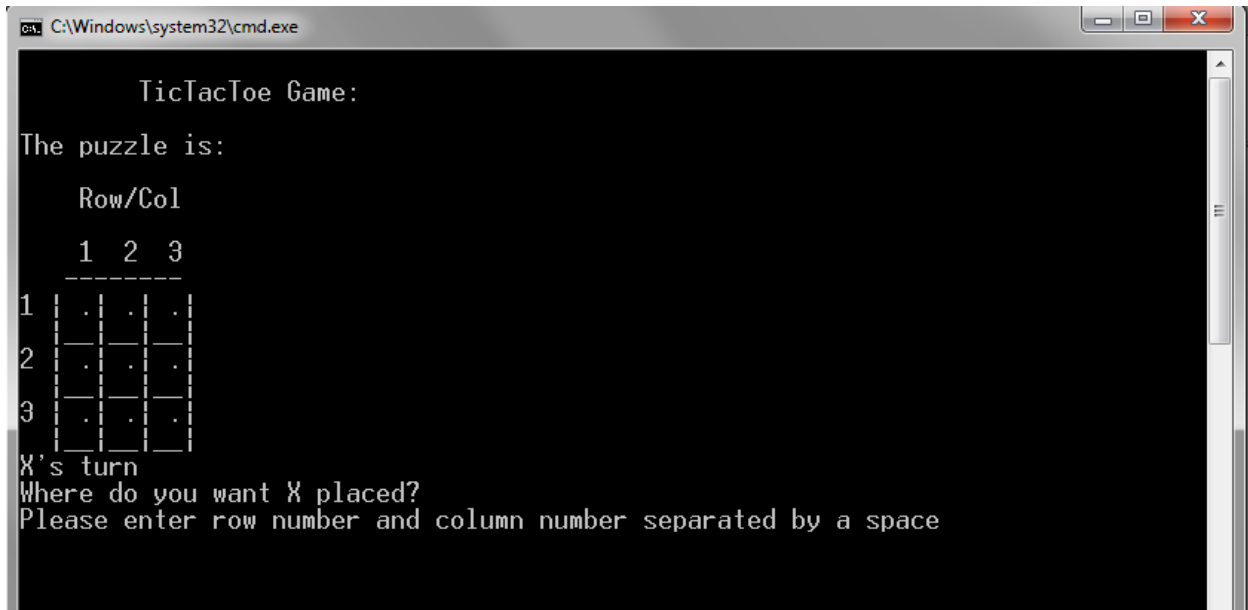


# CIT 149: Java I

## Chapter 7

### Programming Assignment 2

Complete #7 on page 564. Name the class TicTacToe.java. When the program is run it will display as following:



```
C:\Windows\system32\cmd.exe

    TicTacToe Game:
The puzzle is:
    Row/Col
      1  2  3
-----
1 | . | . | . |
2 | . | . | . |
3 | . | . | . |
X's turn
Where do you want X placed?
Please enter row number and column number separated by a space
```

```
C:\Windows\system32\cmd.exe

TicTacToe Game:

The puzzle is:

  Row/Col
  1 2 3
-----
1 | . | . | . |
2 | . | . | . |
3 | . | . | . |

X's turn
Where do you want X placed?
Please enter row number and column number separated by a space
1 1
The puzzle now is:
  Row/Col
  1 2 3
-----
1 | X | . | . |
2 | . | . | . |
3 | . | . | . |

O's turn
Where do you want O placed?
Please enter row number and column number separated by a space
2 3
The puzzle now is:
  Row/Col
  1 2 3
-----
1 | X | . | . |
2 | . | . | O |
3 | . | . | . |

X's turn
Where do you want X placed?
Please enter row number and column number separated by a space
_
```

At the end the program will declare either a winner or a tie.

```

Where do you want U placed?
Please enter row number and column number separated by a space
3 1
The puzzle now is:
  Row/Col
    1 2 3
  -----
1 | X | X | 0 |
2 | X | 0 | 0 |
3 | 0 | . | X |
  |___|___|___|

0 is the WINNER!
Do you want to play again (yes/no)?

```

```

The puzzle now is:
  Row/Col
    1 2 3
  -----
1 | X | X | X |
2 | 0 | 0 | . |
3 | . | . | . |
  |___|___|___|

X is the WINNER!
Do you want to play again (yes/no)?
-

```

Notice that in this last screen shot I purposely had X win.

I'm sure many of you are worried about this assignment. Here are some hints to help you along. Although the publisher does give me solutions to the assignments in the book, I wrote the program separate of theirs, using the Sudoku puzzle and removing what I didn't need and altering some other things. Here are the things I did:

- What type of data will the board hold? We do not need integers but characters so I changed the board to char type rather than int type. I figured I did not need a start board for this assignment so I deleted that array entirely. Since I changed the array from a int type to a char type and remove the start board I recompiled the program expecting to see quite a few errors. I read the errors and made the changes until it compiled again. Makes it simpler this way. Some changes I had to make was in places where I had int value, I had to change to char value, and everything it mentioned the start array I removed entirely.

- I figured I would definitely need the toString() method but I do not need 9 rows and columns. I only needed 3 so I adjusted the board accordingly, first removing the numbers after 3 and keeping only 3 of the |\_\_| at the bottom. I also changed my for loops from <9 to < 3. On the hyphens I estimated how many I would need but adjusted them as I went along.
- Next I went down to the addInitial() and addGuess() method and thought do I really need these? I decided that since I did not have any initial values and I wasn't guessing anything, I could delete both of these methods entirely.
- Next was the getValueIn() method. This method never was really used in the Sudoku program so I deleted it as well. Get rid of the garbage is my way of thinking and leave only the necessary code.
- Do I need the reset() method. This was an easy decision since the directions specifically say that you give the user the choice as to whether or not to play again. If they want to play again we would need to reset the board. The only change I made in this method was the last line of code. I changed: board[i][j] = start[i][j]; simply to board[i][j] = '.'; and changed the for loop to go only to 3 instead of 9.
- For the isFull and getAllowedValues I had to think this over. I decided I did not need either of them. Since we have only 3 rows/columns we can easily use another way of detecting whether the board is full and since there are only Xs and Os I did not need to checked every row and column for the values of numbers.
- The next 4 methods I decided I did need because I will need to check to see if there are 3 Xs or Os in a row, column or diagonal. I did change the checkSub to checkDiagonal just because I like to name my methods so I know what they will be doing. So do I need the same code in each of these methods? Of course not. For the checkPuzzle we do not need a for loop at all. We do need if and if/else statements though. Instead of the looksGood = looksGood && checkRow(i); etc. I decided that my other 3 methods would not need parameters since I only had an X or O to work with. So I used if and else if statements to check what is returned by each of the other 3 methods.
- For the checkRow method, of course, I remove the int row. In the method I wanted to see how little code I could get away with. I knew I needed a for loop but it couldn't be like the one in the Sudoku. So I wrote down all the possible locations for the rows that correspond to the board and came up with the following:

```
public boolean checkRow()
{

    // See if there are either 3 Xs or 3 Os in any row

    for(int i = 0;i< 3 ;i++)
        if(board[i][0] == 'X' && board[i][1] == 'X' && board[i][2] == 'X')
        {
            setWinner('X');
            return true;
        }
        else if(board[i][0] == 'O' && board[i][1] == 'O' && board[i][2] == 'O')
        {
            setWinner('O');
            return true;
        }
    }
```

```

        return false;
    }

```

- I'll get to the setWinner() I used in a minute. The checkCol() method is similar except where I have [i][0] etc. I would reverse them to [0][i] because the column will change.
- For the checkDiagonal(), or checkSub() if you kept that name, I had to do separate if and if/else statements. There are only four so that wasn't a problem.
- The next method I had to decide on was the initializePuzzle() method. We do not have an initial values so I deleted this one right away.
- That leaves the main method. But before we go there I'll go back to the reference of a setWinner() method. I wanted to make things as easy on myself as possible. So I created a mutator and accessor method for the winning player. Both are extremely easy and do not have any excess code. Just write them as simple as possible.
- Now for the main method. If you look back at the directions you see that X goes first. The only thing that users will input is the row and column. So my first thought was what did I need to do to get by with as little code as possible. I prefer less code, for less confusion later. Also it's easier to read. So this is what I came up with:

- - First going by the Sudoku Puzzle I created a TicTacToe object.
  - Like Sudoku I created integers for the row and column I would enter.
  - Since I know we cannot go beyond 9 entries I decided to simplify things by creating an integer to keep track of the number of entries.
  - I also created a char data type and set it to X since X is the first player.
  - I needed to display the puzzle so I used a simple System.out.println(puzzle); Where puzzle is the name of my TicTacToe object.
  - I remove the code for whether I wanted to set, get or quit because I did not need them. I simply displayed whose turn it was using the value of the char variable. For example if I named it value I had:

```
System.out.println(value + "'s turn\nWhere do you want " + value + " placed?");
```

- I then asked the user to enter the row and column just like in Sudoku.
- To see whether that space was occupied I simply used an if statement as in:

```

if(puzzle.board[row][col] == 'X' || puzzle.board[row][col] == 'O')
{
    System.out.println("That space is already taken, try again");
}

```

- Else I changed the board to that value and displayed the puzzle again as I did previously.
- I also decided that since I would be asking the same questions for O I would use the same code by using an if statement of:

```

if(value == 'X')
    value = 'O';
else
    value = 'X';

```

- I also needed to keep track of the number of times I went through the puzzle so I added one to the integer for counting.

- Next I needed to check to see if I had a winner or not and also whether the integer was equal to 9 yet. So I used:

```
if(puzzle.checkPuzzle())
{
    System.out.println("\n" + puzzle.getWinner() + " is the WINNER!");
    break;
}

else if(count == 9)
{
    System.out.println("\nNo winner and board is full");
}
```

- Starting with the code of `System.out.println(value + "'s turn\nWhere do you want " + value + " placed?");` and ending here I put the code within a loop. Think about what type of loop. I wanted the code to run at least once. This is your clue as to which loop you want.
- After the end of that loop I decided that it had to be a nested loop since I wanted to give the user the choice of whether to play again. I needed to create a variable first though. So I went back up to where I created my other variables and created a boolean and set it to yes. Then I put a while loop around the other loop.
- After the other loop ended I asked the user whether they wanted to play again and told them to enter yes if they did. If they answered yes I set the boolean to equal yes.

This was long but it should give you an idea of where to go and help you complete the assignment.

Compress the `TicTacToe.java` and `TicTacToe.class` into a single zip or rar file and submit to the appropriate drop box.