Lab 6: Event Handling and Thread

Instruction

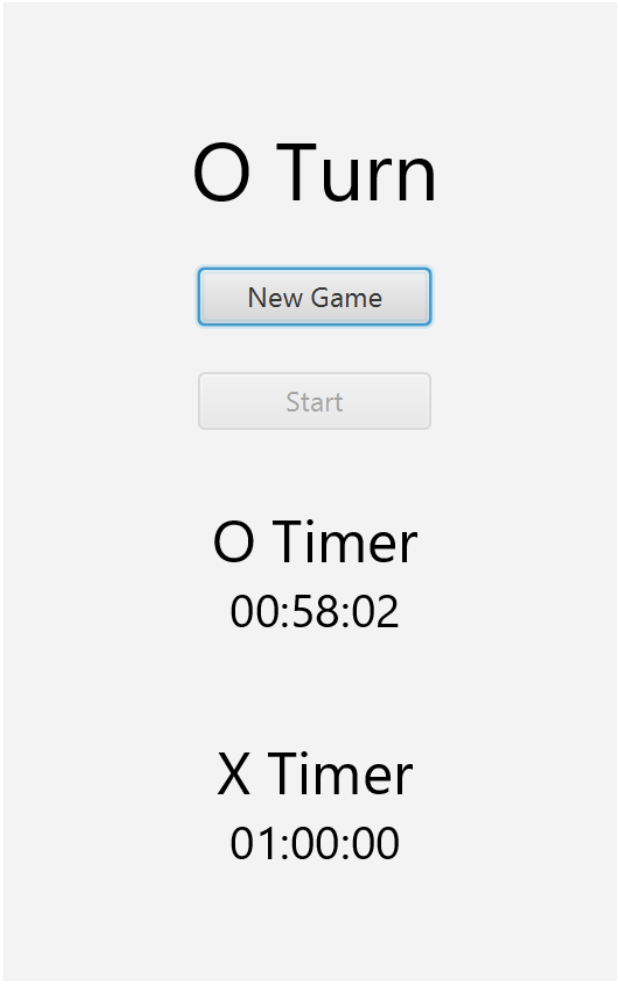
1. Click the provided link on CourseVille to create your own repository.
2. Open “Create a project” and set project name in this format **Progmeth_Lab6_2024_1_{ID}_{FIRSTNAME}**
    - Example: **Progmeth_Lab6_2024_1_6531234521_Samatcha**.
3. Initialize git in your project directory
    - **Add .gitignore.**
    - Commit and push initial codes to your GitHub repository.
4. Set VM options in your build configuration as follows: **--module-path “<path to JavaFX>” --add-modules --add-modules javafx.controls,javafx.fxml,javafx.graphics,javafx.media**
5. Implement all the classes and methods following the details given in the problem statement file which you can download from CourseVille.
    - **The provided files contain two folders:** src and res. Add both to your project and mark res as resource folder.
    - You should create commits with meaningful messages when you finish each part of your program.
    - Don’t wait until you finish all the features to create a commit.
6. Export your project into a jar file called **Lab6_2024_1_{ID}** and place it at the root directory of your project. The Jar file must be runnable and must contain the source code!
    - Example: **Lab6_2024_1_6131234521.jar**
7. Push all other commits to your GitHub repository.

1. Problem Statement: Tic Tac Toe (Demo video in Example.mp4)

1.1 Overview

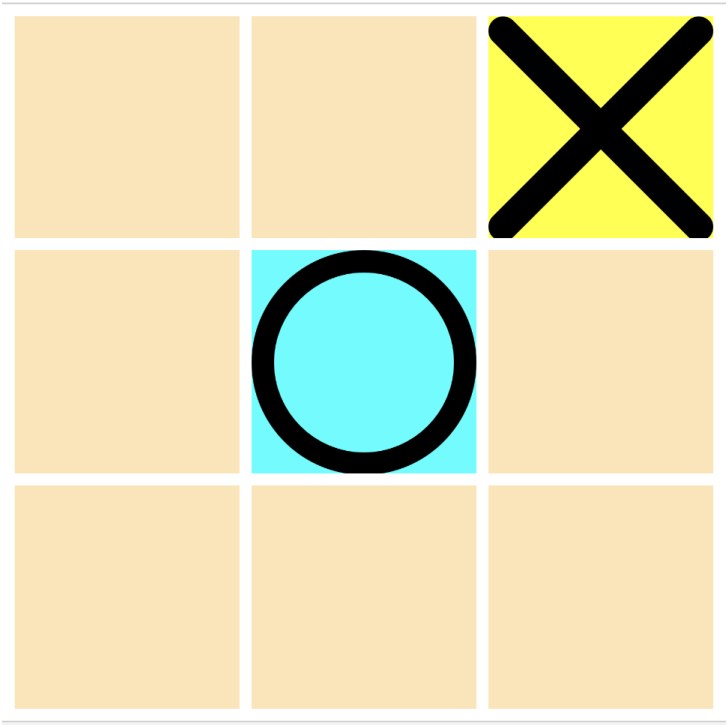
This application can be separated into 3 different sections, each section built from one or more panes.

The first section is the ControlPane.

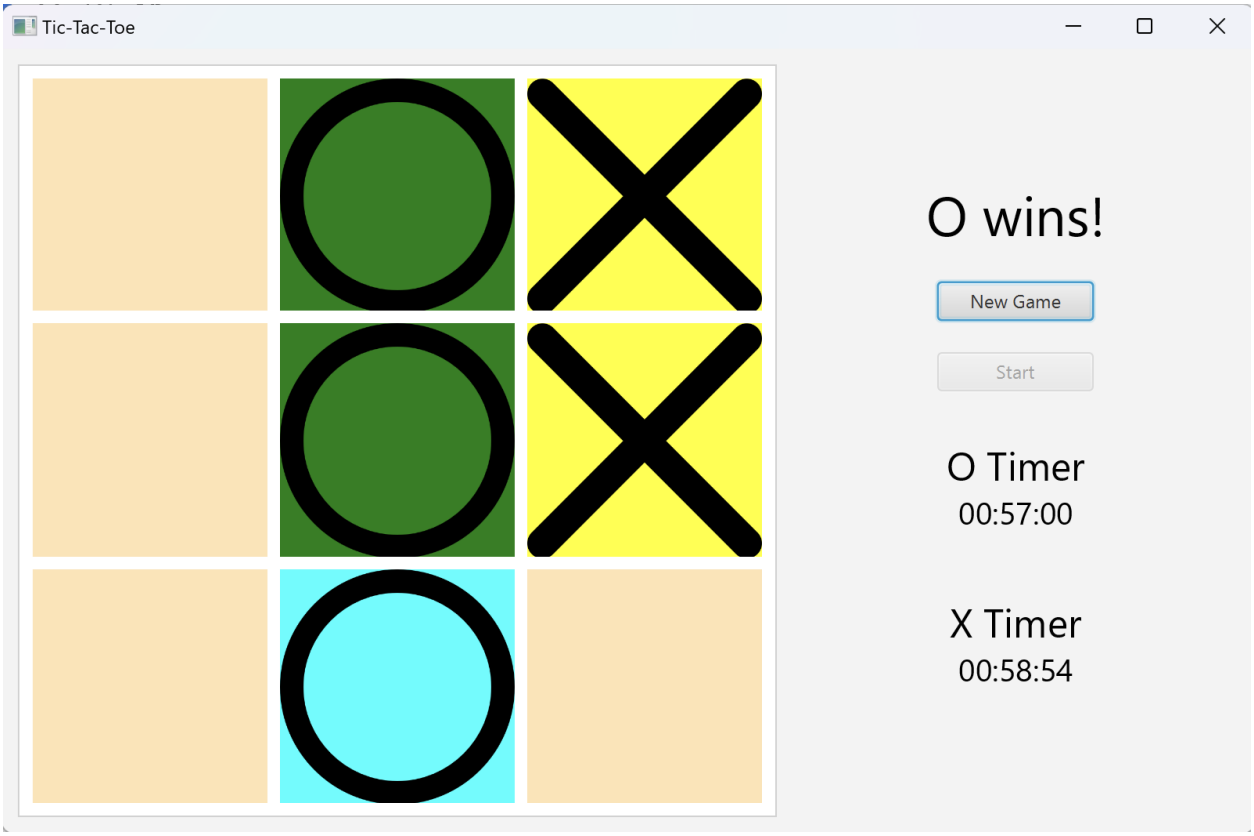


- New Game Button is used to clear the board to ready to play, Start Button is used to start the first turn which will be O player's turn, the Timer of each player is displayed below those two buttons.

The second section is the TicTacToe Pane



The pane consists of 9 TicTacToe cells which will change to O or X when clicked depending on whose turn it is.



When one of the players win the winning will flip to green color one by one.

2. Implementation Details:

Most of the Interface structures and game logics have been provided. You only need to implement the functionality of the Program. Those spots are marked with **/*FIX CODE*/**

NOTE:

1. The interface of the program **must be responsive all the time**. You can interact with the game while the timer is running, and the flipping animation can be seen without freezing.
2. There should never be any exception raised if the program is implemented correctly.
3. There is a possibility to implement the entire program without using thread (Eg. AnimationTimer). You will get the score, **but you will not get any point from thread part.**
4. Only **relevant** methods/fields/classes will be shown below. (There are a lot more methods/fields/classes, but you do not have to touch them to finish the assignment)
5. Feel free to create your own **private** methods/fields.
6:**PLEASE WATCH THE EXAMPLE VIDEO**

** Noted that Access Modifier Notations can be listed below*

+ (public), # (protected), - (private), <u>underlined</u> (static), ALL_CAPS (final)

2.1 package `logic`

2.1.1 Class `GameLogic`

This class is the main logic of the game. It contains many methods related to the game state.

2.1.2.2 Methods

+ void startCountDownTimer(int pl)	Starts the countdown timer **/*FIX CODE*/** The following code will make the timer starts counting down. But it will not work if got called by the main application thread. Make a separate Thread. Don't forget to handle Exception.
+ void runCountDownTimer(int pl) throws InterruptedException	Handling the timer counting down itself. **/*FIX CODE*/** The following code contains UI update with JavaFx, which can cause an error if running in a different thread. You need to alter the code to make this works.
+ void runWinningPattern()	Starts the winning Pattern flip **/*FIX CODE*/** The following code will make the timer cell of winning pattern start flipping but it will make the program freeze if called by the main Thread. Make a separate Thread. Don't forget to handle Exception.

+ void WinningPattern(int[] score) throws InterruptedException	Handling the flipping of winning pattern. **/*FIX CODE*/** The following code contains UI update with JavaFx, which can cause an error if running in a different thread. You need to alter the code to make this works. Between each loop, also make Thread wait for 200 ms.

3. Criteria (10 points, will be scaled down to 2.5)

Thread

- Cell Flip animation **plays successfully without crashing the program** = 2 points
- Cell Flip animation **plays one by one** = = 2 points
- Countdown Timer works = 2 points
  (no freezing of the program or crash)
- No Exception has been raised throughout the program = 2 points
  (From trying to update UI on non-JavaFX thread)
- *The JAR file runs properly.* *= 2 points*
  (You may have to modify some more code)