Exam 3, Faculty of Engineering, Chulalongkorn University
Course ID: 2110215 Course Name: Programming Methodology I
First Semester, Date: 11 December 2020 Time: 9.30-11.30AM

Name .... Student ID. .... No. in CR ....

Instructions

1. Write your Student ID, full name, and your number in CR58 in the space provided on this page.
2. Your answer must be on the computer center's machine in front of you.
3. Documents and files are allowed inside the exam room; however, internet and flash drive are prohibited. Borrowing is not allowed unless it is supervised by the proctor.
4. You must not carry mobile phone and flash drive during the exam.
5. ***** You must not bring any part of this exam paper outside. The exam paper is a government's property. Violators will be prosecuted under a criminal court and will receive an F in the subject. *****
6. Students who wish to leave the exam room before the end of the exam period, must raise their hands and ask for permission before leaving the room. Students must leave the room in the orderly manner.
7. Once the time expires, student must stop typing and must remain seated quietly until the proctors collect all the exam papers or given exam booklets. Only then, the students will be allowed to leave the room in an orderly manner.
8. Any student who does not obey the regulations listed above will receive punishment under the Faculty of Engineering Official Announcement on January 6, 2003 regarding the exam regulations.
    a. With implicit evidence or showing intention for cheating, student will receive an F in that subject and will receive an academic suspension for 1 semester.
    b. With explicit evidence for cheating, student will receive an F in that subject and will receive an academic suspension for 1 year.

Please sign and submit

Signature (..)

Important Rules

- You must not bring any part of this exam paper outside. The exam paper is a government's property. Violators will be prosecuted under a criminal court and will receive an F in the subject.
- It is a student's responsibility to check the file. If it is corrupted or cannot be open, there is no score.
- For each question, a table is given, showing (color-coded) whether or not you have to modify or create each method or variable.

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

+ (public)
# (protected)
- (private)
Underline (static)
Italic (abstract)

Set-Up Instruction

- Set workspace to "C:\temp\progmeth2020_1\Exam3_2110215_(your id)_(FirstName)" (if not exist, you must create it)
    - For example, "C:\temp\progmeth2020_1\Exam3_2110215_631234521_John"
- All your files must be in the workspace.
- The code outside of the workspace will not be collected, or graded

Scoring (Total 20 points)

- Part1 = 10 points
- Part2 = 10 points

Part 1: JavaFX GUI

1. Objective

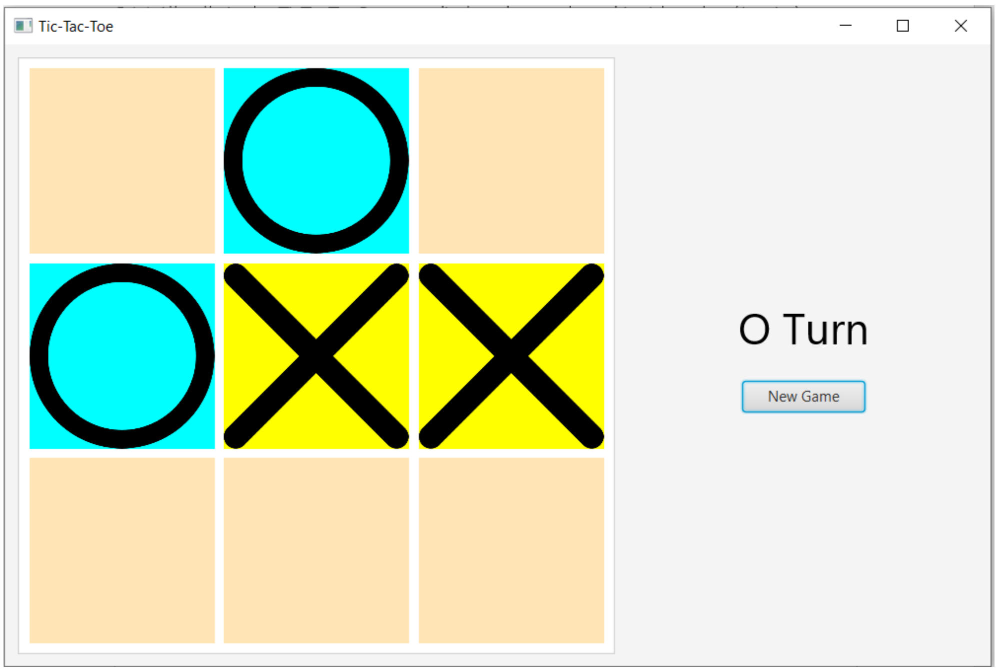
1) Students are able to implement GUI using JavaFx.

2. Instruction

1) Create Java Project named "**2110215_Exam3_Part1**".
2) Copy folders inside "**toStudent/Part1/src**" to your project directory src folder.
3) Copy "**toStudent/Part1/res**" folder to your project directory folder.
4) Example command line parameter is given in file "**vmArgument.txt**"
5) You are to implement the following classes (detail for each class is given in section 3 and 4)

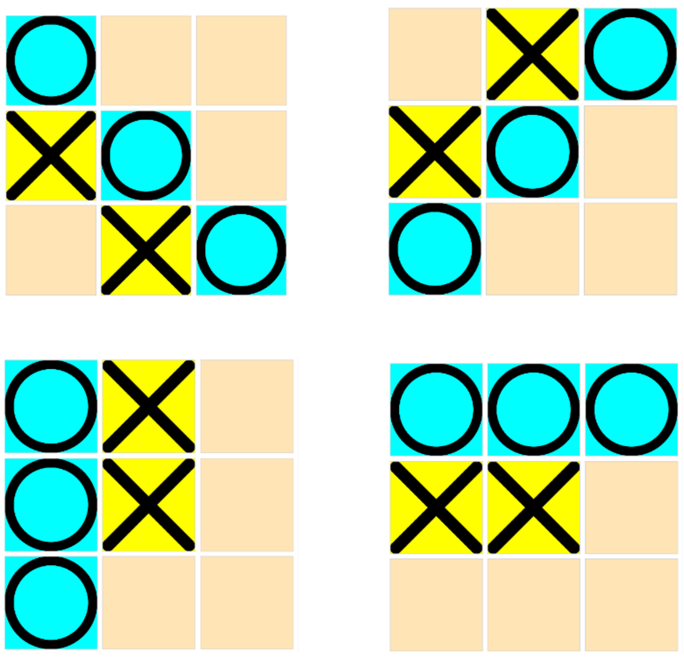
    a) **TicTacToeCell** (package gui)
    b) **ControlPane** (package gui)

3. Problem Statement: Tic-Tac-Toe

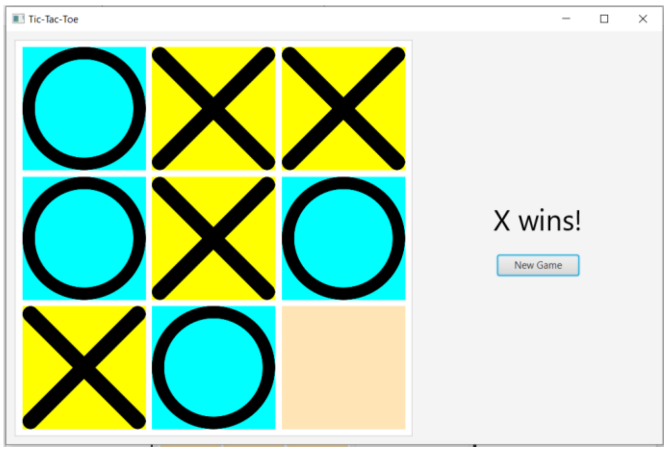


Sample screenshot of the application.

Tic-Tac-Toe is a very simple game. Two Players, X and O, have to take turns marking the spaces in a 3×3 grid.


Winning conditions in Tic-Tac-Toe.

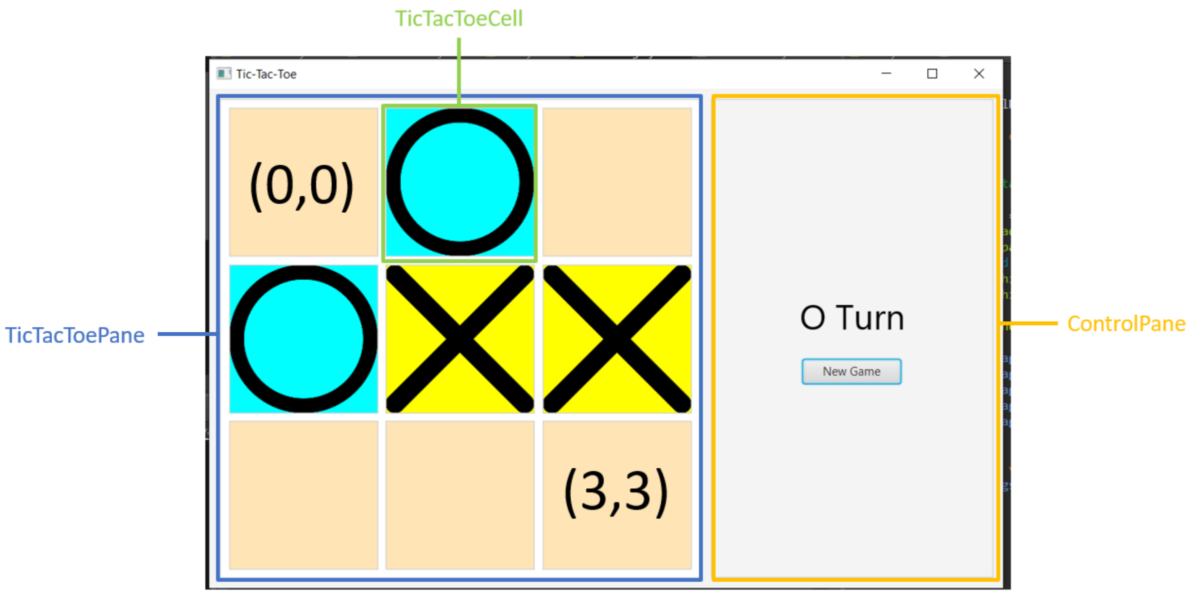
The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner.



Sample winning screenshot of the game.

You have to finish the Tic-Tac-Toe game application that contains the game pane and the control pane which contains new game button and gives the information about whose turn and who wins.

4. Implementation Detail



Detailed GUI of the Application.

The class package is summarized below.

* In the following class description, only details of IMPORTANT **fields and methods are given.** *

4.1 Package gui

4.1.1. public class **TicTacToeCell**: This class represents a Pane that can be marked by O or X.

Field

Name	Description
- boolean isDrawn	Represent that the cell has been drawn or not.
- Color baseColor	The base color of the cell when it does not have anything drawn on it.
- int xPosition	Position of the cell in X-axis. As a default of position in JavaFX GridPane, X-axis is aligned from left to right. (You can see in Detailed GUI of the Application figure)
- int yPosition	Position of the cell in Y-axis. As a default of position in JavaFX GridPane, Y-axis is aligned from top to bottom. (You can see in Detailed GUI of the Application figure)

- final String oURL	URL of O image. The value should be added in constructor as "o.png"
- final String xURL	URL of X image. The value should be added in constructor as "x.png"

Constructor

Name	Description
+ TicTacToeCell (int x, int y)	/* FILL CODE */ Constructor method. **This method is partially provided.** Initializes with the following specifications: - assign oURL as "o.png" and xURL as "x.png" - set xPostion as x and yPosition as y - set preferred width and height to 150. - set minimum width and height to 150. - set baseColor as MOCCASIN - call initializeCellColor() to initialize *cell color* - code for EventHandler is **already given.**

Method

Name	Description
- void onClickHandler()	/* FILL CODE */ This method is called when the cell is clicked. Does the following: - check if the game has ended by using GameLogic.getInstance().isGameEnd() - if the game has ended, do nothing. - if not, check if the cell has been drawn. If it has been drawn, do nothing. - if not, check whose turn it is by using GameLogic.getInstance().isOturn() and use draw(Image image, Color backgroundColor) method to draw the cell ● If it is O turn use oURL to get the image of O and **AQUA** color as backgroundColor ● If it is X turn use xURL to get the image of X and **YELLOW**

	color as backgroundColor ● after drawing, memorize game state by using GameLogic.getInstance().drawNumber(xPosition, yPosition);
- void draw(Image image, Color backgroundColor)	- set the Background with backgroundColor with image
+ void initializeCellColor()	/* FILL CODE */ Set the **Background** to be **filled** with baseColor and set *isDrawn* to false. This method is used for initializing and resetting the cell.
+ getter/setter for each field.	All necessary getters/setters are already provided.

4.1.2. public class **TicTacToePane**: This class represents the grid with TicTacToeCell.

Field

Name	Description
- ArrayList<TicTacToeCell> allCells	List that contains TicTacToeCell objects in the grid.

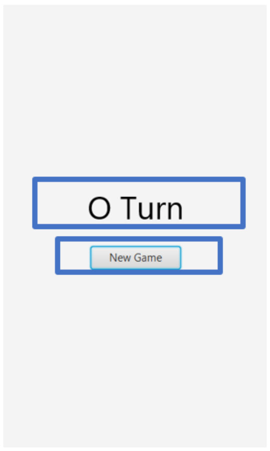
Constructor

Name	Description
+ TicTacToePane()	Constructor method. Initializes with the following specifications: - initializing allCells - set horizontal gap and vertical gap as 8 - set the inset padding of 8 and preferred width as 500 - set alignment as **CENTER** - set border to **LIGHTGRAY** color, stroke style **SOLID**, corner radii **EMPTY**, with **DEFAULT** width. - set background as **WHITE** color - initialize TicTacToeCell and add them to allCells and to this pane

Method

Name	Description
+ ArrayList<NumberSquare> getAllCells ()	Getter method for *allCells*.

4.1.3. public class **ControlPane**: This class is the pane that contains game information text and restart game button. Items in the pane is arranged vertically.



Field

Name	Description
- Text gameText	The Text for displaying whose turn it is, or who wins.
- Button newGameButton	The button for beginning a new round.
- TicTacToePane ticTacToePane	A TicTacToePane that is updated by this ControlPane.

Constructor

Name	Description
+ ControlPane (TicTacToePane ticTacToePane)	/* FILL CODE */ Constructor method. Sets ticTacToePane field to match the parameter. Then, initializes with the following specifications: - set the alignment to **CENTER**. - set preferred width to 300. - set spacing to 20. - call initializeGameText() to initialize *gameText.*

	- call initializeNewGameButton() to initialize *newGameButton*. - add gameText and newGameButton field to this pane's children in correct order.

Method

Name	Description
- void initializeGameText()	/* FILL CODE */ - Initializes *gameText* with text "O Turn" - set *gameText* font with size 35
- void updateGameText(String text)	/* FILL CODE */ - set *gameText* with text *text*
- void initializeNewGameButton()	/* FILL CODE */ - initialize *newGameButton* with text "New Game". - set the button preferred width to 100. - set onAction to handle with newGameButtonHandler() method. (See below)
- void newGameButtonHandler ()	/* FILL CODE */ This method is the handler method for *newGameButton*. Does the following: - resetting game state using GameLogic.getInstance().newGame() method - set *gameText* text to "O Turn" - resetting all cells in ticTacToePane by using initializeCellColor()

4.2 Package logic

4.2.4. public class **GameLogic**: **This class is already provided.** Only useful fields and methods are shown here.

Field

Name	Description
- <u>GameLogic instance</u>	Instance that represents GameLogic class. This implementation confirms that we have only one GameLogic.
- boolean isGameEnd	A field that shows if the game has ended or not.
- boolean isOTurn	A field that shows if the current turn belongs to O.
- ControlPane controlPane	ControlPane that will update when game state is changed.

Method

Name	Description
+ void drawNumber (int x, int y)	This method should be called when TicTacToeCell has been drawn. GameLogic will memorize the state of the game and check whether the game ends or not. Then, update the ControlPane information text. If the game does not end, change the turn and update the ControlPane information text.
+void newGame()	Reset the game state to its initial state.
+ static GameLogic getInstance()	Getter of instance. Use this method whenever you want to use the method in GameLogic.

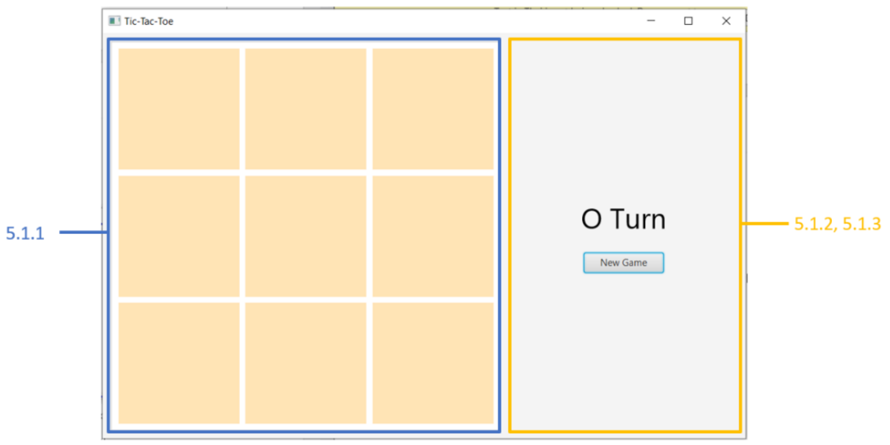
4.2 Package main

4.2.1. public class **Main**: This class contains main method. It is an entry point of the application. **This class is already provided.**

5. Scoring Criteria

The maximum score for the problem is 10.

5.1 Initializing program

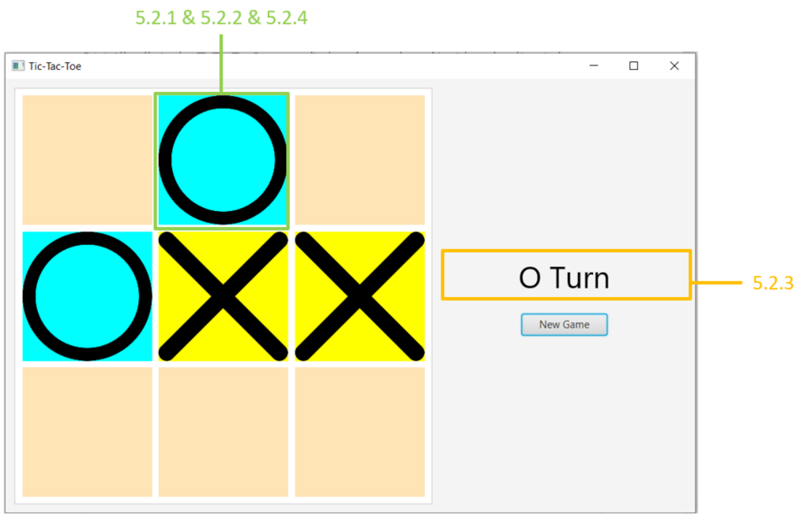


5.1.1 All cells in the TicTacToePane are displayed properly and in right color. (1 point)

5.1.2 Text in Control pane is displayed properly. (0.5 point)

5.1.3 Button in Control pane is displayed properly. (0.5 point)

5.2 Drawing cells



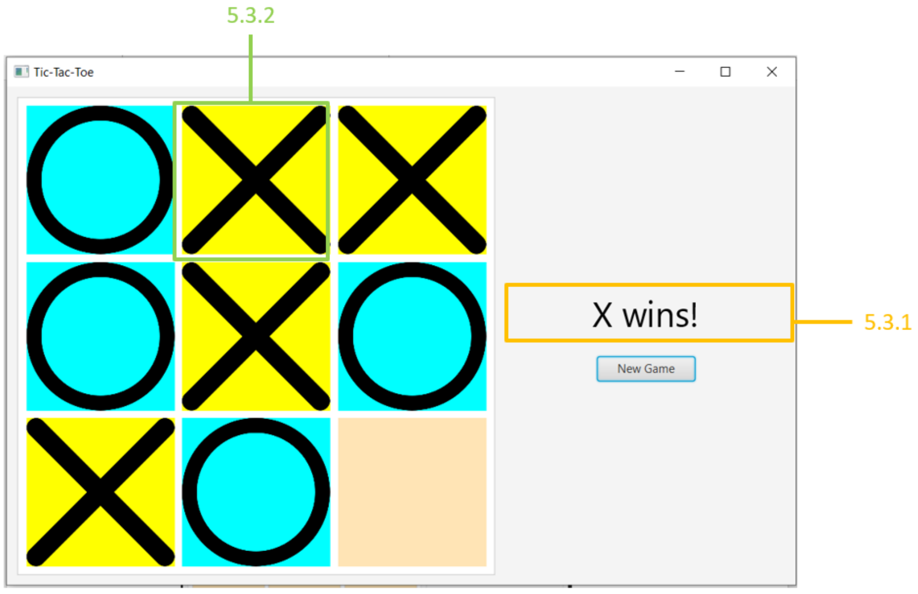
5.2.1 During each player's turn, clicking on an empty cell will draw that player's mark. (1 point)

5.2.2 Drawn cells displayed in proper images and color. (1 point)

5.2.3 Game state does not change when a cell is clicked twice. (1 point)

5.2.4 Text in control pane is displayed properly. (1 point)

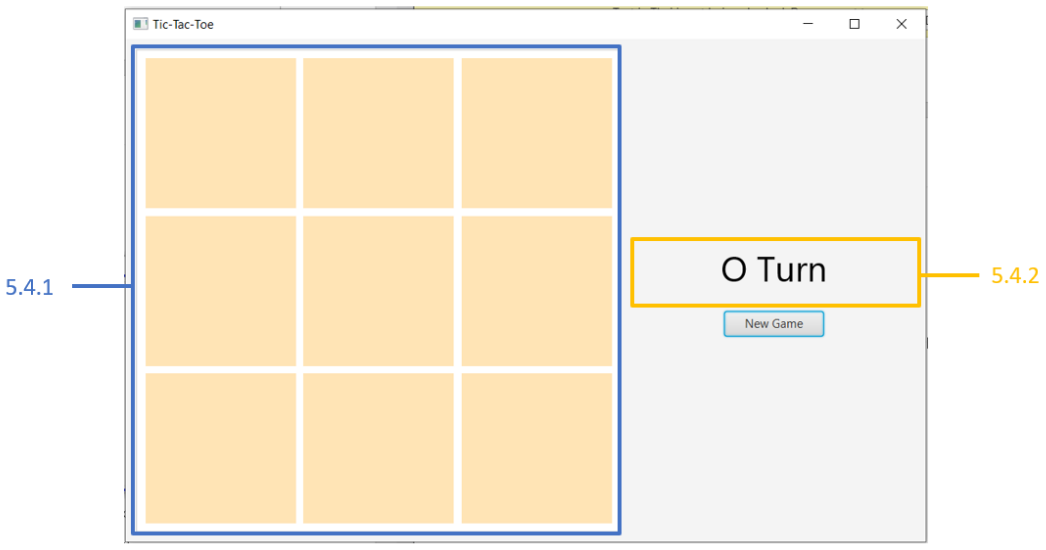
5.3 Game ending



5.3.1 When game reaches the end, text displays the right winner. (1 point)

5.3.2 When game reaches the end, all cells cannot be clicked. (1 point)

5.4 New game



5.4.1 When new game button is clicked, all cells in TicTacToePane reset to its initial state. (1 point)

5.4.2 When new game button is clicked, the text in control pane resets to its initial state. (1 point)

Part 2: Java Thread

1. Objective

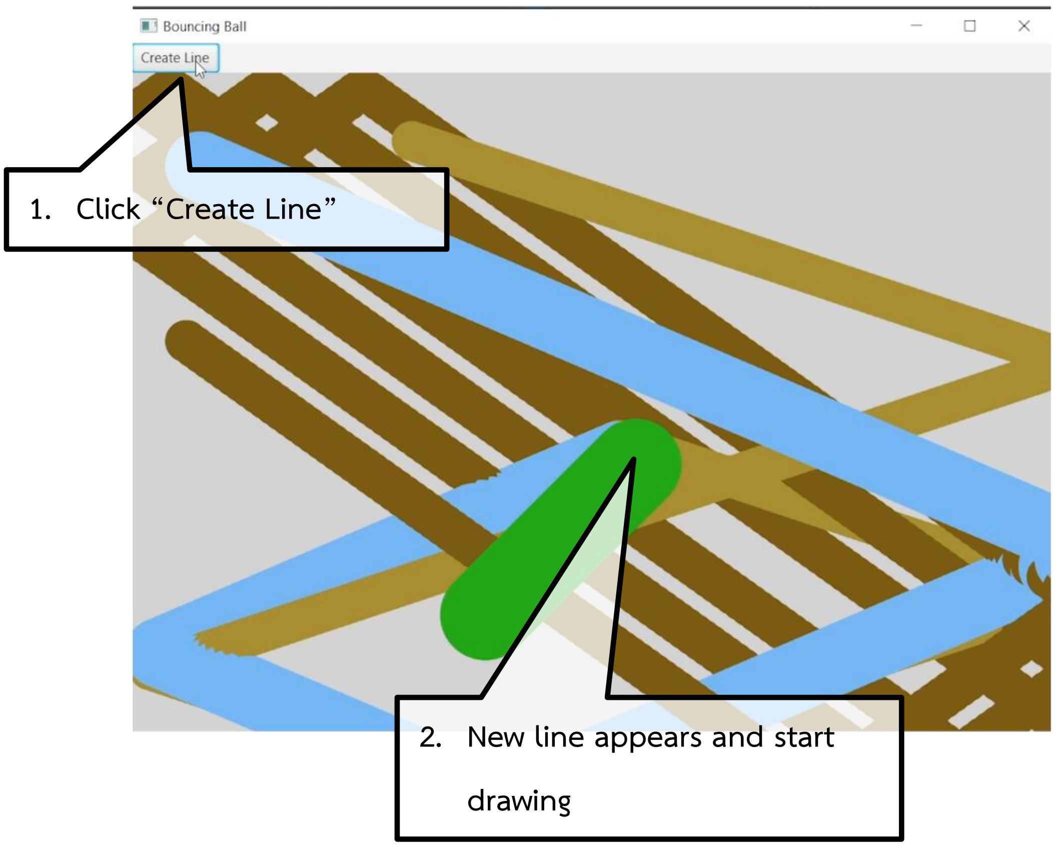
1) Be able to implement Java thread to make a program responsive.

2. Instruction

1) Create Java Project named "**2110215_Exam3_Part2**".
2) Copy folder "application" and "data" (in "toStudent/Part2") into your project src folder.
3) Fix the code (detail shown below) inside.
4) Example command line parameter is given in file "vmArgument.txt"

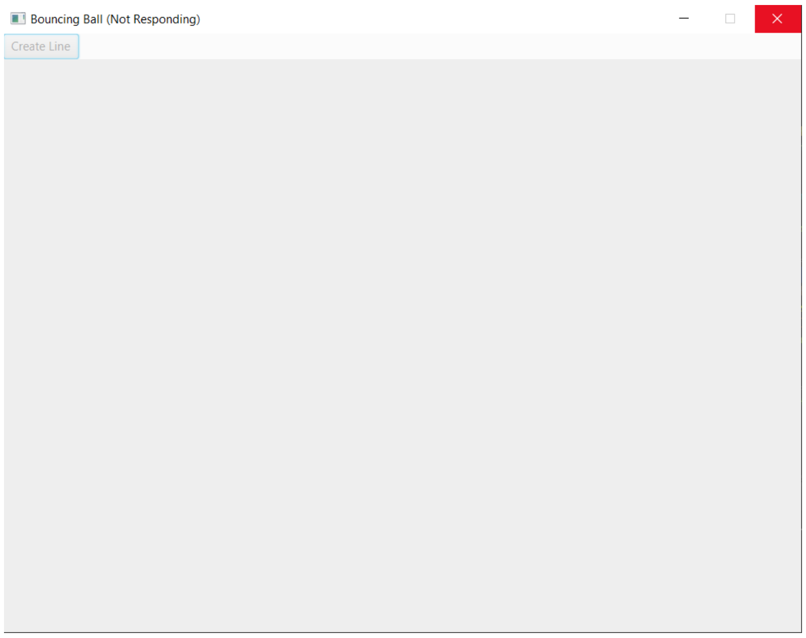
3. Problem Statement: Bouncing Line.

main.java is a Javafx application that show bouncing lines animation. Click on "Create Line" button will create a single line (technically, it consists of multiple circles created and form into a line) in the canvas below and it will start drawing and bouncing-off canvas wall. (Size, color, and speed of each line will be random)



For more information, a video file "Exam 3-2 Example.mp4" is given to show you how the finished application should look like.

If you try out main.java and click on "Create Line" button. You will see that application will not respond at all. This is because application tried to draw the line indefinitely. Without thread implemented, other functions like button and canvas don't get a chance to run.



A screenshot of the frozen application.

Your task is to use threads to allow bouncing lines to be shown on the canvas. **For a line, it always has the same speed no matter how many other lines are on the canvas.**

The application might be slower with high number of lines, we will only look at 1-20 lines cases.

4. Implementation Detail

The code is given in **main.java**.

A variable **GraphicsContext ctx** is used to draw the line on canvas.

You do not need to know any other details of the user interface aside from that.

Hint : GraphicsContext ctx is related to JavaFX.

- You can add your own methods and variables.
- **You MUST NOT change the line "Thread.sleep(10);" If you do, <u>you get 0 point</u>.**
- If no threads are used to solve this question, **<u>you get 0 point</u>.**

Line.java and **LineField.java** classes are given, **You MUST NOT change anything in These 2 classes**. If you do, **<u>you get 0 point</u>**.

5. Scoring Criteria

The maximum score for the problem is 10.

5.1 Click "Create Line" button once - create a line and show it moving on the canvas while the application is still responding **(5 points).**

5.2. After creating more lines, the lines don't slow down and not causing any exception (from 2 to 20 lines) **(5 points).**