Player.java

1. Create a new Java Application project,
2. Add a new class called Player.java,
3. Declare a String field called name to hold the player’s name,
4. Create an empty constructor,
5. Declare getter and setter for the name field.

Cell.java

1. Create a class called Cell.java that extends the JButton class to create each cell as a button object,
2. Declare two integer fields called row and col to hold the cell row and column,
3. Declare a boolean field called isEmpty to contain the status of the cell object,
4. Create getters and setters for the row and col fields,
5. Create three methods for the isEmpty field: two to set its value true or false and one to return its current value.

TicTacToeGame.java

1. Add a new class to the project called TicTacToe.java that implements the ActionListener event from the AWT class,
2. Import necessary libraries,
3. Declare integer constants for the number of rows, columns and players,
4. Declare integer constants for the computer and human player,
5. Declare and initialize app name variable,
6. Declare variable to hold number of cells used,
7. Declare an array of class Player to hold the players,
8. Declare a two-dimensional array of class Cell to hold the game cells,
9. Declare an enumeration to hold the cross (x) and circle (o) letters,
10. Declare and initialize a variable of type Letter enum to hold the current player,
11. Declare and initialize a JFrame object to hold the game’s window,
12. Declare a JLabel to show who’s turn is it,
13. Declare a new object of Random class type,
14. Create an empty constructor,
15. Create an initialization method,
16. Instantiate and initialize the Player array,
17. Get the player’s name,
18. Iterate through the cell array add an ActionListener to each and add it to the frame,
19. Configure and show the frame window,
20. Create a method called launch to randomly select the initial player.
    1. If the computer is to play first, call the method that handles this,
    2. Otherwise update the current player and wait for the user to click a cell button,
21. Create a method called computerPlay,
22. Get random numbers for the row and column and try to update the according cell,
    1. If the cell selected is empty, fill it and check if there is a winner,
       1. If there’s no winner, wait for the user to play,
       2. Else, clear the cell array and ask the user to play again,
    2. If not, get new random numbers and loop again,
23. Create a method that returns the number of cells used to check for a tie,
    1. Loop though the array and increment a variable for each filled cell,
24. Create a method to check for a winner or a tie,
    1. If there is a winner, let the user know who won and prompt to play again,
    2. If not, check if there’s a tie,
    3. If there’s no tie, return false to the calling method to keep playing,
25. Create a method called repeat to check if the user wants to play again,
    1. If the user wants to play again, call the clear method,
    2. Otherwise exit the game,
26. Create a method called clear to iterate through the cell object array and set each cell to empty,
27. Override the actionPerformed method to handle the user’s clicks,
    1. If the user clicks an empty cell, fill the cell and check if there is a winner,
       1. If the checkWin method returns false, it means that we can keep playing, call the method to let the computer play,
    2. If the cell is already used, let the user know.

GameLauncher.java

1. Create a new tester class.
2. Declare a main method,
   1. Create a new object of type TicTacToeGame,
   2. Call the initialization method of the object.