CPSC-442X Python Programming

Assignment 3: Tic-Tac-Toe GUI

Due: April 5th, 2016 at 11:59 PM

For this assignment you need to implement the Tic-tac-toe game in Python using tkinter GUI under the following guidelines:

1. Create four classes `Player`, `Deck`, `TicTacToe`, and `DataAccess`.
2. For the class `Player`, it must have the following attributes:
   - `name` (e.g. Alice, X, O …etc.)
   - `id` integer value used in the table Player (e.g. 1, 2, 3 …etc.)
   - `playing_mark` (X or O),
   - `statistics` is a dictionary with three items: won, drawn, lost

   It should implement the methods:

   - `__init__`: the class constructor
   - `get_score()` which should return `((won * 2) + draw - lost)`.
   - And for comparison, it should implement the `__lt__` operator to compare players score.
3. For the class `Deck`, it must have the following data attributes:
   - `board`: a 3×3 board implemented as a list or any other data structure.
   - `player1_choices`: a list or any other data structure that contains the indexes of cells that player 1 choose.
   - `player2_choices`: a list or any other data structure that contains the indexes of cells that player 2 choose.

   It should also implement the following methods:

   - `__init__`: the class constructor
4. The class `DataAccessLayer` must have the following attributes:
   - `conn`: A connection object that connects to the database tic_tac_toe.db.
   - `cur`: A cursor object to execute queries

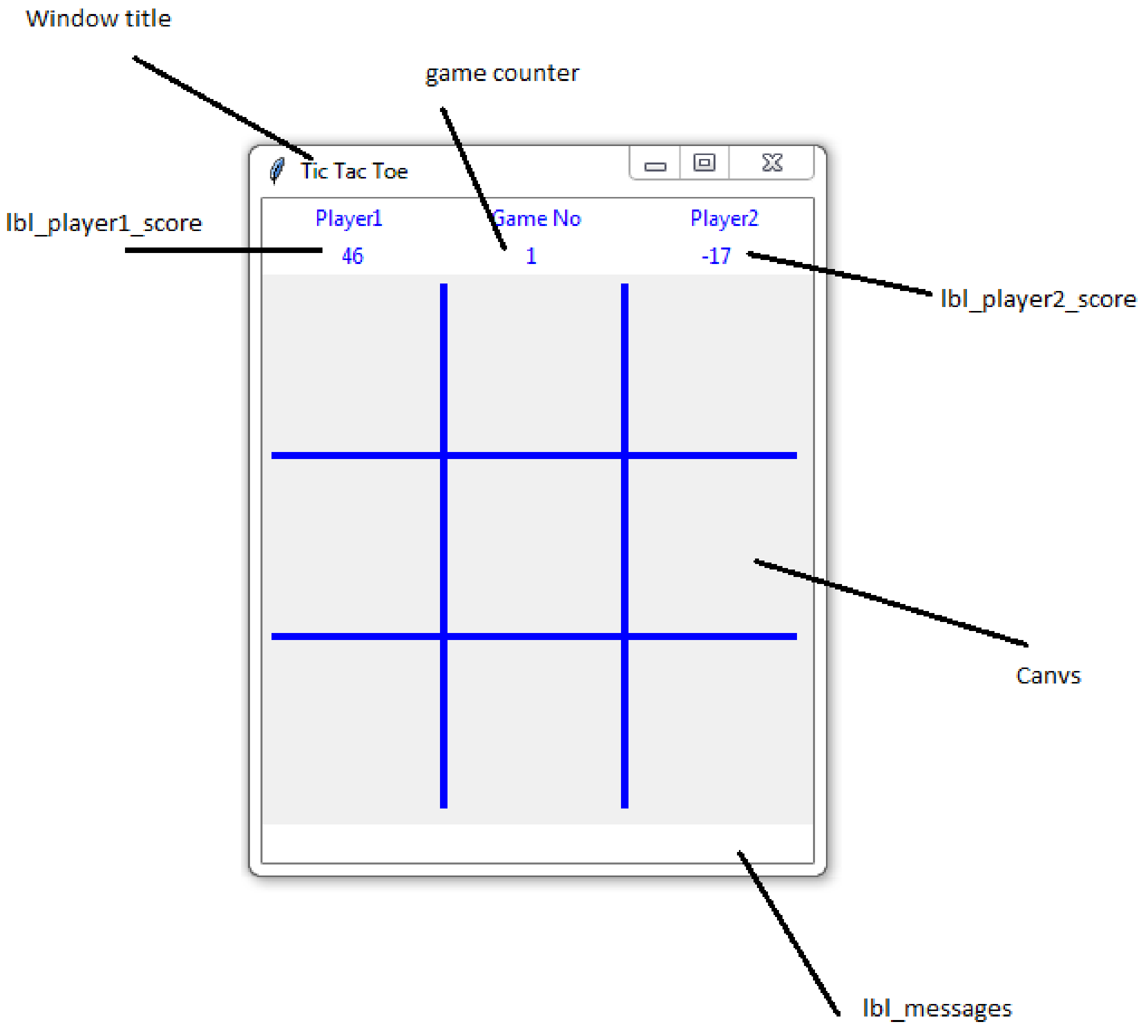
   It should also implement the following methods:

   - `__init__`: the class constructor
   - `get_player_score`: uses a select query to get the player statistics by id, if the player does not exists, then return 0's for the three values (Won, Drawn, Lost)
   - `save_player_data`: uses a select query to make sure a player exists with the passed id. if the player does not exist, then add a player record to the table Player. If the player exists, then update the three statistics values in the table. Make sure to commit in this method.
   - `__del__`: optional function used as a class destructor to close the connection.

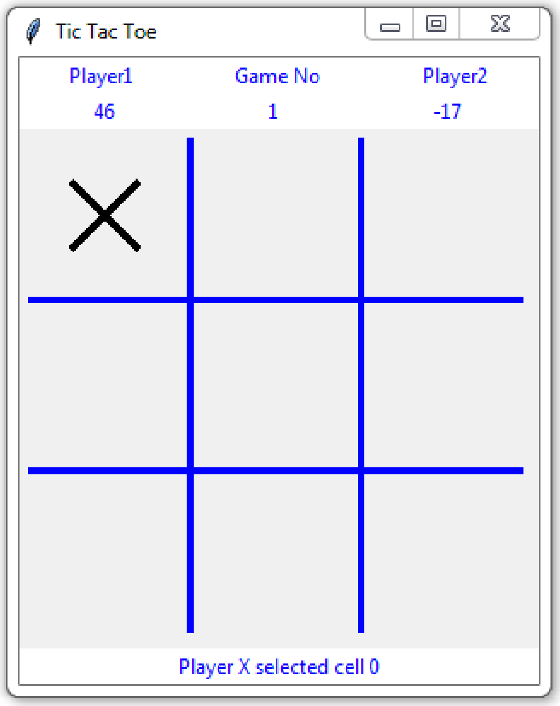
5. The class `TicTacToe` must have the following attributes:
    - `deck_list`: a list of objects of the Deck class, where the current Deck is the last item.
    - `player1`: an instance of the class Player representing player 1.
    - `player2`: an instance of the class Player representing player 2.
    - `root: ` a Tk() object that represents the main window.
    - `data_acces`: is an instance of the class DataAccessLayer to access the save and get object.
    - `status_frame:` a tkinter object that will hold the top level labels for player scores and game count.
    - `lbl_player1_score:` a tkinter label
    - `lbl_player2_score:` a tkinter label
    - `lbl_game_counter:` a tkinter label
    - `playing_canvas:` a tkinter Canvas which has four lines and will be used to render the playing marks.
    - `lbl_messages:` a tkinter label to replace any print messages from the previous code.

    Additionally it should implement the following methods:

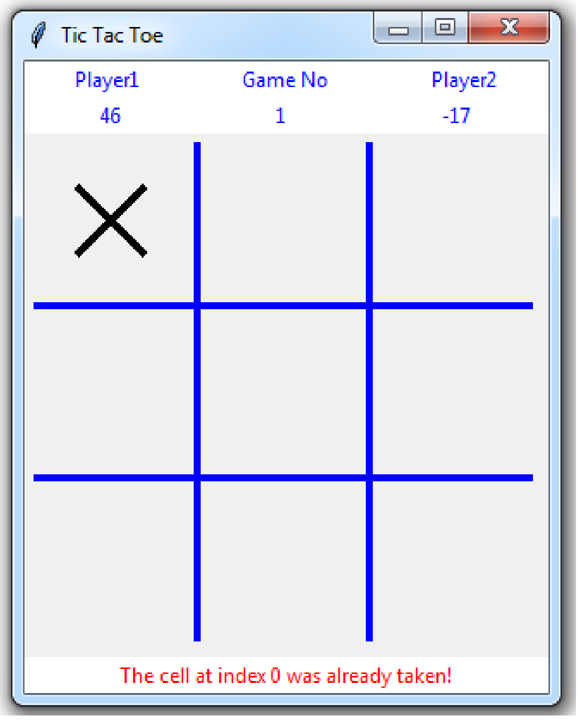
    - `__init__`: the class constructor, use it to initialize all the properties and also to call the `data_access.get_player_score()` you can assume that player1 id is 1 and player2 id is 2 and hardcode those values in the player objects.
    - `validate_user_input()`: validate if the user input is an int between 0 - 8 and it was not played previously.
    - `is_game_over()`: check if the game is over by finding if a user won or if the board is full, if true then call `data_access.save_player_data()` , return true and display an askquestion dialog, else return false.
    - `get_user_input()`: a method to get user input from a click on the canvas object then call `validate_user_input()` to validate the user input, if user's input is valid, then store the value in the Board and the check if the game is over by calling `is_game_over(),` else show an error message in the lbl_messages.
    - `start_game()`: the main game logic should go here in this class, it should add a list item to DeckList, and while the game is not over, keep calling `get_user_input()` for each user. Once a game is over, display each user data.



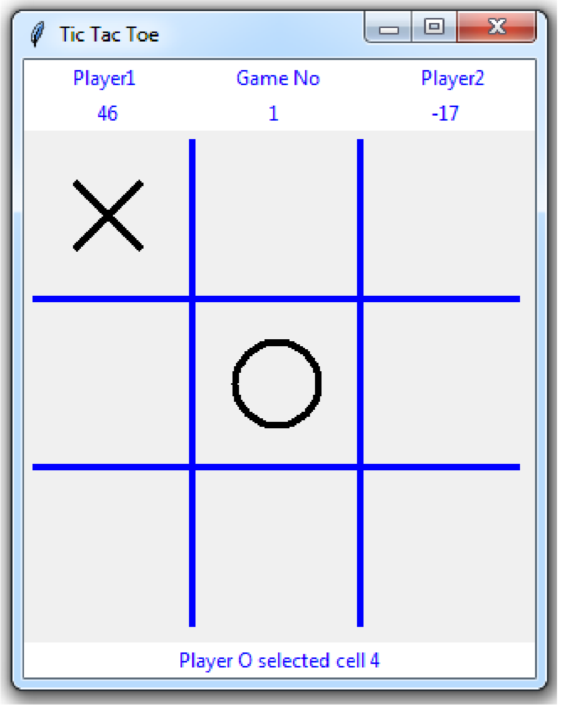
Window title
game counter
lbl_player1_score
lbl_player2_score
Canvs
lbl_messages
Tic Tac Toe
Player1
Game No
Player2
46
1
-17



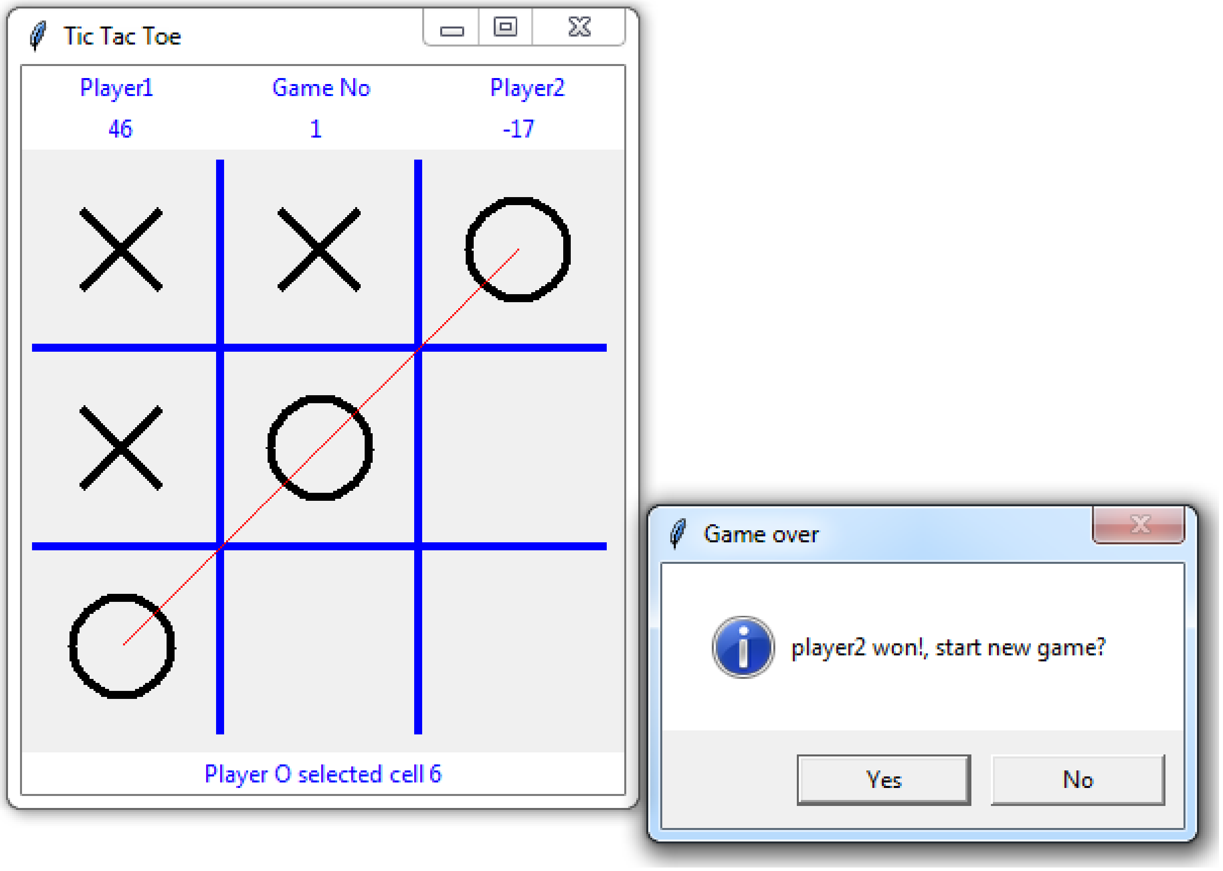
Player X selected cell 0



The cell at index 0 was already taken!



Player O selected cell 4



Player O selected cell 6

player2 won!, start new game?