SCHOOL OF ENGINEERING AND TECHNOLOGY

FINAL ASSESSMENT FOR THE BSC (HONS) INFORMATION TECHNOLOGY; BSC (HONS) COMPUTER SCIENCE; BACHELOR of SOFTWARE ENGINEERING (HONS)YEAR 2

ACADEMIC SESSION 2024; SEMESTER 3

PRG2104: OBJECT ORIENTED PROGRAMMING

Project **DEADLINE: Week 14**

INSTRUCTIONS TO CANDIDATES

- This assignment will contribute 50% to your final grade.
- This is an individual assignment.

IMPORTANT

The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

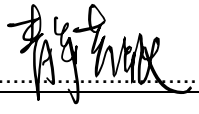
- Coursework submitted after the deadline will be awarded 0 marks
-

Lecturer's Remark (Use additional sheet if required)

I..... (Name) .....std. ID received the assignment and read the comments..... (Signature/date)

Academic Honesty Acknowledgement

"I, Yeap Ching Wei, verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties *(refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme)* for any kind of copying or collaboration on any assignment."



…..... (Student's signature / Date: 22 Aug 2024)

Table of Contents

1. GitHub Information

Username : 21047451

Link : https://github.com/sunwaydcis/finalprojectv2-21047451

2. Presentation Video Link

https://youtu.be/oK8huKXC_lA?si=QMoJUAJOp0OUSYYa

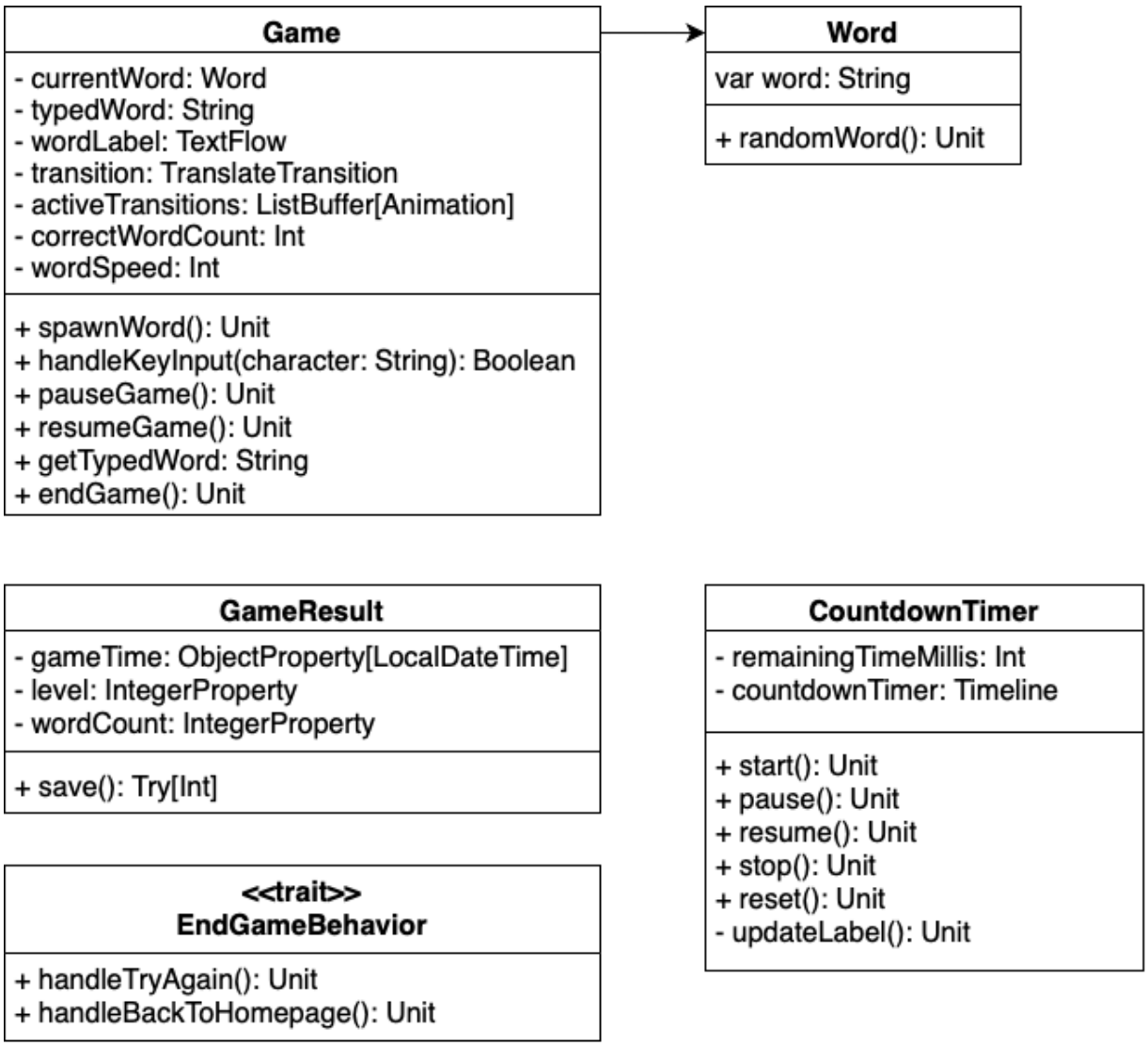
3. Introduction

In my project, I did a typing game called “NAMANANA”. I got the inspiration and design ideas for the game GUI from a music video, “NAMANANA” of artist “Lay Zhang”. I combined the desert theme design and the game logic of typing word to process the project. This game will have two different levels, default speed and sped up level. The game is set so player can play the sped up level only when they completed the default level.

4. Objectives

The primary objective of this typing game is to enhance the typing skills in a challenging and an enjoyable manner. It can help player to enhance typing accuracy by encouraging careful and correct key presses. This game helps in developing hand-eye coordination and fine motor skills required for typing which may improve focus and attention through timed challenges and quick responses.

5. UML Diagram



6. Architecture and Functionalities

"NAMANANA" typing game is structured in three different packages, which are view, util, and model. Table below stated the overview of the classes within each packages and their functionalities.

Package	Classes/Traits/Objects	Functionalities
View	FXML Files	Define the GUI of the game.
Controllers	MainApp	Control the whole program such as the methods used for FXML window interchange.
	HomePageController	Connected to Homepage.fxml to control the related buttons, manage the background of the window, logo and character animations.
	GameController	Connected to Game.fxml to manage game logic and user interface interactions.
	ShowResultsController	Connected to ShowResults.fxml which listed all the past game results.
	HowToPlayController	Connected to HowToPlay.fxml which to show the rules and guidelines of the game.
	FirstLevelEndGameController	Connected to FirstLevelEndGame.fxml where the game result provided and buttons for players to choose to change their location in the game. This controller

		guides player to second level of the game.
	SecondLevelEndGameController	Connected to SecondLevelEndGame.fxml where game result provided and buttons for players to choose to change their location in the game.
Util	Database	Set up a database to store game results.
	MusicPlayer	Set up media player to play background music and sound effects.
Model	Countdown Timer	Create a countdown timer which used in the game interface.
	EndGameBehavior	A trait that contains methods which the FirstLevelEndGameController and SecondLevelEndGameController have been overridden.
	Game	Game logic is located and implemented here for convenient modification and optimization.
	GameResult	Create the table, retrieve results, and initialize instances of “GameResult”.
	Word	Word related method to randomly select a word from a word.txt.

7. Program Features

7.1. Homepage

Figure above shows that when cursor is placing on a button, the button gives response to show that the player is pointing on it by changing the word color. A sound effect is applied when the button is clicked. Homepage is constructed with four buttons:

- Start Game Button

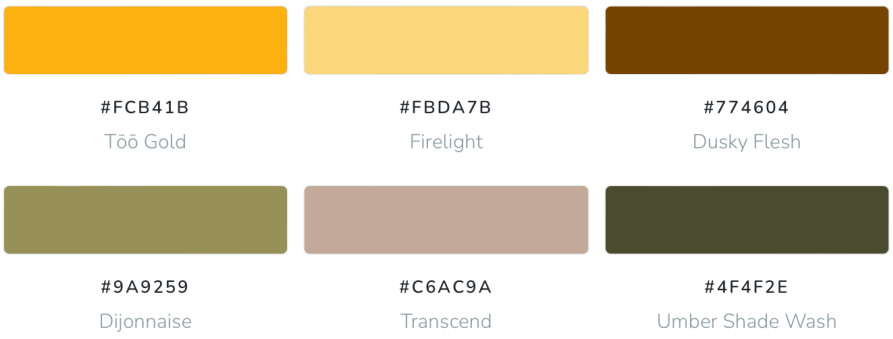
  Start the game where the function linked will call the game to start. Player can start to practice their typing skills by clicking this button.
- Results Button

  This button is linked to a page where all results can be found.
- How to Play Button

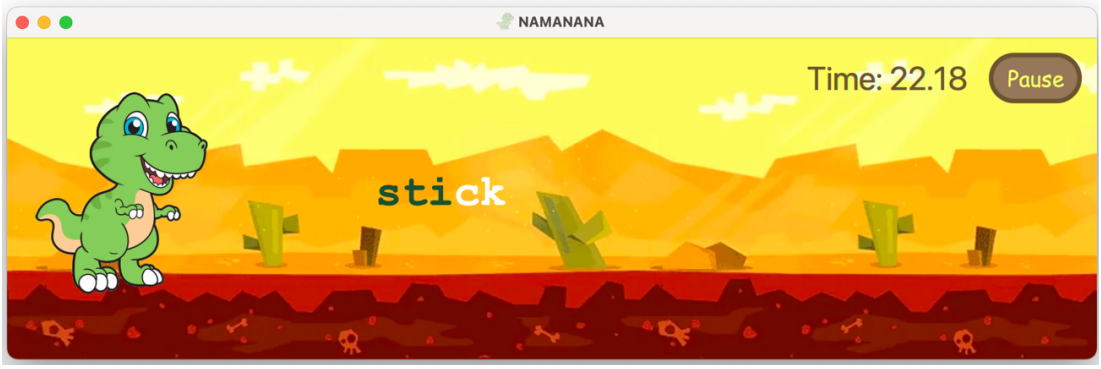
  This button provides rules and guidelines to the player who wanted to know how to play the game.
- Music Button

  This button is used to turn the background music on and off.

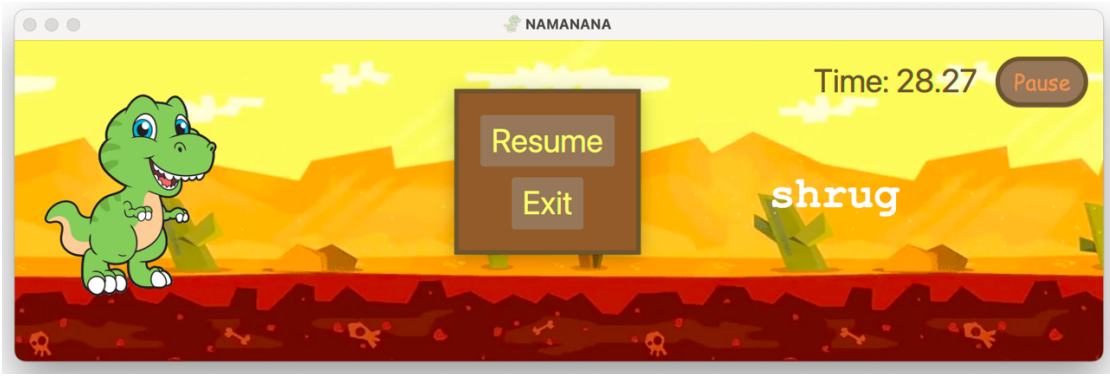
These are the colors that have been used in this game.



7.2. Game



The word will spawn out from the rightmost and slide to the leftmost of the window. All the FXML window are 739*295 except this game window, which is 1000*295. The character which be typed will turn into dark green color, which is #295135. When the whole word is typed correctly, it will disappear, otherwise, it will just slide to the leftmost of the window. A sound effect is applied when the correctly typed word disappeared. The countdown timer is set to start at 30 seconds and will drop to 0. When the times up, the game will jump to end game window to display the number of correct word typed. The dinosaur does not have any function but moving left to right to make a sense that it is interacting with the player.

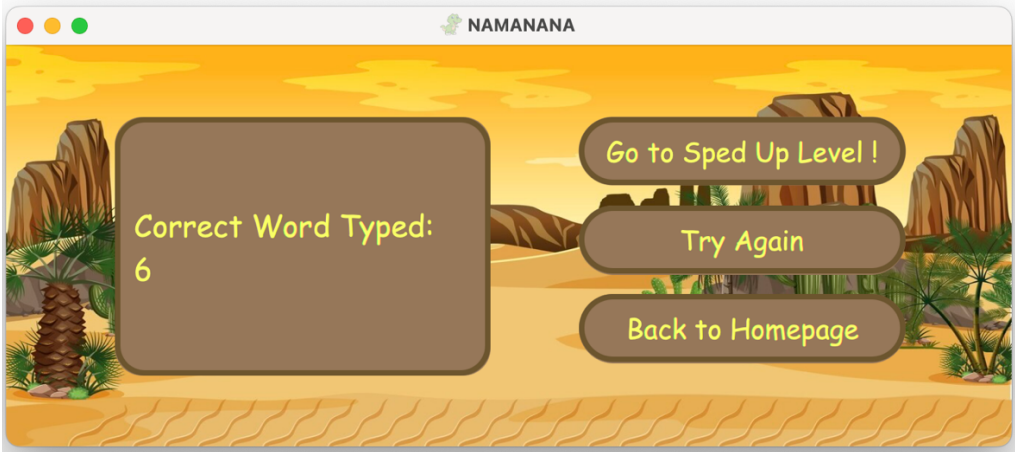


When the “Pause” button is pressed, a pop-up box will appear and all animation, which are the background animation, dinosaur left-right movement, word spawning and sliding across the window, and keyboard detection all will be paused. The pop-up box is set to be no button to close this window. Therefore, the player only can choose either continue or exit and back to the homepage. When the pop-up window is showing, the game window is not allow to control by the user, which means that the user cannot force quit from the game window.

7.3. End Game Window

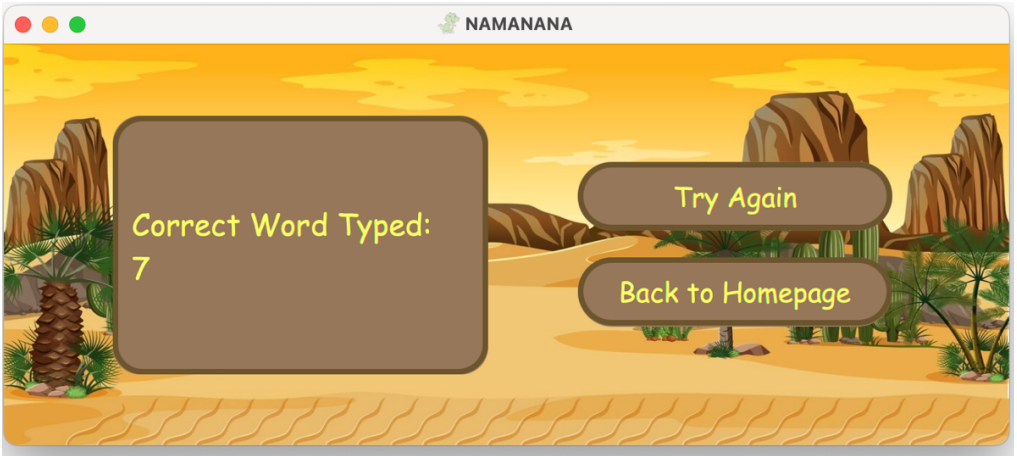
This game will have two levels as told in the introduction, therefore there will be two different end game windows.

7.3.1. First Level End Game Window



This is the first level end game window. It will show the number of correct typed word. There are also three buttons provided. The “Go to Sped Up Level!” button is a way to let the player to go to the next level. The duration for the word to travel from rightmost to leftmost window is used to set the speed of the game. For this first level end game, the duration is set to 6000 milliseconds. When the “Try Again” button is clicked, it will let the player to try again the first level game. For the “Back to Homepage”, it will lead the player to the homepage.

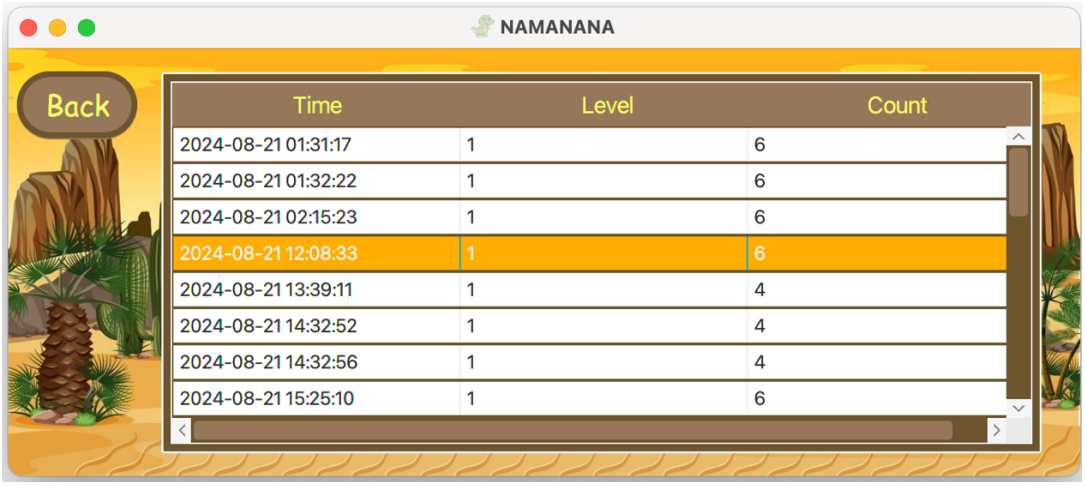
7.3.2. Second Level End Game Window



This is the second level end game window. This “Try Again” button is to try again the second level game. The second level game duration is set to 4000

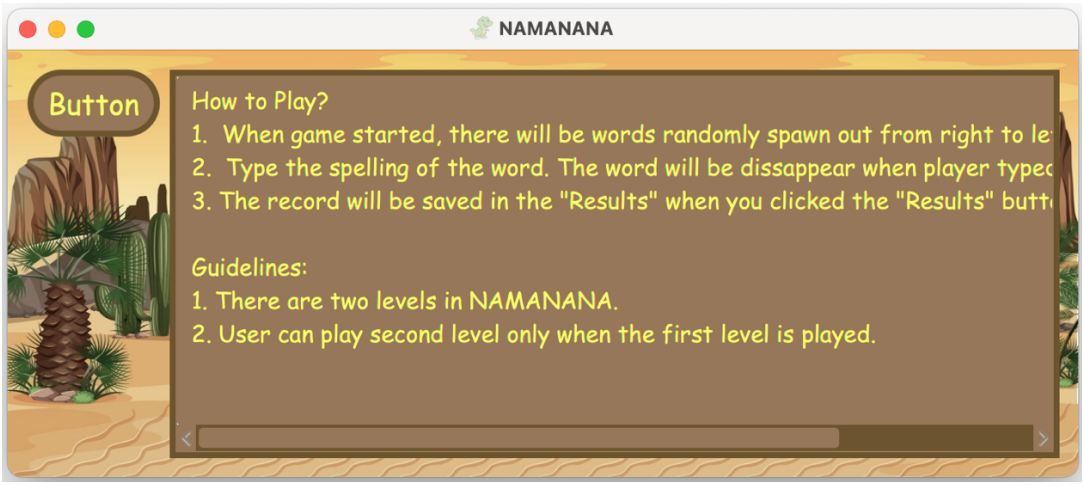
milliseconds, which the speed of the word to pass through the window is a faster than the first level game.

7.4. Results Window



This window is opened when the “Results” button is clicked to view the past game results. When a particular result is clicked, it will change the color of the whole line, including the color of the word.

7.5. Show Rules Page



This page is opened by clicked the “How to Play” button in the homepage. This page is to tell player how this game goes and guidelines to reach second level.

8. Game Functionalities

8.1. Game Pause & Resume Functionality

The pause game functionality in the game window allows player to take a rest during the game. It ensures that when the game is pause, nothing in the window is moving, including the countdown timer.

8.2. Sped-Up Game

The sped-up game which can be play after the first level game is to ensure that the player is familiar with the game and well-prepared for the second level game which is more challenging.

8.3. Character Highlight on Correct Input

When the player in typing the spawned out word, if the character is typed correctly, the character will change the text-fill colour to show that the player is correctly typed. The typing output is observable.

8.4. Database Data Retrieval Configuration

The results are saved and retrievable from the database. All the results can be found in the results window. At the end game page, the result will also display for the player respectively.

9. Object-Oriented Concept

9.1. Inheritance

Inheritance allows classes to inherit characteristics from other class or traits. In this typing game, the “FirstLevelEndGameController” and “SecondLevelEndGameController” inherit from the “EndGameBehavior” trait.

```scala
package cword.model

trait EndGameBehavior {
  def handleTryAgain(): Unit
  def handleBackToHomepage(): Unit
}
```

```scala
@sfxml
class FirstLevelEndGameController(private val goToSpedUpButton: Button,
                                  private val tryAgainButton: Button,
                                  private val returnButton: Button,
                                  private val numWordTyped: Label) extends EndGameBehavior{
```

```scala
class FirstLevelEndGameController(private val goToSpedUpBut

  override def handleTryAgain(): Unit = {
    clickButtonSoundEffect()
    MusicPlayer.pauseBackgroundMusic()
    MainApp.startGame(1, 6000)
  }

  override def handleBackToHomepage(): Unit = {
    clickButtonSoundEffect()
    MainApp.showHomepage()
  }
}
```

```scala
@sfxml
class SecondLevelEndGameController(private val tryAgainSecondLevelButton: Button,
                                   private val returnButton: Button,
                                   private val numWordTyped: Label) extends EndGameBehavior{

  MusicPlayer.playBackgroundMusic()

  numWordTyped.text = "Correct Word Typed: \n" + MainApp.gameData.last.wordCount.value.toStri
  numWordTyped.style = "-fx-font-size: 22px; -fx-text-fill: #faff7f; -fx-font-family: 'Comic

  override def handleTryAgain(): Unit = {
    clickButtonSoundEffect()
    MusicPlayer.pauseBackgroundMusic()
    MainApp.startGame(2, 4000)
  }

  override def handleBackToHomepage(): Unit = {
    clickButtonSoundEffect()
    MusicPlayer.pauseBackgroundMusic()
    MainApp.showHomepage()
  }
}
```

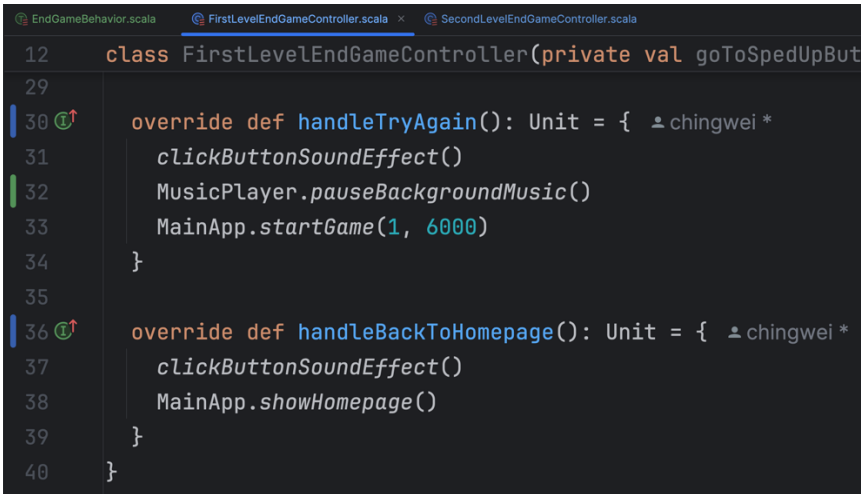
The “FirstLevelEndGameController” class extends “EndGameBehavior”, requiring it to provide concrete implementations of the “handleTryAgain()” and “handleBackToHomepage()” methods, specific to the first level of the game.

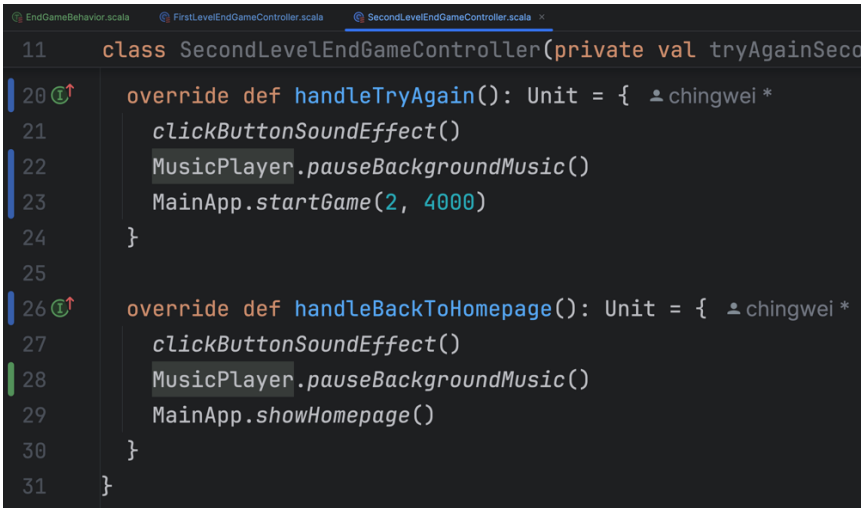
For the “SecondLevelEndGameController”, it also extends “EndGameBehavior”, implementing these methods according to the second level’s requirements.

9.2. Polymorphism

Polymorphism allows us to treat objects of different classes through a common interface or trait. This concept is demonstrated through the use of "EndGameBehavior" trait in this typing game.

By implementing this trait, other controller classes can provide their own specific implementations for these methods.





10. Personal Reflection

10.1. Strengths

10.1.1. Coding Structures

This typing game is well-structured into different classes and packages, which are model, view, and util. This separation of concerns makes the code more maintainable and easier to manage. Object-oriented concepts are well-applied by encapsulating functionality within classes like “Game”, “Word”, and “CountdownTimer”.

10.1.2. Code Reusability

The use of utility objects like “MusicPlayer” and “Database” allows for code reusability across different parts of the game. From the methods like “playBackgroundMusic()” and “clickButtonSoundEffect()” can be used in various controllers, avoiding redundancy.

10.1.3. GUI

The implementation of animations such as the “ScaleTransition” for the title and dinosaur at the homepage and media handling such as the background video in the Game.fxml enhances the visual appeal of the game. The game window features a responsive design with a countdown timer and pop-up window for pausing.

10.1.4. Database

The database utility demonstrates an understanding of persistent data storage, enabling the game to save and retrieve results.

10.2. Weaknesses

10.2.1. Coupling between Classes

The main object directly manages game state transitions and interacts with controllers, which introduces tight coupling between the “MainApp” and view controllers. This can make the code less flexible and harder to modify.

10.2.2. Limited Error Handling

The code lacks with comprehensive error handling, especially in file operations like reading from word.txt. This could lead to runtime exceptions if the file is missing or incorrectly formatted.

10.3. Problems Encountered during Development

10.3.1. Multiple Endgame Messages

There is an issue of multiple “Ending game…” messages when the game was tested running. The message was printed for debugging purpose. This issue was due to the “endgame()” method being triggered multiple times.

10.3.2. CSS Integration

The elements that are not in scene builder cannot be list in the CSS file. Therefore, the inline code is unable to move into the CSS file.

11. Conclusion

This typing game is successfully developed as the original idea, which involved creating two levels using ScalaFX and JavaFX. The code structure includes the “MainApp” for managing the interchanging of the window, controllers to control and interact the user interface, and others to handle logics and functionalities. This utilizes the object-oriented programming concepts. A user-friendly UI makes this game more attractive and easier to read and handle by the player. Overall, this game provided a great environment to enhance the typing skills of the player to higher their efficiency of practice.

12. References

12.1. Concept References

[1] OpenAi, “ChatGPT,” *chatgpt.com*, 2024. https://chatgpt.com

12.2. Background Music & Sound Effect Resources

[1] “- YouTube,” *Youtu.be*, 2024. https://youtu.be/yY6bNTYDwI4?si=6zO_30HAy6c-rDvN (accessed Aug. 22, 2024).

[2] “- YouTube,” *Youtu.be*, 2024. https://youtu.be/8usQCG6WHzE?si=6r3a22lNzYDcPdFa (accessed Aug. 22, 2024).

[3] “- YouTube,” *Youtu.be*, 2024. https://youtu.be/YNSbL-Cek1c?si=ozPHBUI-tqitMKKA (accessed Aug. 22, 2024).

12.3. Background Video & Image Resources

[1] “3D Logo and Text Effects Generator | TextStudio,” *www.textstudio.com*. https://www.textstudio.com

[2] “Free Vector | Desert with palms and cactus nature landscape scene,” *Freepik*, 2020. https://www.freepik.com/free-vector/desert-with-palms-cactus-nature-landscape-scene_10803668.htm#query=cartoon%20desert&position=2&from_view=keyword&track=ais_hybrid&uuid=5832476c-0767-4262-b51b-8476ddee63f3 (accessed Aug. 22, 2024).

[3] Adobe Stock, “Adobe #549242675,” *Adobe Stock*. https://images.app.goo.gl/XfETKZAEiS3cyXh88 (accessed Aug. 22, 2024).

[4] “Image Color Finder | Pick Colors From Images | HEX, RGB & HSL,” *imagecolorfinder.com*. https://imagecolorfinder.com

[5] “Behance,” *Behance.net*, 2023. https://www.behance.net/gallery/65838927/Game-Backgrounds (accessed Aug. 22, 2024).