**Tic – Tac – Toe Interview Challenge**

The goal of this challenge is for a candidate to show off their coding prowess, it is designed to be fun and not too challenging. The application, working or not, will then be a point of discussion in the interview.

The challenge is to build a simple Tic-Tac-Toe command line application.

This can be written in either Java or Python

Please email a copy of the code to [sam.doherty@reydonsports.com](about:blank) in either a zip attachment or provide a link to a publicly accessible GitHub repo.

Time allotted: 5-6 Hours

**Minimum Requirements**

* The game runs in the command line. We are more concerned with the internal design than a beautiful GUI.
* Input is passed in the form of coordinates e.g. "1,1" to place the symbol in the middle of the board.
* Players play on a 3x3 board, with three consecutive symbols required to win the game.
* Two human players can play against each other on the same computer.
* A board is printed after each player moves.
* The game ends when one of the players wins or the board is full.

**Bonus Requirements**

* Boards bigger than just 3x3 are supported.
* Use of TDD

**Example Output when running (Edit as you see fit):**

“Run Command - Java/ Python”

* Welcome to my Tic Tac Toe Game!

0 1 2

0 [ ][ ][ ]

1 [ ][ ][ ]

2 [ ][ ][ ]

Player X Go:

0,0

* Player X chose 0,0

0 1 2

0 [X][ ][ ]

1 [ ][ ][ ]

2 [ ][ ][ ]

Player O Go:

0,1

* Player O chose 0,1

0 1 2

0 [X][ ][ ]

1 [O][ ][ ]

2 [ ][ ][ ]

Player X Go:

1,1

* Player X chose 1,1

0 1 2

0 [X][ ][ ]

1 [O][X][ ]

2 [ ][ ][ ]

Player O Go:

0,2

* Player O chose 0,2

0 1 2

0 [X][ ][ ]

1 [O][X][ ]

2 [O][ ][ ]

Player X Go:

2,2

* Player X chose 2,2

0 1 2

0 [X][ ][ ]

1 [O][X][ ]

2 [O][ ][X]

Player X WINS!!!!!!

Application exists

**Tic – Tac – Toe Interview Challenge (Response)**

AGILE Requirements:

* User Stories
* User Acceptance Tests
* Workflow
* Requirements (Details)
* Wireframes

Template reference:

<https://orthogonal.io/insights/what-makes-a-good-requirement-document-for-an-agile-project/#:~:text=A%20good%20requirement%20document%20for,requirements%20in%20details%20and%20wireframes.>

**Define Tic Tac Toe** <https://en.wikipedia.org/wiki/Tic-tac-toe>

A game in which two players seek in alternate turns to complete a row, a column, or a diagonal with either three O's or three X's drawn in the spaces of a grid of nine squares; noughts and crosses.

**USER STORIES**

*This states all of the scenarios of the users involved. These should read:*

*As a SOME ROLE,  
I want to DO SOMETHING,  
So that I CAN GET SOME BENEFIT*

*The user stories are critical to lay out exactly who is going to do what, and for what reason(s).*

**Scenario 1 (Human vs Human)**

* *As a Human Player 1,*
* *I want to play a game of tic tac toe such that I go first on a board of HxW size against another human*
* *So that when I place* three of my marks in a horizontal, vertical, or diagonal row (adjacent) I am declared the winner
* *As a Human Player 2,*
* *I want to play a game of tic tac toe such that I go second on a board of HxW size against another human opponent*
* *So that when I place* three of my marks in a horizontal, vertical, or diagonal row (adjacent) I am declared the winner

**Scenario 2 (Human vs AI)**

* *As a Human Player ,*
* *I want to play a game of tic tac toe such that I can select if I go first or second on a board of HxW size against an AI opponent*
* *So that when I place* three of my marks in a horizontal, vertical, or diagonal row (adjacent) I am declared the winner
* *As an AI Player ,*
* *I want to play a game of tic tac toe such that I can select if I go either second or first depending on the human choice on a board of HxW size against an AI opponent*
* *So that when I place* three of my marks in a horizontal, vertical, or diagonal row (adjacent) I am declared the winner

**Scenario 3 (AI vs AI)**

* *As a human observer,*
* *I want arrange for 2 AI opponents to play a number of game of tic tac toe against each other*
* *So that I can review the outcomes in order to determine the overall winner*

* *As an AI Player 1,*
* *I want to play a game of tic tac toe such that I go first on a board of HxW size against another AI*
* *So that when I place* three of my marks in a horizontal, vertical, or diagonal row (adjacent) I am declared the winner
* *As an AI Player 2,*
* *I want to play a game of tic tac toe such that I go second on a board of HxW size against another AI*
* *So that when I place* three of my marks in a horizontal, vertical, or diagonal row (adjacent) I am declared the winner

**USER ACCEPTANCE TESTS**

*These should include all scenarios outlined in the user stories. These should not be too detailed (they don’t need to mention specific screens or a complete list of actions to execute the steps). These should read:*

*GIVEN that condition 1 and condition 2….  
WHEN I do step 1, and step 2…  
THEN, desired result 1, desired result 2….*

*These define a set of actual scenarios a tester could walk through to assert that the feature is complete. These are not detailed test scripts that you find in UAT. They are meant to convey a set of tests that all involved can walk through to understand how the feature will work.*

**Scenario 1 Start a (Human vs Human) Game**

**GIVEN**, I start the instance, such that 2 Humans want to play a game of tic tac toe, on a board of HxW size

**WHEN**, I initialize a program with parameters, Humans=2” , height = H, Width = W

**THEN**, The console should display :

- a tic tac toe board with W cell in width , H cells in height

-an indicator will then show the players who’s turn it is(PLAYER 1 /PLAYER 2) ,and their marker(X / O)

-The player to go first will be labelled PLAYER 1

-A series of valid move will be entered manually by each human player, until the victory conditions are met

**Scenario 2 Start a (Human vs AI) Game**

**GIVEN**, I start the instance, such that 1 Humans wants to play a game of tic tac toe, on a board of HxW size

**WHEN**, I initialize a program with parameters, Humans=1,goFirst=TRUE/FALSE , height = H, Width = W

**THEN**, The console should display:

- a tic tac toe board with W cell in width , H cells in height

-an indicator will then show the player who’s turn it is(PLAYER 1 /PLAYER 2) ,and their marker(X / O)

-The player to go first will be labelled PLAYER 1

-If the AI went first the board will have the AI’s marker displayed and the Human will be labelled Player 2

-After each valid move entered manually by a human player, the AI will make a turn until the victory condition are met

**Scenario 3 Start a (AI vs AI) Game**

**GIVEN**, I start the instance, such that 0 Humans wants to play a game of tic tac toe, on a board of HxW size

**WHEN**, I initialize a program with parameters, Humans= 0,goFirst=TRUE/FALSE , height = H, Width = W

**THEN**, The console should display :

- a tic tac toe board with W cell in width , H cells in height

-an indicator will then show the player who’s turn it is(PLAYER 1 /PLAYER 2) ,and their marker(X / O)

-The player to go first will be labelled PLAYER 1

-A series of valid move will be made automatically by each Ai player, until the victory conditions are met

**Playing the game**

**Given** 2 players are playing tic tac toe

**WHEN** a player enters the coordinates (x,y) , and confirms it as a move ,and it is valid

**THEN**, The console should display:

- The tic tac toe board with the players marker (X/O) in the coordinates entered

-an indicator will then show the Player who’s turn it is(PLAYER 1 /PLAYER 2) ,and their marker(X / O)

**WHEN** a player enters the coordinates (x,y) , and confirms it as a move ,and it is not valid

- The tic tac toe board will be displayed without the invalid move

-A message will show in the console that the move was not valid

-An indicator will then inform the correct Player who’s turn it is(PLAYER 1 /PLAYER 2) ,and their marker(X / O)

**Winning The Game**

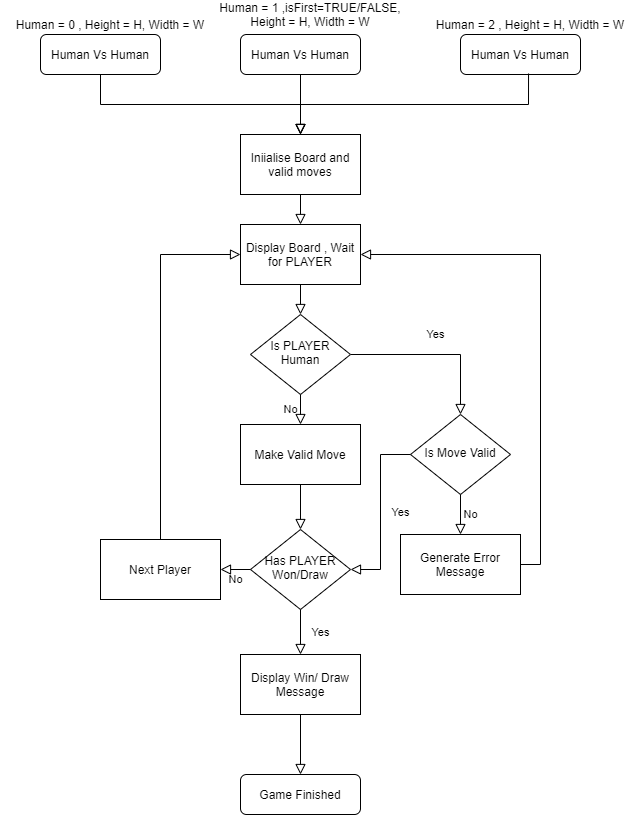
**GIVEN** That the players successfully enter valid moves to play a game of tic tac toe

**WHEN** a player who enters a valid coordinate such that they have three of their markers adjacent in a horizontal, vertical, or diagonal row

**THEN** , the player to submit the move that satisfies the winning condition will be declared the winner

**WORKFLOW**

*This should include a picture of the screens involved. Error states and view changes based on role should be documented. This picture is worth a thousand words, as the details of the flow through the feature can be quite complex, and this is hard to spell out the details in the next section.*

****

**REQUIREMENTS (DETAILS)**

*These are the details of the feature. Document all screen(s) and every field, label, validation, message, and action. This is essentially the functional specification of the details of the screen(s) involved. Because it is in the context of the wireframe (next section), it is more concise. You can simply reference the field name, rather than verbosely state everything about the field. You can keep the details to field length, required, etc.*

**Minimum Requirements**

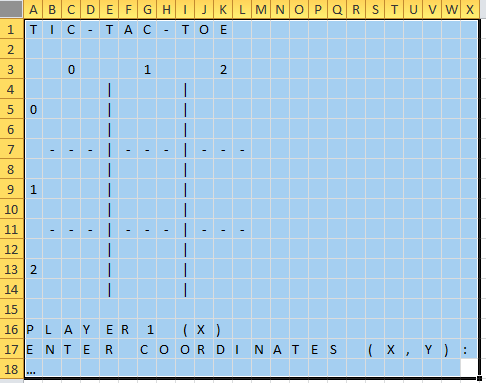
* The game runs in the command line. We are more concerned with the internal design than a beautiful GUI.
* Input is passed in the form of coordinates e.g. "1,1" to place the symbol in the middle of the board.
* Players play on a 3x3 board, with three consecutive symbols required to win the game.
* Two human players can play against each other on the same computer.
* A board is printed after each player moves.
* The game ends when one of the players wins or the board is full.

**Bonus Requirements**

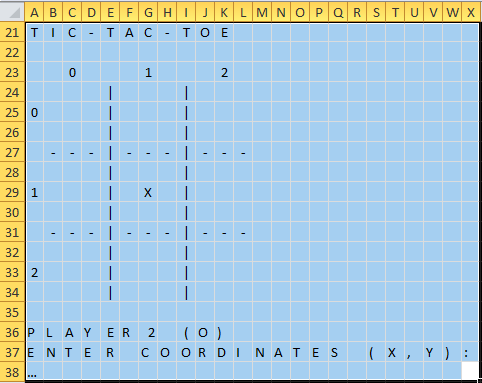
* Boards bigger than just 3x3 are supported.
* Use of TDD
* *Make the AI play itself like in* [*Wargames*](https://www.imdb.com/title/tt0086567/)

**Wireframes***A picture is required for each screen involved. Wireframes can be simple drawings on a whiteboard that are photographed or a set of boxes created in Visio or OmniGraffle (or anything really). Some are photoshop-ed, and others are HTML.*

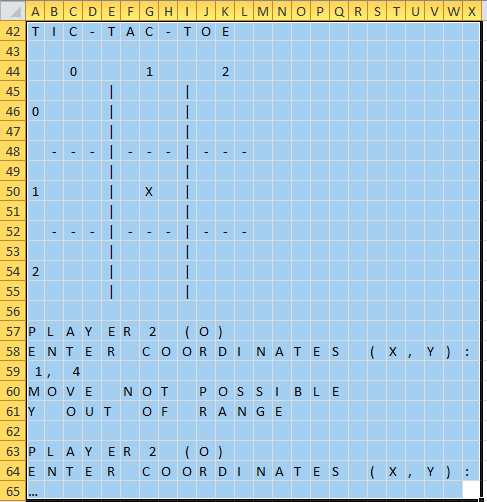
**3x3 Game Board First Turn**



**3x3 Game Board Turn 2**

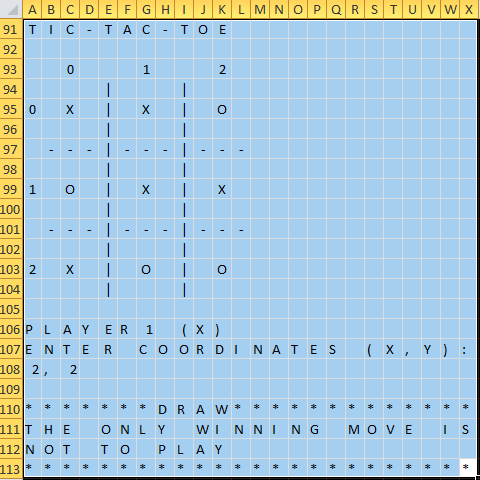


**3x3 Game Board Invalid Move Example**

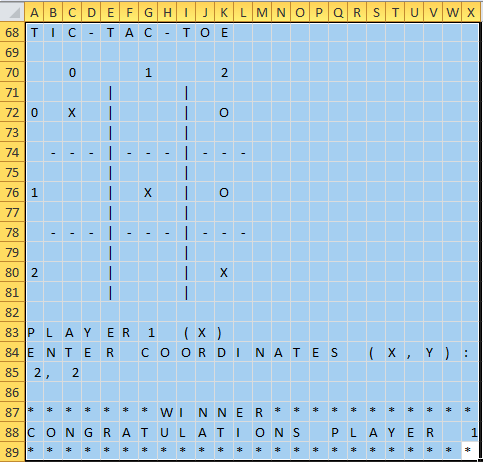


Error Message relevant to invalid move

**3x3 Game Board Draw**



**3x3 Game Board Winner**



**4x4** **Game Board Winner**

