**Describing the language features that you have used to solve the solution, and explaining how the solution conforms to the stated paradigm.**

In this solution, the tic-tac-toe used the classes and object feature to separate the code into three different classes which are board, player, and game. These classes can represent the game board, a player, and the overall game. The solution has applied the list comprehension to create a 3x4 grid board used in the game. The conditional statements (if statements) also have been applied to identify when the player wins, loses, and draws and some other game logic to ensure the game is on the right track. For the user input part, the code has applied error handling (try and except) to make sure when the user inputs a number that is out of range number or inputs some invalid details (letter or symbol). The game will block them, show the error message, and allow the player to enter again. The tic-tac-toe also uses the loops to check the winning condition, so after the user inputs, the game will use the loop to check if the game wins or draws in every round. Also, the game uses a loop to support the player's play round to round to the end of the game.

This version of the tic-tac-toe is suitable for this Object-Oriented (OO) paradigm because the tic-tac-toe uses the classes and objects to create three classes: board, player, and game. This solution also uses the constructor to make the class into the object. Also, this solution used association because the program makes the class into the object and uses it in other classes. For example, inside the game class, use the board object and player object, and those object functions used in the game class. These principles of classes, objects, and association align with the main tenets of the OO paradigm, making this program conform to the OO paradigm.