LAPLAP

By GG_Nein

Members

Kridtin Chawalratikool ID : 5931003221

Kantanat Siripipatworakun ID : 5931005521

1 Introduction

LapLap is a music rhythm game developed as a term project of 2110215 Programming Methodology class. It is written entirely with Java language with Javafx library. It uses various crucial aspects of Java language such as Object Oriented Programming (OOP), thread, image and audio integration, and many more. For example, inheritance, polymorphism, access modifier setter/getter are used in this project. This project has begun on 20th November 2017, and with dedication and tremendous endeavour, it was finally completed on 11th December 2017. We hope that it's fun to play for all adults, children and the instructors as well.

2 User Manual

LapLap is a single player game that player select a mode, song, and then play by pressing any of 3 buttons to match the circle that appear on top and slide downward on the screen. This will happen continuously meanwhile music that player chose is still playing. When players type the button correctly, they get a score.

The game has five scenes, a main menu scene, a select mode scene, a select song scene, an in-game scene, and a game result scene.

Figure 1: the main menu scene of the Game

At the main menu scene, player can:

- Press **ENTER** to go to the select mode scene, or
- Press **ESC** to exit the program.

When player presses **ENTER**, the program's window switches to the select mode scene as shown in figure 2.

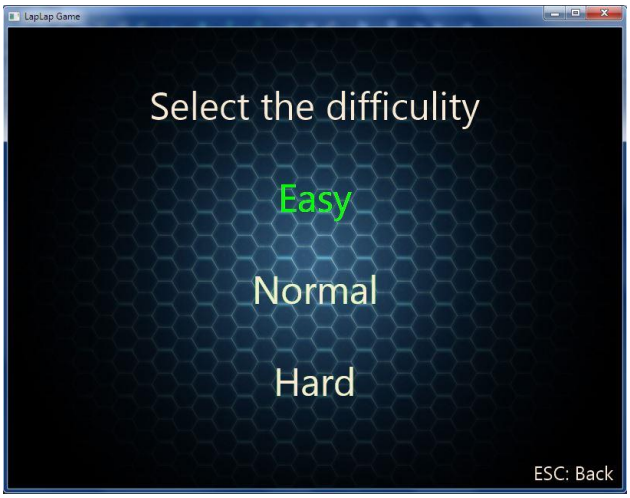

Figure 2: the Select Mode Scene of the Game

At the select mode scene, player can:

- Choose difficulty mode with the green highlight text by pressing **UP** or **DOWN**
- And then press **ENTER** to go to the select song scene, or
- Press **ESC** to go back to the main menu scene

When player have chosen difficulty mode and press **ENTER**, the program's window will switches to the select song scene as shown in figure 3.

Figure 3: the Select Song Scene of the Game

At the select song scene, player can:

- Choose song with the green highlight text by pressing **UP** or **DOWN**
- And then press **ENTER** to play the game, or
- Press **ESC** to go back to the select mode scene

When player have chosen difficulty mode and press **ENTER**, the program's window will switches to the game scene.

In the game scene, there are 2 screens, the game loading screen as show in figure 4 and the in-game scene as shown in figure 5.

Figure 4: the game loading screen

In the game loading screen:

- Will start counting down 'Game start in 3' and number 3->2->1
- After that, it will launch the in-game scene

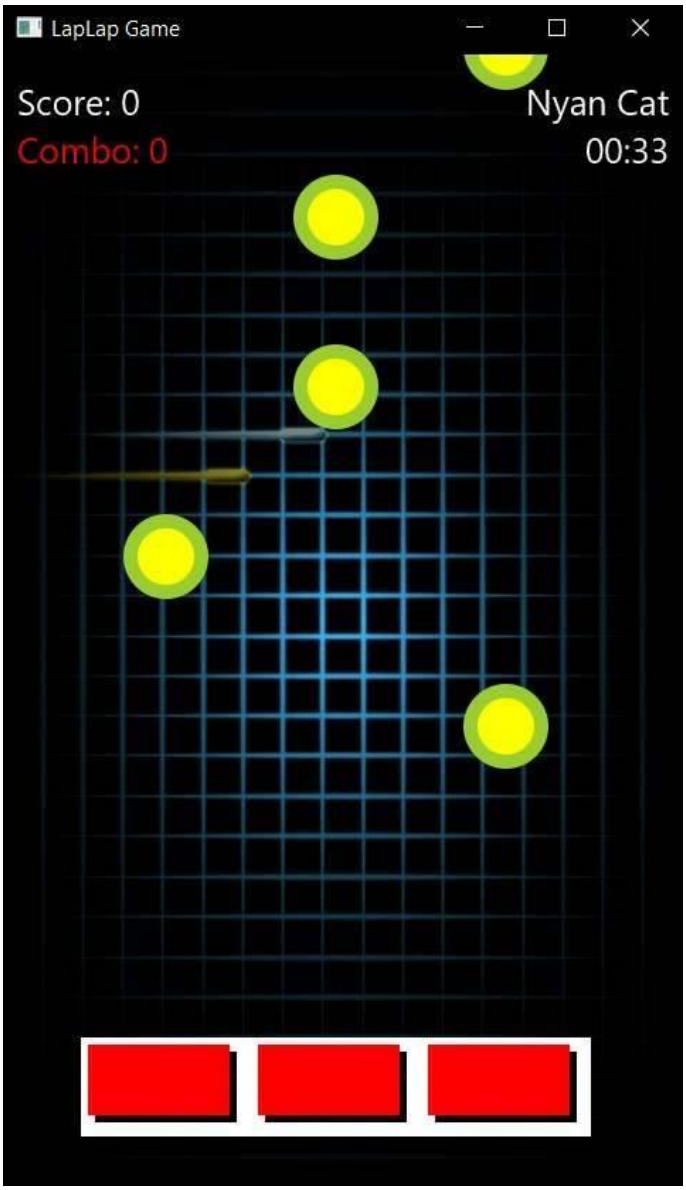

Figure 5: the in-game scene

In the in-game scene, it is displaying:

1. Score text at the top-left corner of the screen
2. Combo text at the top-left corner and under the score text
3. A song name chosen by player at the top-right of the screen
4. A time of the song that player choose at the top-right and under the song name
5. A white rectangle that there are 3 red squares inside, at the bottom-center of the screen
6. Many of yellow-in-green circles that moving downwards from top of the screen

   And the Song chosen by player will be played

- When the circles move inside the red rectangle, Player can press these three buttons for matching the circle:
    - **A** for the left rectangle
    - **S** for the middle rectangle
    - **D** for the right rectangle
- When player press these buttons, red color will turn to orange color
- If player can match the circle, player will get the score and combo will increase
- If player match circles without any miss, the score and combo will increase much more
- If player press button but not match the circle, player will not get score and combo will reset to 0

When player play game until the song is finished, the program's window will switches to a game result scene as shown in figure 6.

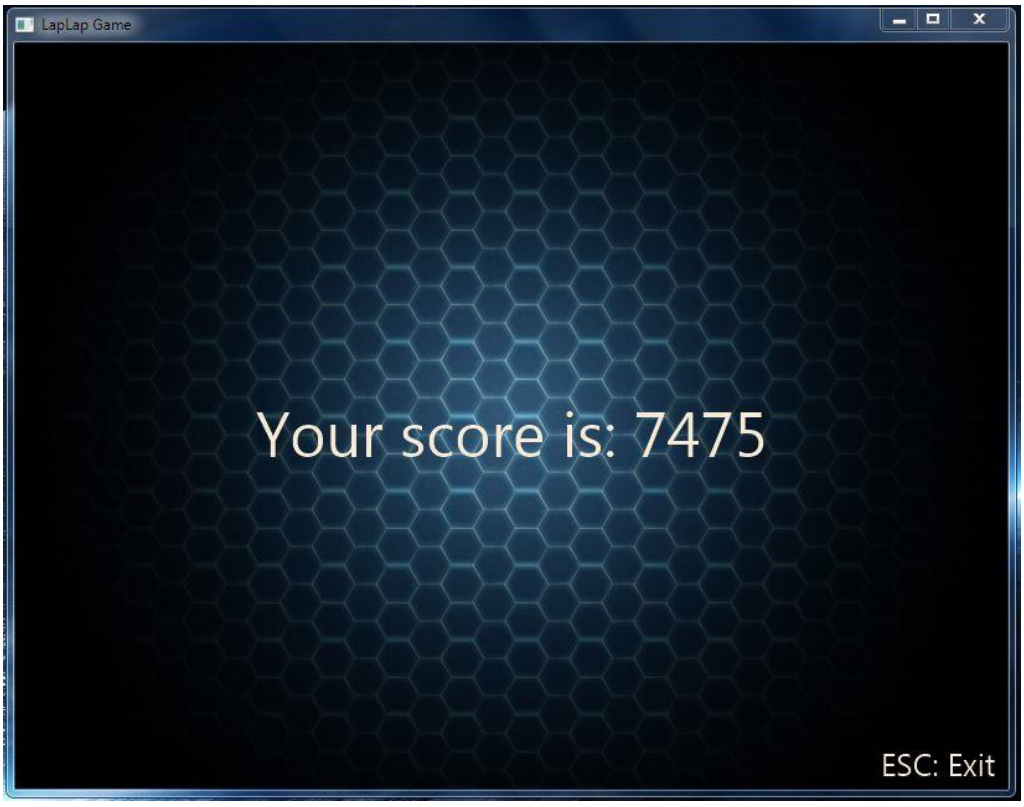

Figure 6: the Game Result Scene of the Game

At the game result scene, it will show:

- The Score of player after playing the game
- Player can press **ESC** button to exit the game

3. Implementation Detail

The diagram of the program is shown in Figure 6 below. There are 7 packages, in the total of 21 classes and 2 interfaces. The first package is application package which have 2 classes, Main and GameMain. Next package is graphic package, contains all canvases in this game and Canvas manager. It containsCanvasManager, DisplayModeSelect, DisplaySongSelect, GameResult, InGame, MainMenu, Pause and the interface, Drawable. Next package is input, contains InGameInput. Next one is logic package, contains Button, CountDownTimer, GameLogic, GameModel, Items, Note class. Next is sharedObject, which contains RenderableHolder class and Renderable interface. The song package contains Pair, Song, SongResources class. Last one is window package, contains SceneManager class. The UML diagram is shown below.

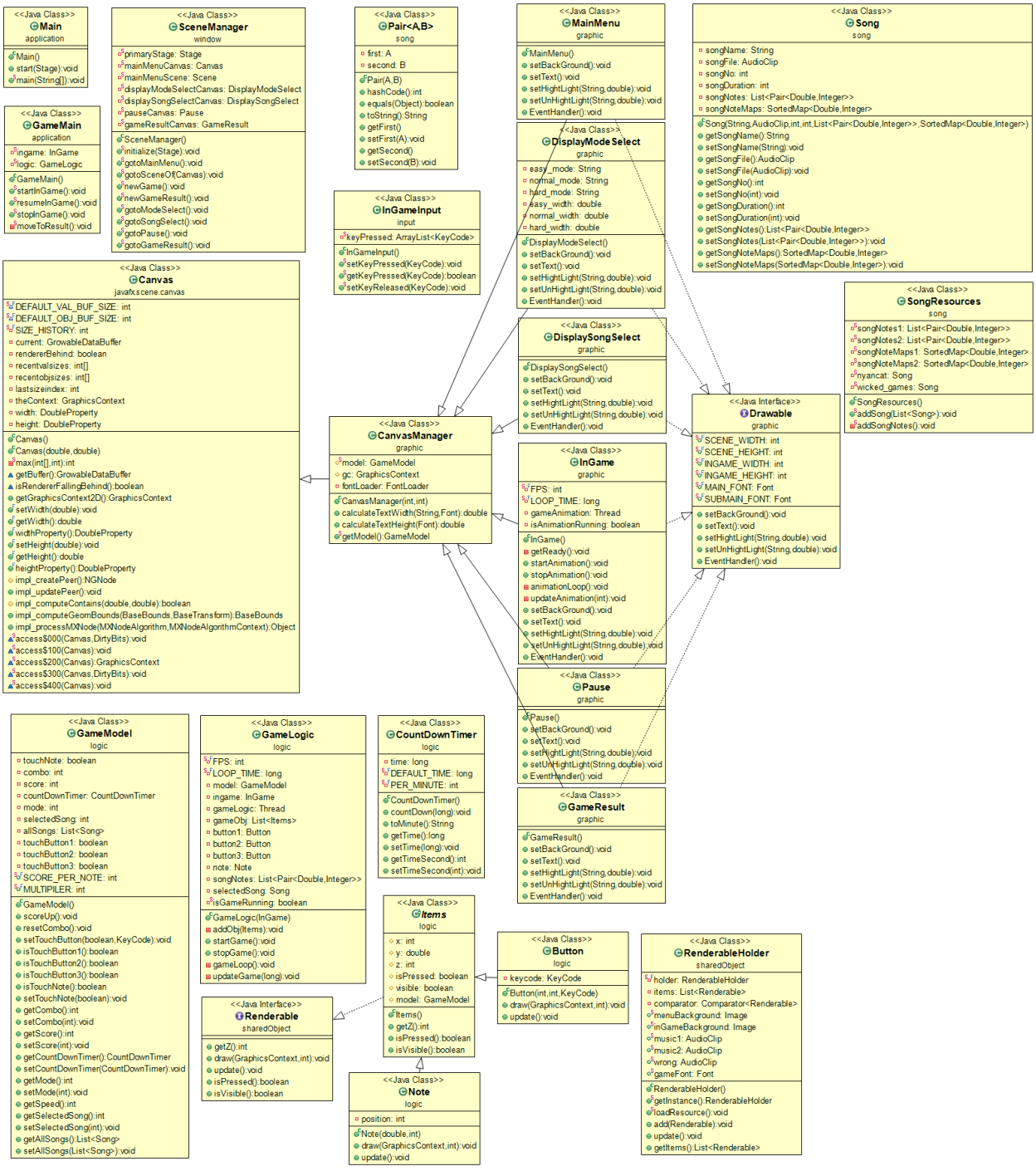

Figure 7: The UML diagram of the program

3.1 Package application

3.1.1 Class Main extends Application

3.1.1.1 Method

+ void start(Stage primaryStage)	The main entry for the JavaFX applications. There are the code setOnCloseRequest to completely close the application.
+ void main(String[] args)	An entry of the application.

3.1.2 Class GameMain

3.1.2.1 Field

- InGameingame	For call the canvas and startAnimation
- GameLogic logic	For call startGame

3.1.2.2 Method

+ void startInGame()	- Initialize the fields - Go to the scene of ingame - Start the logic and animation threads
+ void stopInGame()	- Stop the logic and animation threads - Call moveToResult method
+ void moveToResult()	- Initialize the result canvas - Move to that canvas

3.2 Package graphic

3.2.1 Class CanvasManager extends Canvas

3.2.1.1 Field

# GameModel model	To collect the data from the player and use it in the whole game
# GraphicContextgc	To draw of each canvas
# FontLoaderfontLoader	To use calculate width and height of the text

3.2.1.2 Constructor

+ CanvasManager(intsceneWidth, intsceneHeight)	- Initialize canvas by setting the width and height - Initialize fontLoader with Toolkit.getToolkit().getFontLoader(); - Initialize model when the model didn't initial before.

3.2.1.3 Method

+ double calculatedTextWidth(String text, Font font)	Calculate the string width
+ double calculateTextHeight(Font font)	Calculate the string height
+ GameModelgetModel()	Return model

3.2.2 Class MainMenu extends CanvasManager implements Drawable

3.2.2.1 Constructor

+MainMenu()	- Set the width and height with SCENE_WIDTH and SCENE_HEIGHT - Initialize gc - call the setBackground(), setText(), EventHandler() methods

3.2.2.2 Method

+ voidsetBackground()	Set the background of canvas with menuBackgroundin RenderableHolder
+ void setText()	Set the text of canvas - Title - Description - Esc: Exit Game
+ void setHighLight(String selected_mode, double selected_width)	Blank method (not use, but need to implements)
+ void setUnHightLight(String unsel_mode, double unsel_width)	Blank method (not use, but need to implements)
+ void EventHandler()	Add listener to listen when the player press - Escape: Exit Game - Other keys: Move to next canvas

3.2.3 Class DisplayModeSelect extends CanvasManager implements Drawable

3.2.3.1 Field

- String easy_mode	String “Easy”
- String normal_mode	String “Normal”
- String hard_mode	String “Hard”
- double easy_width	Width position to draw easy_mode
- double normal_width	Width position to draw normal_mode
- double hard_width	Width position to draw hard_mode

3.2.3.2 Constructor

+ DisplayModeSelect()	- Set the width and height with SCENE_WIDTH and SCENE_HEIGHT - Initialize gc - call the setBackground(), setText(), EventHandler() methods

3.2.3.3 Method

+ voidsetBackground()	Set the background of canvas with menuBackgroundin RenderableHolder
+ void setText()	Set the text of canvas - Title - Modes - Esc: Back
+ void setHighLight(String selected_mode, double selected_width)	Set the text highlight when the player chose it
+ void setUnHightLight(String unsel_mode,	Set the text unhighlight when the player left

double unsel_width)	it
+ void EventHandler()	Add listener to listen when the player press - Escape: back to previous canvas - Enter: choose the selected mode and move to next canvas - Up/Down: Change the highlight position

3.2.4 Class DisplaySongSelect extends CanvasManager implements Drawable

3.2.4.1 Constructor

+ DisplaySongSelect()	- Set the width and height with SCENE_WIDTH and SCENE_HEIGHT - Initialize gc - call the setBackground(), setText(), EventHandler() methods

3.2.4.2 Method

+ voidsetBackground()	Set the background of canvas with menuBackgroundin RenderableHolder
+ void setText()	Set the text of canvas - Title - Songs - Esc: Back
+ void setHighLight(String selected_mode, double selected_width)	Set the text highlight when the player chose it
+ void setUnHightLight(String unsel_mode, double unsel_width)	Set the text unhighlight when the player left it

+ void EventHandler()	Add listener to listen when the player press - Escape: back to previous canvas - Enter: choose the selected song and move to next canvas - Up/Down: Change the highlight position

3.2.5 Class InGame extends CanvasManager implements Drawable

3.2.5.1 Field

- int FPS	60 Frames per second
- long LOOP_TIME	Time between each update of a game animation
- Thread gameAnimation	Thread of animation
- Boolean isAnimationRunning	Variable that control the thread loop to start or not

3.2.5.2 Constructor

+ InGame()	- Set the width and height with INGAME_WIDTH and INGAME_HEIGHT - Initialize gc - call the EventHandler() methods

3.2.5.3 Method

- void getReady()	Show the screen that count down before the game start
+ void startAnimation()	- Initialize gameAnimation

	- Set isAnimationRunning = true - Start the animation thread
+ void stopAnimation()	Set isAnimationRunning = false
- void animationLoop()	Loop the animation - Call getReady() - Start the song - Loop while isAnimationRunning is true, call the updateAnimation
- void updateAnimation(int count)	- Set background - Render the items from RenderableHolder - Set text
+ void setBackground()	Set the background of canvas with inGameBackgroundin RenderableHolder
+ void setText()	- Add score and combo text at the top left - Add song name and time at the top right
+ void setHighLight(String selected_mode, double selected_width)	Blank method (not use, but need to implements)
+ void setUnHightLight(String unsel_mode, double unsel_width)	Blank method (not use, but need to implements)
+ void EventHandler()	Add listener to listen when the player press - A, S, D: always return only 1 tick if the player hold the button or not

3.2.7 Class GameResult extends CanvasManager implements Drawable

3.2.7.1 Constructor

+ GameResult()	- Set the width and height with SCENE_WIDTH and SCENE_HEIGHT - Initialize gc - call the setBackground(), setText(), EventHandler() methods

3.2.7.2 Method

+ voidsetBackground()	Set the background of canvas with menuBackgroundin RenderableHolder
+ void setText()	Set the text of canvas - the player's score - Esc: Exit - if the player reach all combos, show OK alert with the text
+ void setHighLight(String selected_mode, double selected_width)	Blank method (not use, but need to implements)
+ void setUnHightLight(String unsel_mode, double unsel_width)	Blank method (not use, but need to implements)
+ void EventHandler()	Add listener to listen when the player press - Enter/Escape: Exit the game

3.2.8 Interface Drawable

3.2.8.1Field

+ int SCENE_WIDTH	800px
+ int SCENE_HEIGHT	600px
+ intINGAME_WIDTH	480px
+ intINGAME_HEIGHT	800px
+ Font MAIN_FONT	Monospace with size 50
+ Font SUBMAIN_FONT	Monospace with size 25

3.2.8.2 Method

+ void setBackGround()	
+ void setText()	
+ void setHightLight(String selected_mode, double selected_width)	
+ void setUnHightLight(String unsel_mode, double unsel_width)	
+ void EventHandler()	

3.3 Package input

3.3.1 Class InGameInput

3.3.1.1 Field

- ArrayList<KeyCode>keyPressed	Initialize ArrayList

3.3.1.2 Method

- void setKeyPressed(KeyCodekeyCode)	- If the keyPressed don't have that keyCode, insert the keycode into the list
- booleangetKeyPressed(KeyCodekeyCode)	- return true if there is already have the keyCode in the list, otherwise return false
- void setKeyReleased(KeyCodekeyCode)	- remove the keycode when release

3.4 Package logic

3.4.1 Class Button extends Items

3.4.1.1 Field

- KeyCode keycode	The keycode from the player

3.4.1.2 Constructor

+ Button(int x, int y, KeyCode keycode)	- Set the parameter x, y, keycode to the field - Set z value to 50 (under the Note)

3.4.1.3 Method

+ void draw(GraphicsContextgc, int count)	Draw the button - if isPressed true, draw white stroke and red square - if false, draw white stroke and orange square
+ void update()	- ifgetKeyPressed() from InGameInput is true, set isPressed true, setTouchButton in model true with the keycode field, otherwise set all above to false - add local variable selectedSong and time_now, call from model - iftime_now is in a map of songnotescheck if the button the player pressed is A,S, or D match with the note position, set the touchnote in model true and the score is up. Otherwise, set touchnote false.

3.4.2 Class CountDownTimer

3.4.2.1 Field

- long time	Time now
- long DEFAULT_TIME	65 x 10^9 L
- int PER_MINUTE	60 seconds

3.4.2.2 Constructor

+ CountDownTimer	Set the initial time to default time

3.4.2.3 Method

+ void countdown(long elapsedTime)	Minus time with elapsedTime
+ String toMinute()	Return String of time in format XX:XX
+ long getTime()	Return time
+ void setTime(long time)	Set the time with given parameter
+ intgetTimeSecond()	Return time in second
+ void setTimeSecond()	Set the time in long with the input in second

3.4.3 Class GameLogic

3.4.3.1 Field

- int FPS	60 Frames per second
- long LOOP_TIME	Time between each update of a game animation
- GameModel model	A model to call the function
- InGameingame	An ingame field to call the function
- Thread gameLogic	A thread needs to run together with gameAnimation thread in InGame
- List<Items>gameObj	List that contains all buttons and notes
- Button button1	Left button
- Button button2	Center button
- Button button3	Right button
- Note note	Field of note
- List<Pair<Double, Integer>> songNotes	List that collect songNotes
- Song selectedSong	The song that the player selected
- Boolean isGameRunning	Variable that control the thread loop

3.4.3.2 Constructor

+ GameLogic(InGame ingame)	- set model from CanvasManger - set ingame from parameter given - initialize 3 buttons with coordinate (60, 700), (180, 700), and (300, 700). Each of button parameter have a KeyCode A, S, D - set selectedSong from model - set songNotes from selectedSong - initialize note from songNotes - call addObj(Items item) for all notes and buttons

3.4.3.3 Method

- void addObj(Items item)	Add item in gameObj and RenderableHolder
- void startGame()	- set isGameRunning true - initialize gameLogic with the thread that run gameLoop method - start the thread
- void stopGame()	Set isGameRunning to false
- void gameLoop()	- Run thread.sleep for the screen that countdown before game start - Loop while isGameRunning is true, call the updateGame
- void updateGame(long elapsedTime)	- update the buttons and notes (Note: notes shoud be visible, otherwise don’t update) - call countdown method from CountDownTimer with elapsedTime

	- if the game's time reach 0, stop the game

3.4.4 Class GameModel

3.4.4.1 Field

- Boolean touchNote	The note is touch or not
- int combo	Combo when the player touch on the notes continuously
- int score	Score of the player
- CountDownTimer countDownTimer	Field to access CountDownTimer class
- int mode	0 is easy 1 is normal 2 is hard
- int selectedSong	The number of the song that the player selected
- List<Song> allSongs	List of all songs
- Boolean touchButton1	State of the left button
- Boolean touchButton2	State of the center button
- Boolean touchButton3	State of the right button
+ int SCORE_PER_NOTE	Score of each note, 100 pts
+ int MULTIPILER	Multipiler when have combos

3.4.4.2 Constructor

+ GameModel()	- set all touchButton to false - set touchNote false - set combo and score to 0 - initialize CountDownTimer

	- set mode and selectedSong to 0 - initialize allSongs - Add the songs to allSongs by calling addSong(List<Song> allSongs) in SongResources

3.4.4.3 Method

+ void scoreUp()	If touchNote is true, score will increase with (mode + 1)*(SCORE_PER_NOTE + combo*MULTIPILER)
+ void resetCombo	Set to combo back to 0
+ void setTouchButton(Boolean touchButton, KeyCode keycode)	Setter of all the touchButtons state If keycode = A: set touchButton1 If keycode = S: set touchButton2 If keycode = D: set touchButton3
+ boolean isTouchButton1()	Getter of touchButton1
+ boolean isTouchButton2()	Getter of touchButton2
+ boolean isTouchButton3()	Getter of touchButton3
+ Boolean isTouchNote()	Getter of touchNote
+ void setTouchNote(Boolean touchNote)	Setter of touchNote
+ int getCombo()	Getter of combo
+ void setCombo(int combo)	Setter of combo
+ int getScore()	Getter of score
+ void setScore(int score)	Setter of score
+ CountDownTimer getCountDownTimer()	Getter of countDownTimer
+ void setCountDownTimer(CountDownTimer timer)	Setter of countDownTimer
+ int getMode()	Getter of mode

+ void setMode(int mode)	Setter of mode
+ int getSpeed()	Return a speed to use in Note class. The speed is 5 + 15*(mode)
+ int getSelectedSong()	Getter of selectedSong
+ void setSelectedSong(int song)	Setter of selectedSong
+ List<Song> getAllSongs()	Getter of allSongs
+ void setAllSongs(List<Song> allSongs)	Setter of allSongs

3.4.5 Class Items implements Renderable

3.4.5.1 Field

# int x	Value in the x-axis
# double y	Value in the y-axis
# int z	Value in z-axis
# boolean isPressed	State of the items, press or not
# boolean visible	State of the items, show or not
# GameModel model	Model field to connect with GameModel

3.4.5.2 Constructor

+ Items()	- set isPressed to false - set visible to true - set model from CanvasManager

3.4.5.3 Method

+ int getZ()	Return z
+ Boolean isPressed()	Return isPressed

+ Boolean isVisible()	Return visible

3.4.6 Class Note extends Items

3.4.6.1 Field

- int position	Position of Note, left, center, right

3.4.6.2 Constructor

+ Note(double y, int position)	Set position, y (the time of note that need to touch) with the parameter given and set z value to 100 (over the Button)

3.4.6.3 Method

+ void draw(GraphicContext gc, int count)	Draw the note with circle, by the position field, and y-axis set the value depend on the speed/the time of note that need to touch of the game
+ void update()	- create local variable selectedSong for calling the song that the player selected - create local variable time_now to get time that already passed in the song (by songDuration minus with countdowntimer) - if the player touch the button 1/2/3 and position value is match with the button, and the time_now is in the map of songNoteMaps, remove the key, value of time_now and set

	isPressed true, visible false

3.5 Package sharedObject

3.5.1 Class RenderableHolder

3.5.1.1 Field

- RenderableHolder holder	Initialize holder itself
- List<Renderable> items	List of Renderable items
- Comparator<Renderable> comparator	To compare the z value and sort the list
+ Image menuBackground	The background of almost canvas
+ Image inGameBackground	The background of InGame canvas
+ AudioClip music1	Music of Nyancat
+ AudioClip music2	Music of Wicked Games
+ AudioClip wrong	The wrong sound
+ Font gameFont	An external font

3.5.1.2 Constructor

+ RenderableHolder()	- initialize list - initialize comparator to compare the z value

3.5.1.3 Method

+ RenderableHolder getInstance()	Return holder field
+ void loadResource()	Load the resources before the game launch - menuBackground, inGameBackground, music1, music2, wrong, gameFont

+ void add(Renderable note)	Add item to items list and sort
+ void update()	Update the list when the player already pressed the notes
+ List<Renderable> getItems()	Return items list

3.5.2 Interface Renderable

3.5.2.1 Method

+ int getZ()	
+ void draw(GraphicContext gc, int count)	
+ void update()	
+ boolean isPressed()	
+ boolean isVisible()	

3.6 Package song

3.6.1 Class Pair (From StackOverFlow)

3.6.1.1 Field

- A first	First value of pair
- B second	Second value of pair

3.6.1.2 Constructor

+ Pair(A first, B second)	Set the first and second with parameters given

3.6.1.3 Method

+ int hashCode()	
+ Boolean equals()	Check two pairs are equal or not
+ String toString()	Return the pair into string
+ A getFirst()	Getter of first
+ void setFirst(A first)	Setter of first
+ B getSecond()	Getter of second
+ void setSecond(B second)	Setter of second

3.6.2 Class Song

3.6.2.1 Field

- String songName	Name of song
- AudioClip songFile	Song file
- int songNo	Number of song
- int songDuration	Duration of the song
- List<Pair<Double, Integer>> songNotes	List of notes of the song
- SortedMap<Double, Integer> songNoteMaps	Map of notes of the song

3.6.2.2 Constructor

+ Song(String songName, AudioClip songFile, int songNo, int songDuration, List<Pair<Double, Integer>> songNotes, SortedMap<Double, Integer> songNoteMaps)	Initialize all of fields with given parameters

3.6.2.3 Method

+ String getSongName()	Getter of songName
+ void setSongName(String songName	Setter of songName
+ AudioClip getSongFile()	Getter of songFile
+ void setSongFile(AudioClip songFile)	Setter of songFile
+ int getSongNo()	Getter of songNo
+ void setSongNo(int songNo)	Setter of songNo
+ int getSongDuration()	Getter of songDuration
+ void setSongDuration(int songDuration)	Setter of songDuration
+ List<Pair<Double, Integer>> getSongNotes()	Getter of songNotes
+ void setSongNotes(List<Pair<Double, Integer>> songNotes)	Setter of songNotes
+ SortedMap<Double, Integer> getSongNoteMaps()	Getter of songNoteMaps
+ void setSongNoteMaps(SortedMap<Double, Integer> songNoteMaps)	Setter of songNoteMaps

3.6.3 Class SongResources

3.6.3.1 Field

- List<Pair<Double, Integer>> songNotes1	Initialize songNotes1
- List<Pair<Double, Integer>> songNotes2	Initialize songNotes2
- SortedMap<Double, Integer> songNoteMaps1	Initialize songNoteMaps1
- SortedMap<Double, Integer> songNoteMaps2	Initialize songNoteMaps2
- Song nyancat	Initialize nyancat song with - “Nyan Cat” - RenderableHolder.music1 - 0

	- 35 - songNotes1 - songNotesMaps1
- Song wicked_games	Initialize wicked_games song with - "Wicked Games" - RenderableHolder.music2 - 1 - 4*60 + 41 - songNotes2 - songNotesMaps2

3.6.3.2 Method

+ void addSong(List<Song> songs)	Call the method addSongNotes() and add song to the songs
- void addSongNotes()	Add all notes into the 2 songNotes and songNoteMaps

3.7 Package window

3.7.1 Class SceneManager

3.7.1.1 Field

- Stage primaryStage	A main stage of the game
- Canvas mainMenuCanvas	Initialize a mainmenu canvas
- Scene mainMenuScene	Initialize a mainmenu scene with mainMenuCanvas
- DisplayModeSelect displayModeSelectCanvas	Canvas of DisplayModeSelect
- DisplaySongSelect displaySongSelectCanvas	Canvas of DisplaySongSelect

- GameResult gameResultCanvas	Canvas of GameResult

3.7.1.2 Method

+ void initialize(Stage stage)	- Set the primaryStage from the given parameter - show the primaryStage
+ void gotoMainMenu()	- set scene of primaryStage to mainMenuCanvas - Call the requestFocus method to mainMenuCanvas
+ void gotoSceneOf(Canvas canvas)	- initialize scene from canvas - set scene to primaryStage - Call the requestFocus method to the canvas
+ void newGame()	Initialize displayModeSelectCanvas and displaySongSelectCanvas if they do not exist
+ void newGameResult()	Initialize gameResultCanvas if it does not exist
+ void gotoModeSelect()	Call method gotoSceneOf(displayModeSelectCanvas)
+ void gotoSongSelect()	Call method gotoSceneOf(displaySongSelectCanvas)
+ void gotoGameResult()	Call method gotoSceneOf(gameResultCanvas)