**Visual Novel**

**Visual Novel Project**

This project is a visual novel game, where players experience a story with interactive elements. The game consists of multiple chapters, each containing text, character sprites, and background music.

**Game Mechanics:**

* The player progresses through the story by clicking the **Next** button.
* Characters appear with different expressions, updating dynamically based on the dialogue.
* Background music and sound effects enhance the storytelling experience.
* Some chapters may include **choices** that affect the storyline.

**Main Components:**

**1. Game UI Layout**

The application consists of the following main UI elements:

* **Background Image**: Displays the setting for each scene.
* **Character Sprites**: Characters appear with different expressions.
* **Text Box**: Displays dialogue and narration.
* **Next Button**: Advances the story.
* **Choice Buttons** (if applicable): Allow the player to make decisions that affect the story.

**2. Story Progression**

* Each chapter follows a structured format with preloaded story text.
* The game reads story scripts from a text file and displays them dynamically.
* The player clicks **Next** to advance through the chapter.
* If a choice appears, the player selects an option that determines the next sequence.

**3. Code Structure**

The project is organized into an **object-oriented structure**, with the following key components:

**a) Chapter Class**

Each chapter extends the Chapter abstract class, which provides:

* Background setup
* Character image updates
* Speaker visibility control
* Text animation handling

**b) Interfaces**

* **HaveBackgroundMusic** - Handles background music functionality.
* **HaveText** - Manages text animations, dialogue updates, and character speech effects.

**c) Utility Classes**

* **TextBase** - Reads and stores story text from files.
* **GameLogic** - Handles logical conditions such as choices and event triggers.

**4. Gameplay Example**

1. The player starts the game, and the first chapter loads.
2. Background music begins playing.
3. The first dialogue appears with a character sprite.
4. The player clicks **Next** to proceed through the story.
5. At certain points, choices may appear (if applicable).
6. The game continues until the chapter ends, and the next chapter loads.
7. Upon completion, the game either ends or loops back to the main menu.

## **Implementation Details**

**3.1 Package logic**

This package contains the core game logic for the visual novel. It manages game states, player choices, and other relevant gameplay data.

**3.1.1 Class GameLogic**

This class follows the **Singleton pattern**, ensuring that only one instance of the game logic exists throughout the execution of the game. It tracks various game states, including the current chapter, player choices, and relationship values.

**Field**

|  |  |
| --- | --- |
| **Field** | **Description** |
| - GameLogic instance | The single instance of GameLogic |
| - boolean haveMeat | Tracks whether the player has obtained meat. |
| - int currentChapter | Stores the current chapter number. |
| - double arisaLike | Represents the player's relationship points with Arisa. |

**Method**

|  |  |
| --- | --- |
| **Method** | **Description** |
| + getter/setter for some variables |  |

**3.2 Package Util**

This package contains utility classes and interfaces that support the functionality of the visual novel. It includes text handling, background music management, and UI-related utilities.

**3.2.1 Class TextBase**

This class handles the loading and management of story text for the visual novel.

**Field**

|  |  |
| --- | --- |
| **Field** | **Description** |
| + ArrayList<String[]> texts | Stores the loaded story text as an array of strings. |
| + final int emotionIndex | Index for the emotion data in the text array. |
| + final int speakerIndex | Index for the speaker name in the text array. |
| + final int textIndex | Index for the actual dialogue text. |
| + final int soundEffectIndex | Index for the sound effect associated with the dialogue. |
| + final int talkingSoungIndex | Index for the talking sound effect. |
| + final int readingStatusIndex | Index for reading status of the text. |
| + final int quesion1Index | Index for question 1 in case of choices. |
| + final int answer1Index | Index for answer 1 in case of choices. |
| + final int quesion2Index | Index for question 2 in case of choices. |
| + final int answer2Index | Index for answer 2 in case of choices. |
| + final int maxTextIndex | Maximum index value used in text parsing. |

**Method**

|  |  |
| --- | --- |
| **Method** | **Description** |
| + TextBase() | Constructor of this class. |
| + TextBase(String url) | Loads text data from a file specified by URL. |
| + TextBase(ArrayList<String[]> texts) | Initializes TextBase with a predefined text list. |
| + ArrayList<String[]> getStoryTexts() | Returns the list of story texts. |
| + void setTexts(ArrayList<String[]> texts) | Updates the story texts. |

**3.2.2 Interface HaveText**

Defines methods related to text handling and animations in the visual novel.

**Method**

|  |  |
| --- | --- |
| **Method** | **Description** |
| + Timeline createTimeline(TextFlow textBox) | Creates a timeline animation for text display. |
| + void handleNextText(Stage primaryStage, TextFlow textBox, int fromAnswerBox) | Handles transitioning to the next dialogue. |
| + boolean isRunning() | Returns true if a text animation is currently running. |
| + void playTalkingSound(String talking) | Plays a sound effect for character speech. |

**3.2.3 Interface HaveBackgroundMusic**

Defines a method for managing background music in the visual novel.

**Method**

|  |  |
| --- | --- |
| **Method** | **Description** |
| + void playBackgroundMusic(String url) | Plays background music from the given file path. |

**3.2.4 Interface HaveBackgroundImage**

Defines a method for setting up background images in the visual novel.

**Method**

|  |  |
| --- | --- |
| **Method** | **Description** |
| + ImageView setupBackground(String url) | Loads and returns an ImageView for the specified background image. |

**3.2.5 Abstract Class Chapter**

This abstract class defines the base structure for each chapter in the visual novel. implements HaveBackgroundMusic, HaveText. It includes methods for managing background music, character images, and story progression.

**Field**

|  |  |
| --- | --- |
| **Field** | **Description** |
| - TextBase storyTexts | Stores the story text for the chapter. |
| - int currentTextIndex | Keeps track of the current dialogue index. |
| - Timeline timeline | Controls text animation timing. |
| - MediaPlayer backgroundMusic | Background music player for the chapter. |
| - MediaPlayer effectPlayer | Handles sound effects. |
| - MediaPlayer effectTalking | Handles talking sound effects. |
| - StackPane stackPane | Main container for chapter elements. |
| - StackPane choiceBoxStack | Displays choice options if applicable. |

**Method**

|  |  |
| --- | --- |
| **Method** | **Description** |
| *# void startChapter(Stage primaryStage)* | Starts the chapter and initializes UI components. |
| *# void stateSetup(Stage primaryStage)* | Sets up the state of the chapter. |
| *# void updateCharacterImages()* | Updates character images based on the current dialogue. |
| *# ImageView createSpeakerImage(String speaker)* | Creates an ImageView for a character sprite. |
| *# void updateSpeakerVisibility()* | Adjusts character visibility based on the speaker. |
| *# void setStoryTexts(String url)* | Loads the story text for the chapter. |
| *# void goToNextChapter(Stage primaryStage)* | Moves to the next chapter in the story. |
| + void playBackgroundMusic(String url) | Plays background music for the chapter. |
| + void loadSoundEffect(List<String> emotions) | Loads sound effects. |
| + ImageView setupBackground(String url) | Sets up the background image for the chapter. |
| + StackPane createTextBoxStack(TextFlow textBox) | Creates a text box UI element. |
| + Button createNextButton(Stage primaryStage, TextFlow textBox) | Creates the "Next" button for story progression. |
| + void handleNextText(Stage primaryStage, TextFlow textBox, int fromAnswerBox) | Handles text progression logic. |
| + void createAnswerBoxFor2(Stage primaryStage, TextFlow textBox) | Creates a choice selection box with two options. |

**3.3 Package Main**

This package contains the core entry point and main menu functionality of the visual novel.

**3.3.1 Class Main**

This class serves as the entry point for the visual novel application.

**Method**

|  |  |
| --- | --- |
| **Method** | **Description** |
| + void start(Stage primaryStage) | Initializes the main menu and displays it. |
| + static void main(String[] args) | Launches the JavaFX application. |

**3.3.2 Class MenuScene**

This class manages the visual novel's main menu scene, including UI elements and transitions.

**Method**

|  |  |
| --- | --- |
| **Method** | **Description** |
| + void showMenuScene(Stage primaryStage) | Displays the menu scene with title, buttons, and effects. |
| - Button createMenuButton(String text) | Creates and styles a menu button. |
| - void addButtonAnimations(Button button) | Adds hover animations to buttons. |
| - ImageView createImageView(String path, double width, double height) | Loads an image and returns it as an ImageView. |
| - void toNextChapter(Stage primaryStage) | Transitions from the menu to the first chapter. |

**3.3.3 Class Chapter1 - Chapter7**

These classes represent the main chapters of the visual novel. Each chapter extends the **Chapter** class and manages character dialogues, animations, background images, and story progression.

**Method**

|  |  |
| --- | --- |
| **Method** | **Description** |
| + void startChapter(Stage primaryStage) | Initializes and starts the chapter. |
| # void stateSetup(Stage primaryStage) | Sets up the chapter's background, text, and character images. |
| # ImageView createSpeakerImage(String speaker) | Loads and returns a character's image. |
| # void updateCharacterImages() | Updates the displayed character image based on dialogue. |
| # void updateSpeakerVisibility() | Adjusts character opacity based on the active speaker. |
| # void setStoryTexts(String url) | Loads the story text for the chapter. |
| # void goToNextChapter(Stage primaryStage) | Transitions to the next chapter after completion. |

**4. End of Session Review**

This document provides a structured overview of the visual novel project, detailing its key components, including the Util package, Main package, and chapters.