INTERNATIONAL UNIVERSITY

VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

School of Computer Science and Engineering

-----*-----**

PROJECT REPORT

FES CHALLENGE

OBJECT-ORIENTED PROGRAMMING (IT069IU)

Semester 1 - Academic year 2023-2024

Course by: **Dr. Tran Thanh Tung**

MSc. Nguyen Quang Phu

Table of Contents

CONTRIBUTION TABLE

No.	Full Name	Student's ID	Task	Contribution
1	Tạ Thị Thùy Dương	ITCSIU21053	GUI, Plan Manager, Interface Design, Debugger	100%
2	Lê Thiên Ngân	ITCSIU21091	Sound, PowerPoint Slide Manager, Debugger	95%
3	Lê Hoàng Vĩ	ITITIU21343	GUI(Background, Guide), Guide, Design	92%
4	Hứa Hoàng Quyên Quyên	ITCSIU21225	GUI(Gems, Background), Time, Design	95%
5	Bùi Văn Minh Triều	ITCSIU21241	UML Diagram, Report Manager, GUI(Setting)	95%

ABSTRACT

“Fes Challenge” was based on Shariki which was developed by Russian developer Eugene Alemzhin, and has emerged as one of the most popular and addictive puzzle video games since its release in 1994 .

The game's success can be attributed to its simple yet engaging match-three puzzle gameplay, where players swap colorful circles to create combinations and clear levels. The progression system, featuring a diverse range of levels with increasing difficulty, keeps players invested and motivated. In addition, the incorporation of boosters, special candies, and diverse game modes adds layers of strategy and excitement, making the gaming experience dynamic and enjoyable.

Furthermore, Fes Challenge employs various psychological techniques to enhance player retention and satisfaction. The use of vibrant colors, cheerful graphics, and satisfying sound effects create a visually and auditorily pleasing experience. The introduction of challenges, in-game events keeps players motivated and invested over the long term. The immediate objective of the project is to give a brief introduction to Fes Challenge, to redesign the game based on fundamentals of design patterns and to apply game rules.

Encapsulation, inheritance, and polymorphism are a few of the concepts related to object-oriented programming (OOP) that are covered in this project. The use of OOP has grown in the software real world due to the future expansion of the software business and the advancement of software engineering. Thus, the primary goal of this project is to become familiar with their enormous benefits in coding management, learn how to apply them to some related Fes Challenge features, and notice the proper outcome.

Chapter 1: INTRODUCTION

A. Objectives

- This project aims to assist members revising and applying knowledge to form a Game completely.
- From this game, students understand and manipulate three fundamental concepts of Object-Oriented Programming (OOP).
- Developing more new features to make the Game more Interesting for Players.

B. The tools used

- Integrated Development Environments: JetBrains IntelliJ IDEA and VS Code
- Code version management: GitHub
- Project management: GitHub - FesChallenge.
- Storage material: Google Drive (**Figure 4**)
- Communication & Weekly Meetings: Zalo (**Figure 1**), Messenger (**Figure 2**), Microsoft Teams (**Figure 3**)

Figure 1: Zalo platform

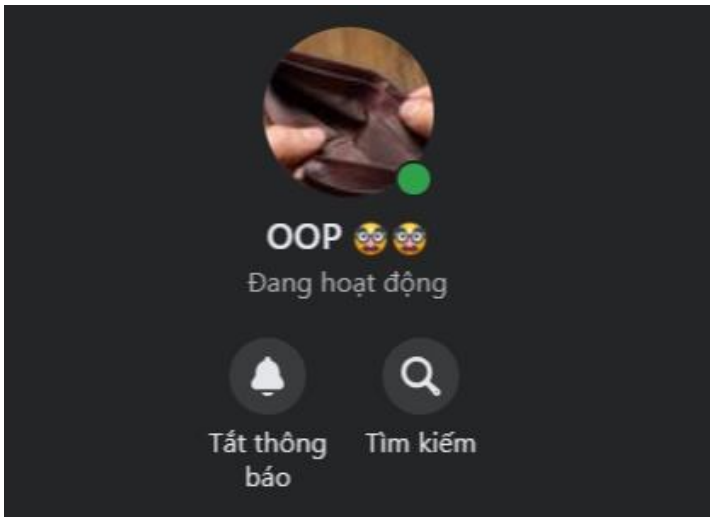

Figure 2: Messenger platform

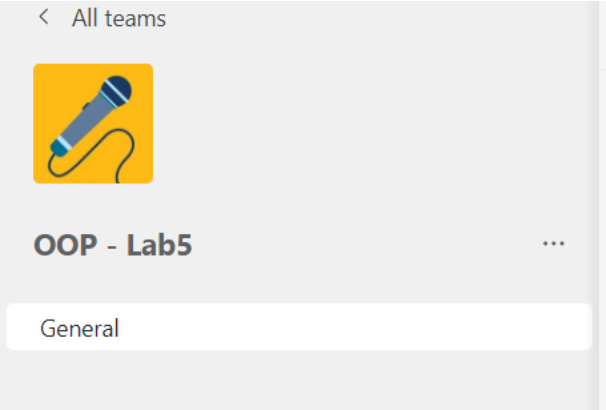

Figure 3: Microsoft Teams platform

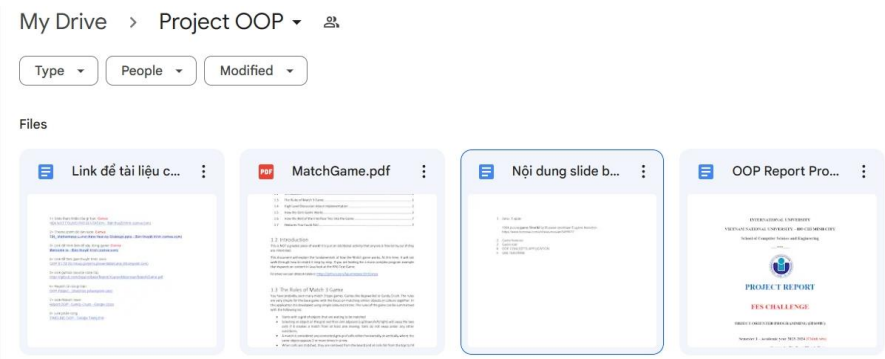

Figure 4: Google Drive

Chapter 2: GAME DESCRIPTION

A. Design

- Game is created with an interface and seven other game elements entirely on our own. As seen in **Figure 5** below, the pieces are fixed designs that are connected to Tet:

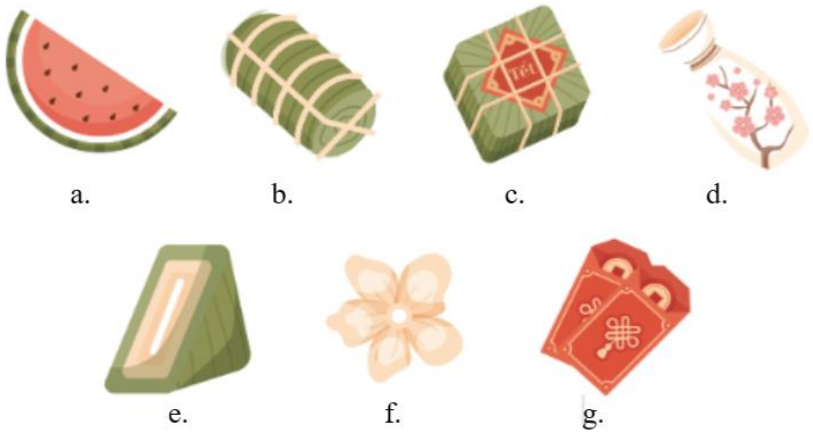

Figure 5: The designed items

a. Watermelon item
b. Tet cake item
c. Chung cake item
d. Vase item
e. Piece of Chung cake item
f. Flower item
g. Red envelope item

- The game has 5 primary interfaces, as seen in Figures (6, 7, 8, 9, and 13) below:

Figure 6: The Start Interface

Figure 7: The Rule Interface

Figure 8: The Setting Interface

Figure 9: The Game Interface

- The Fes Challenge can be interacted by clicking the 'Festival Items' on the Match Board located on the left side of the The Game Interface (Figure 9). The Status Panel (Figure 10) which is located on the right side includes Data Frame (Figure 11) and two buttons (Figure 12) . Players can view the time, goal, the increased point total, and the score via Data Frame at the top of the Status Panel. Restart and Quit are the two buttons at the bottom of the Status Panel that allow you to restart or end the game.

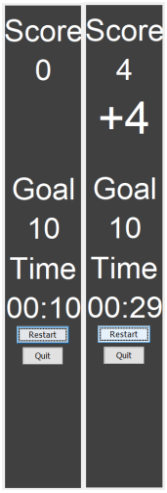

Figure 10: The Status Panel

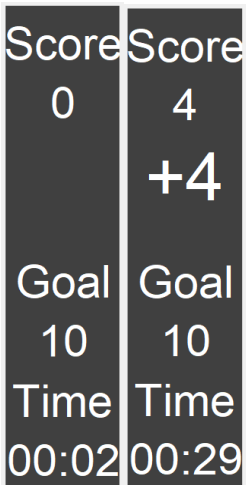

Figure 11: The Data Frame

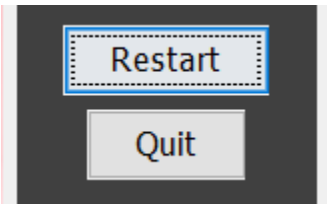

Figure 12: Two buttons: Restart and Quit

Figure 13: The Game Win Interface

B. Game Rule

- Here is all of the basic guide and rule of the game “Fes challenge”:
  - ❖ The rules for the foundational game are straightforward, emphasizing the matching of similar items, represented by different festival items in this application.
  - ❖ The game starts with a grid of objects awaiting matching.

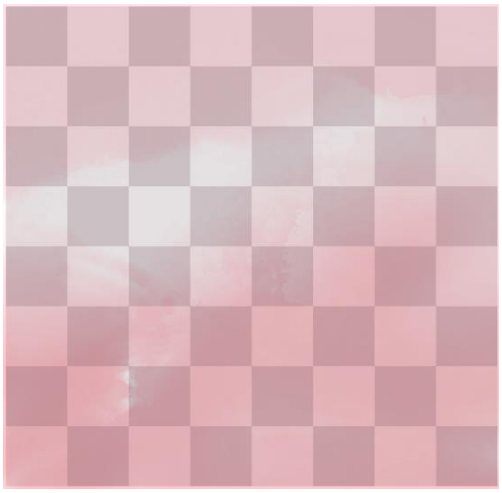

Figure 12: The grid

Figure 13: Festival Items

- To initiate a swap, a player selects an item on the grid and an adjacent one (up/down/left/right). The swap occurs only if it results in a match from at least one moving cell; no swaps happen under other conditions.

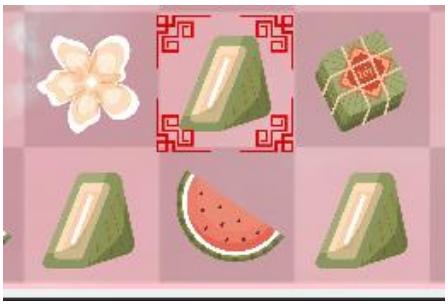

Figure 16: The horizontal row here can be matched

- A match is defined as any connected group of cells, horizontally or vertically, where the same item appears three or more times consecutively.
- Upon a match, the matched items are removed from the board, then two colors dark green and yellow appear in those locations that mark the positions of removed items. All cells above fall from the top to fill the spaces left by the removed cells

Figure 17: Before match

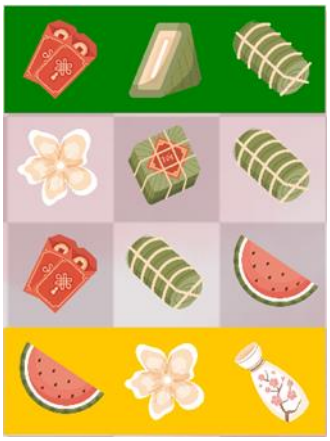

Figure 18: After match, this figure corresponds to the below instruction.

- New items fill in from the top to occupy the gaps after the cells have fallen. This process of resolving matches continues iteratively until no more matches are possible.
- For each matched item, the player earns a score depending on the number of the match item vertically or horizontally.

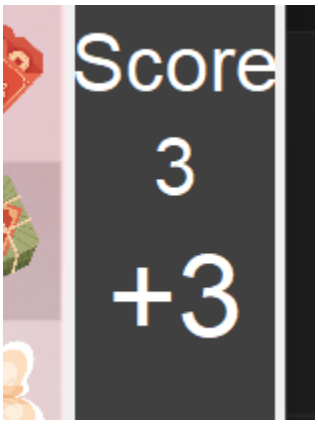

Figure 19: After match, earn a bonus and sum to the total score

- Next, after the player reaches the target score in the level, the winning panel will appear on the screen. The screen will appear “You win!”.

Figure 20: After the player achieves the target, winning!

- In cases that the time is due or the player can not achieve the target score, the system will restart score and time.
- Here is some tip and trick:
  - ❖ Thinking about the impact of each move on the entire board and your level objective
  - ❖ This is all the instructions of the Fes challenge for new players.

C. How the Core Game Works

- To illustrate how the MatchBoard and MatchPanel classes work, this section will dissect their functionality. Position will go unnoticed because it just acts as a coordinate to link with x and y; consequently, further explanation is not required. First off, the MatchBoard defines much of the core code required to implement the game.

- The MatchBoard is constructed in an easy-to-understand way when the game initially launches. The board is simply filled with random numbers, ranging from 0 to the maximum number that the class has passed, in each cell for a specific width and height. Here, the length of the colors array of the MatchPanel will be given to it so that each number corresponds to a distinct color. The MatchBoard's behavior is defined by two sorts of methods. The methods for all the matching, all the switching, and all the rest. To begin with, the following methods fall under the everything else category:

  • setEnforceAdjacent(): This is an extra function used to change the rules of the game if necessary. This function changes the rule around swapping to allow MatchBoard to swap any two tiles. Currently, this function is not used in this project.

- getCellValue(): This function is used to retrieve the integer value saved for a grid cell with a specific x,y position. Used to draw each and every cell.
- fillBoard(): adds a random number to each cell on the board. The following techniques are among those that offer functionality related to matching.(modified)

- The methods that add in functionality related to the matching include the following methods.

  - swapCells(): Switches the values in the cells that are designated by pos1 and pos2. This simply does a swap without doing any validation checks. In addition to carrying out the real swap, it is also used to check for matches in the getMatchesFromSwap() temporary swap.
  - getMatchesFromSwap(): Investigates the possible outcomes of switching the two cells that are described by pos1 and pos2. After temporarily switching the cells, it checks to see if there are any matches before switching them back. The list of changes is sent back to the MatchPanel for evaluation. The rows and columns localized to the swap are the only ones that are searched in this constrained way.
  - shuffleDownToFill(): Obtains a list of all the spots that were designated as matches, fills in the gaps at the top by moving down cells, and then outputs a list of the positions that are now empty.
  - fillPositions(): This function fills all given slots with new random numbers ranging from 0 to (maxNums - 1). This frequently applies to fill the cells following a shuffle down.
  - findAllMatches(): Searches the entire board on every row and every column for matches.
  - getMatchesOnRow(): Adds all matches of three or more in a row found in a single row to the list of matched items.
  - getMatchesOnColumn(): Searches a single column for all matches of 3 or more in a column and adds them to the list of matched elements
  - shuffleDownToFill(): An alternative implementation of shuffleDownToFill that descends all cells above a given point starting at a single location
  - getAffectedPositionsFromAffectedColumns(): Calculates the number of cells that were shifted downward for each column and placed at the top of each column.

- As you can see from looking through the set of methods, the MatchBoard class has all the capability required to execute the matching result and obtain the information required for matches. However, it doesn't help with point selection or interaction itself. All of it takes place within the MatchPanel. The MatchPanel converts the grid of numbers on the MatchBoards into a collection of gems that match the numbers exactly. 0 is image_1, 1 is image_2, etc, ...

- The MatchPanel controls the MatchBoard by performing the selection of positions and directing it to perform the correct set of actions. When the MatchPanel is created it will

start itself with a MatchBoard that is fully randomized. You could make it begin matching right away at the start of the game with a few simple changes, but for the purpose of this version a stable board state is enforced before the player sees this. It means all matches are replaced with new random numbers until there is no longer any match on the board. This may take a few iterations, but it means the player starts with an action right away. A mouse listener is set to begin listening to change between states from selecting cells and the event timer for flowing through multiple stages of matching is configured, but not started. The remaining parts will split up the methods used to create the MatchPanel into similar types. Firstly, the methods used to draw the game, and then the methods used to provide interaction.

- paint(): To generate a visual indication, draw recently matched and recently filled cells from the top cells designated with a rectangle, followed by all the cells. If the game state is ChoosePos2, the method will add a + sign to the cell where pos1 was placed.
- drawAllCells(): Iteratively searching through each cell in the grid, it looks for the proper color to draw a circle with from the colors array using the int values stored in those cells. Calling drawCell() takes care of each individual cell in this way.
- drawCell(): This method helps draw an image corresponding to the value of the cell at a specific location on the matchBoard in the graphical interface of the game Fes Challenge.
- drawRecentMatches(): Draws recently matched cells with orange and recently filled cells with dark green. This function uses various colors to distinguish two different cells.
- drawSelectedPos1(): Creates a white plus sign over the cell containing position one. This serves to draw attention to the player's first of two options.
- drawGrid(): Not utilized, but will create a line grid connecting each cell. The other techniques offer functional modifications according to the interaction that results from clicking the mouse or from pressing the restart button.

- The rest of the methods provide functional changes based on the interaction stemming either from the mouse clicks, or the restart button.
  - mouseClicked(): This is the important function used to change the game's states. It will handle mouse clicks to set pos1 and pos2 for cell swapping in an attempt to transition between states. When two suitable cells are chosen, attemptCellSwap() is used to examine if they can be switched. This function will also call a repaint() if it makes any changes.
  - restart(): Called from the Game object via the StatusPanel’s Restart button. This method will reset the entire current board to a new puzzle and reset the score to 0.
  - setPos1(): Sets the value of pos1 to be swapped before switching to the subsequent state (choosing pos2).
  - setPos2(): Switches to the next state (pause for destroy) after setting the value of pos2 to be utilized for switching.
  - attemptCellSwap(): The attemptCellSwap() method swaps cells at specific positions on a

match board and updates the score based on resulting matches. If matches occur, it performs the swap, fills gaps, and stores information for drawing recent matches. A timer is started for game state transitions. If no matches are found, it prints a message and resets the game state for the player to choose a new first position.

- updateScore(): raises the score by a certain amount before using notifyScoreUpdate() to send the updated score to the StatusPanel via the Game object.
- isValidPosition(): Verifies if the location of the mouse click on the board is a legitimate set of coordinates.
- createStableBoardState(): The createStableBoardState() method iteratively addresses matches on the game board by filling the corresponding positions. It continues this process until no more matches are found, ensuring the board reaches a stable state. The method is crucial for establishing a valid and playable game board by resolving initial matches. (done)
- configureTimer(): The timer activates, it will find all matches on the board. Then update and start the timer again. Once there are no more matches, the game status is swapped back to choosing pos1 (modified).

- And another functionality of the class will describe in below:

SoundMusic class:

- Class creates a new thread for audio playback. Inside the thread, it uses a loop to play the sound specified by the "Sound_path" for the number of times specified by "caseSound".The playback uses the AudioSystem and SourceDataLine classes from the Java Sound API.The playback loop runs until the specified number of repetitions (caseSound) is completed or the state is set to 3 (stop).
  - playSound(): This is the function to read the wav audio file and run it with a given number. Class creates a new thread for audio playback.
  - stopMusic(): This function puts the state into 3, indicating that the playback must be stopped. It waited for the topic to repay the ending sound by using thread.join ().

Timer:

- This is the part of the game that sets the limit for the player and decides whether the player reaches the goal before the specified time. It is placed in the same frame as the score panel to show the remaining time the player needs to complete the game.
  - countdownTimer(): This function creates a countdown timer. Every second, it decrements the value of second by one and updates the time label to display the countdown. If the value of second is smaller than 0, it resets the number of seconds to 59 and reduces the value of minute by one. When the minute and second both reach 0, it stops the countdown, restarts the game, and restarts the countdown automatically.
  - restartTimer(): This function resets the time to 40 seconds and restarts the countdown.

Guide:

- The Guide section offers vital instructions, aiding players in understanding gameplay and navigating the game effectively. It prioritizes user-friendliness, particularly for new players, by providing essential assistance and explanations. Key functionalities within the Guide section include:

  • mousePressed(MouseEvent e) and handleMouseEvent(MouseEvent e): Responds to mouse press events. If the mouse press occurs within the specified area, it calls handleMouseEvent and backToMain.

  • backToMain(): Disposes of the current frame, setting the system look and feel for the UI, creates a new instance of the CandyCrush class

  • mouseMoved(MouseEvent e): Detects mouse movement. If the mouse is over the specified area, it sets the state to back; otherwise, it sets the state to the default state (1).

  • paintComponent(Graphics g): Overrides the paintComponent method to draw the background image based on the current state. If the state is 1, it draws the default guide screen. If the state is back, it draws the guide screen with the mouse over the specified area.

  • mouseClicked, mouseReleased, mouseEntered, mouseExited, mouseDragged: These methods are implemented as part of the MouseListener and MouseMotionListener interfaces but are not used in this specific class. They are implemented to satisfy the interface requirements.

Setting:

- This is a part of the game that allows users to allocate the music of ‘Match sound’ and ‘Main sound’. Other functionalities will be developed to support players who are looking for an excited feeling when playing.

  • mouseClicked(MouseEvent e): Solve the event related to MouseEvent when users click to a specific area in the screen. In the area2, after clicking, game gives user to the Main screen which has a start button, Exit button, Setting and Guide

  • switchBackgroundMusic(): this function has a responsibility for switching the background Music button.

  • backToMain(): Closing the current screen and it opens a new class. In this case, it opens CandyCrush class

  • mouseMoved(MouseEvent e): it is responsible for moving the mouse and showing the painting. For example, the variable ‘state’ is equal to ‘backgroundOn’. It means after moving the mouse to ‘area’, the program draws a new paint.

  • paintComponent(Graphics g): it is a method that is called automatically when a graphic element needs to be redrawn, for example after calling repaint(). This method is responsible for drawing images corresponding to the current state of the user interface based on the value of the state variable.

D. UML Diagram

- These three classes are responsible for showing and playing the Game Area and Counting Score Area.

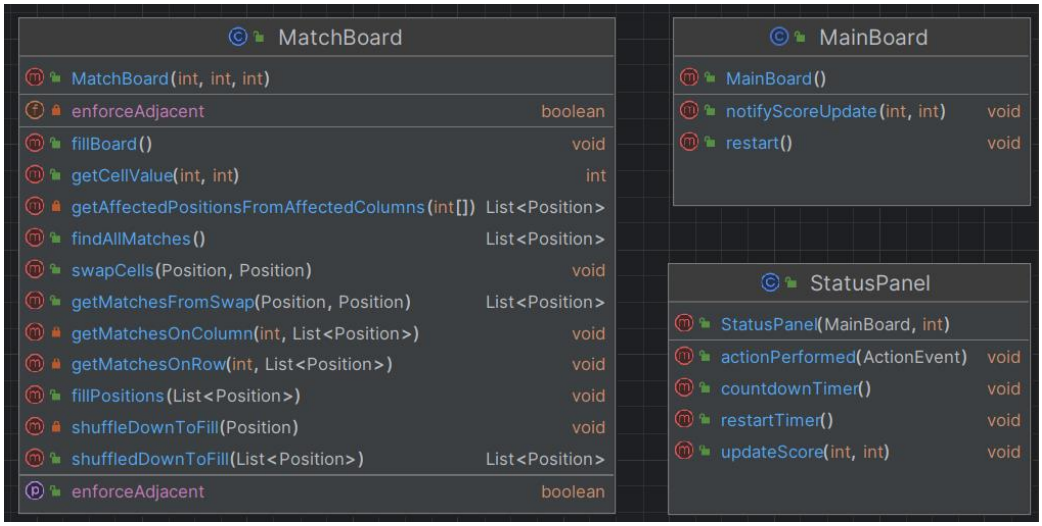

Figure 21: Body_Scence package

- These three classes are responsible for thread of game, location of mouse, swap and other functions

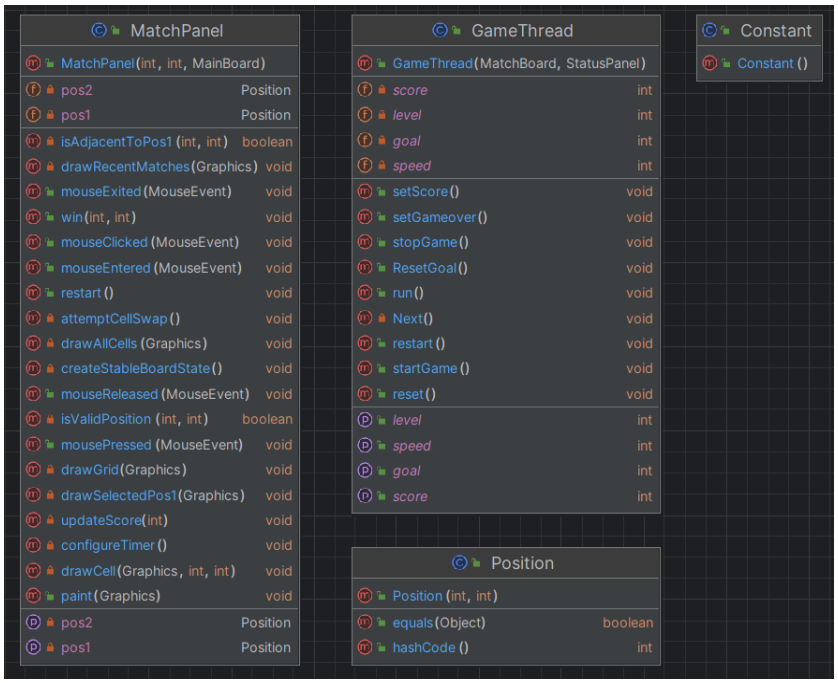

Figure 22: Controls package

- These two classes are responsible for showing the final scene of game including ‘winning screen’ through the operator of players with mouse

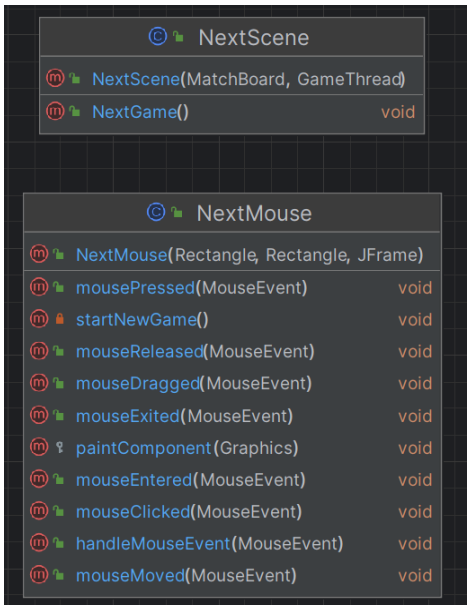

Figure 23: Final_Scence package

- These two classes are responsible for showing the guide scene of game

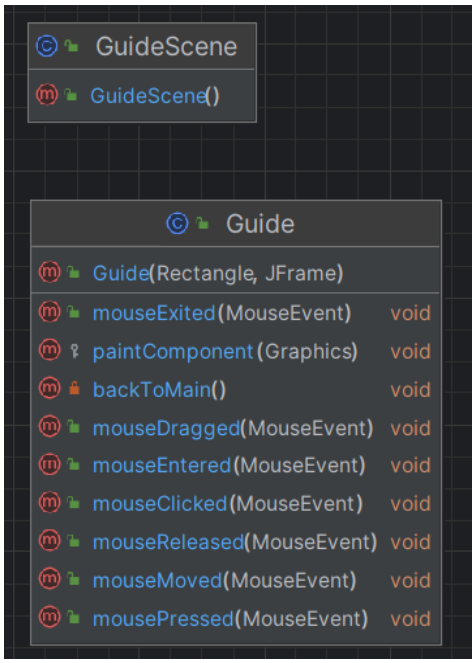

Figure 24: Guide package

- These two classes are responsible for showing the Setting scene of game

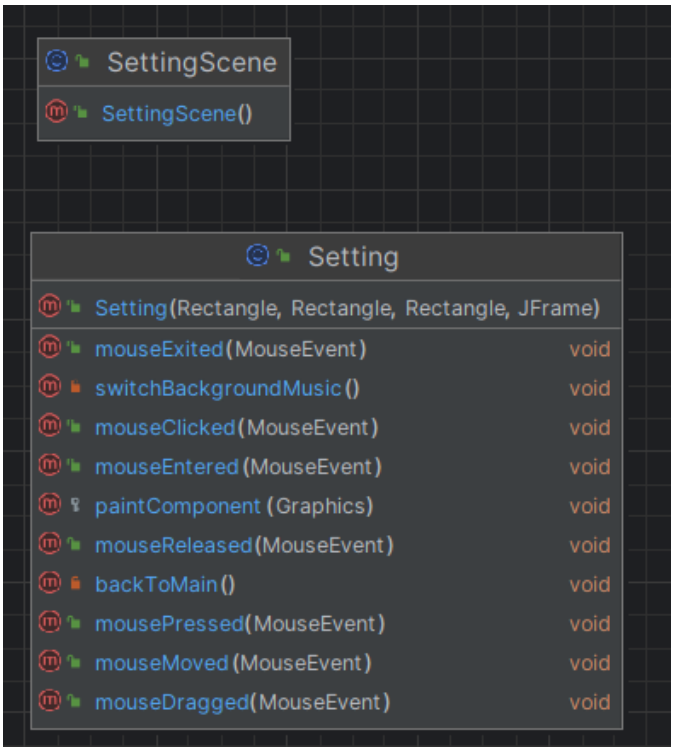

Figure 25: Setting package

- These two classes are responsible for showing the Start scene of the game, buttons to play game or exit, drawing the image of the game.

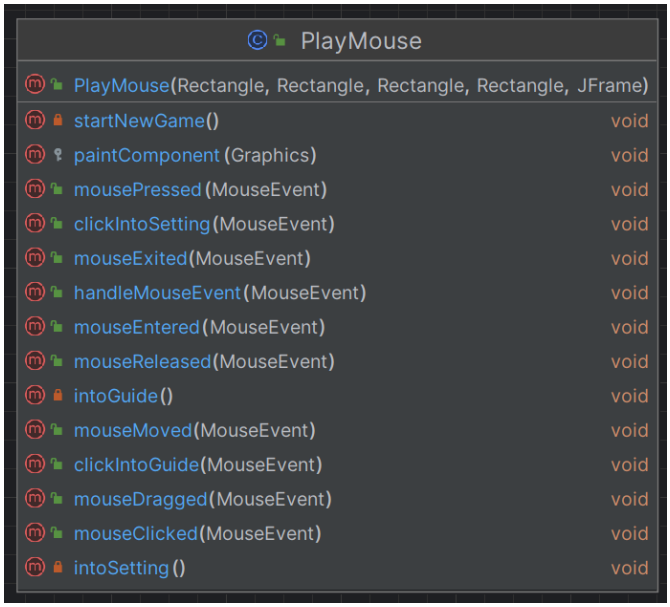

Figure 26: Start_Scence package

Chapter 3: CONCLUSION

A. Summary

In conclusion, the project has been built on OOP features. Through the process of implementing this game project, we have strengthened and practiced coding skills while improving other skills such as teamwork skills and planning for a project. However, the game project has not yet been completed and needs to be supplemented in the future.

B. Shortcoming

Because of the limited time, the game has not been completed yet. When the game is launched, only the 'win scene' appears, while the 'lose scene' has a problem not being displayed and must be changed by restarting the countdown timer as well as the game board. In addition, the game still lacks some features, such as saving game progress (players must start from the beginning), the ability to pause and resume the game, and finally, the code still does not strictly follow the SOLID principles, causing difficulties in developing.

C. Future Works

In the future, team will improve upon these weaknesses by incorporating the following features:

- Save: This feature enables gamers to put their gaming progress on hold and pick it up later.
- Resume: This allows people to continue to play the saved game before.
- Pause: This lets players put the game on hold and come back to it at a later time.
- SOLID code: The code will be modified to adhere to the SOLID principles more closely in order to make it easier to extend and maintain.
- Setting: Active two switches which allow players to turn on/off Background Sound and Matched Sound in the game.
- Level: Level future brings players to more difficult levels via conquering lower challenges.
- Map: This feature shows the movements of overcoming levels and the represents the overall picture of the victories that the player has achieved

Chapter 4: REFERENCE

1) FaTal Cubez. (2014, September 9). *2048 Tutorial Series*. YouTube. https://www.youtube.com/watch?v=GlXWdQSfddw&list=PLig6-gM-fHMGH6jmCpsxW6YbagHgCS3Jd&index=1
2) Freepik. (n.d.). Flat Tet Instagram Posts Collection. Retrieved from https://www.freepik.com/free-vector/flat-tet-instagram-posts-collection_21530587.htm?fbclid=IwAR0slxlDFsed8UTER4FeVvhrC81_xiHaiZZh6AGYxiciL076ckb-XN2zOzc#query=t%E1%BA%BFt%20vi%E1%BB%87t%20collection&position=27&from_view=search&track=ais&uuid=5a61856f-9304-456c-96a0-55ae7107b2d5
3) RyiSnow. (2021, April 14). [Java Code Sample] Create timer (normal/countdown/two digits). https://www.ryisnow.online/2021/04/java-beginner-code-sample-create-timer.html
4) freepik. (n.d.). *Free vector: Flat tet instagram posts collection*. Freepik. https://www.freepik.com/free-vector/flat-tet-instagram-posts-collection_21530587.htm?fbclid=IwAR260N5-DTMebfjNORBDb9l2AfcQ3XofEMkP64pBlacvxTdsOPzjf-6LpNg#query=t%E1%BA%BFt%20vi%E1%BB%87t%20collection&position=27&from_view=search&track=ais&uuid=5a61856f-9304-456c-96a0-55ae7107b2d5
5) Smith, D. (2014). *Information about Candy Crush*. https://www.theguardian.com/science/blog/2014/apr/01/candy-crush-saga-app-brain. https://www.theguardian.com/science/blog/2014/apr/01/candy-crush-saga-app-brain
6) King. (2018, June 19). Video Game Music. Candy Crush Saga. https://downloads.khinsider.com/game-soundtracks/album/candy-crush-saga-gamerip