**ANIME ATTAX**

Group 1 - No Inheritance  
 Members-  
 Kunal Mittal (IMT2023533)  
 Nirbhay Sharma (IMT2023103)  
 Ivan Bhargava (IMT2023022)  
 Harsh Gupta (IMT2023121)  
 Gautam Kappagal (IMT2023082)  
 Akshat Betala (IMT2023019)  
 Harsh Modi (IMT2023607)

**INTRODUCTION**

Anime Attax is a thrilling fantasy card game that captures the spirit of anime and competitive strategy. Players create and manage their decks, featuring a variety of powerful characters and unique abilities inspired by anime aesthetics.

The game’s core revolves around building teams, tactically deploying cards, and adapting to ever-changing scenarios. Each match offers a blend of planning and adaptability, ensuring engaging gameplay that appeals to casual and competitive players alike.

Anime Attax is designed to challenge your strategic thinking while immersing you in a vibrant anime-inspired universe. This project delves into its mechanics, artistic elements, and the innovative concepts that make it a standout experience.

It is a JavaFX-based desktop application that provides a customizable card game experience. This project is built with JavaFX and includes a multi-step setup for game configuration, ensuring a smooth and engaging user experience.

**FRONTEND**

**Features**

**Splash Screen**

* Displays a welcoming splash screen upon startup.
* Features a **2-second pause** followed by a smooth fade-out transition.
* Automatically navigates to the Home Page for a polished onboarding experience.

**Home Page**

* Acts as the main hub for the application.
* Includes navigation to game setup, game modes, and potentially a leaderboard.

**Game Settings Setup (Multi-Step)**

* A **step-by-step process** guides players through configuring the game:
  1. **Step 1**: Choose the game mode (Human vs. Computer (future planned) or Human vs. Human).
  2. **Step 2**: Select the number of rounds to play.
  3. **Step 3**: Enter player names to personalize the experience.
* Includes validation to ensure players provide valid inputs at each step.
* Dynamic button behavior adjusts based on the current step (e.g., "Next" vs. "Start Game").

**Game Settings Persistence**

* Game settings (mode, number of rounds, player names) are stored in a GameSettings object.
* Data is passed to the **Terminal Page** for initiating the game.

**Smooth Transitions**

* Navigation between screens is seamless and visually appealing, using JavaFX animations.

**Getting Started**

**Prerequisites**

Before running the project, ensure you have:

* **Java Development Kit (JDK) 11 or later**
* **JavaFX SDK**
* A JavaFX-compatible IDE (e.g., IntelliJ IDEA, Eclipse).
* Maven

**Installation**

1. Clone the repository:
2. git clone https://github.com/Ivan825/AnimeAttax/tree/main
3. Open the project in your IDE and configure the JavaFX SDK:
   * For IntelliJ IDEA:  
      **File > Project Structure > Libraries**, then add the JavaFX SDK folder.
   * For Eclipse:  
      Add the JavaFX library to the **Build Path**.
4. Ensure the following directory structure for resources:
5. src/main/resources/com/example/consolecardgame/FXML/
   * SplashScreenFXML.fxml
   * HomePageFXML.fxml
   * GameSettingsFXML.fxml
   * TerminalPageFXML.fxml
6. Build and run the project to explore its features.

**Usage**

**Launching the Game**

1. Run the project in your IDE.
2. Observe the splash screen before being taken to the home page.
3. Use the navigation options to start setting up a new game.

**Game Setup**

* Follow the **Game Settings Setup** process to configure the game mode, number of rounds, and player names.
* Click "Start Game" to begin playing after completing the setup.

**Navigating Back**

* You can navigate back to the **Home Page** at any point using the "Exit" button.

**Game Settings Walkthrough**

**Step-by-Step Setup**

1. **Step 1: Game Mode**
   * Choose between "Human vs. Human" or "Human vs. Computer" modes.
   * Input is validated to ensure a selection is made before proceeding.
2. **Step 2: Number of Rounds**
   * Use a spinner to select the number of rounds (1–10).
   * Default value is set to 3 for convenience.
3. **Step 3: Player Names**
   * Enter names for Player 1 and Player 2.
   * Validation ensures both fields are non-empty.

**Button Behavior**

* **Next**: Advances to the next step and validates input for the current step.
* **Back**: Returns to the previous step (disabled on Step 1).
* **Start Game**: Appears on Step 3 to finalize the setup and begin the game.

**Home Page Controller**

The HomePageController manages the logic and interactivity of the **Home Page** in Anime Attax. It provides users with a welcoming interface where they can:

* Start configuring a new game.
* Adjust the volume of the background music.
* Exit the application.

**Key Features**

**1. Background Music**

* The Home Page includes **background music** to enhance user experience.
* Music is loaded from a resource file (BGMusic1.mp3) using the Media and MediaPlayer classes in JavaFX.
* The music is set to play in a **loop** (MediaPlayer.INDEFINITE) to ensure continuous playback while the Home Page is active.

**2. Volume Control**

* A **slider** is provided to let users adjust the background music volume dynamically.
* The VolumeSlider is bound to the MediaPlayer's volume property, allowing real-time volume adjustments.
* The slider's value ranges from 0 to 100, with the media player's volume adjusted proportionally.

**3. Start Game Button**

* Clicking the **Start Game** button transitions the application to the **Game Settings Page**.
* The FXMLLoader loads the GameSettingsFXML.fxml file, creating a new scene with the game settings UI.
* Ensures smooth navigation and prepares the game configuration flow.

**4. Exit Button**

* Clicking the **Exit** button closes the application gracefully.
* Stops the background music before the application exits to free resources.

**Technologies Used**

* **Java**: Core programming language for the application.
* **JavaFX**: For building modern UI components with advanced animations.
* **FXML**: XML-based layout language for JavaFX UI design.
* **SceneBuilder** *(optional)*: Visual editor for FXML files.

**Planned Features**

1. **Card Game Logic**
   * Implement mechanics for dealing cards, shuffling decks, and scoring.
2. **Leaderboard**
   * Save and display high scores using a local file or database.
3. **Enhanced Game Modes**
   * Add more modes, such as "Tournament" or "Timed Matches."
4. **Theming Options**
   * Allow players to customize the appearance of the game.

**BACKEND**

**Pre-Requisites**

Before running the project, ensure you have (other than software mentioned earlier):

· GCC compiler to compile C++ code

· Software to import json file into your code (VS code does it automatically)

· json.hpp (Referred Github Repo:<https://github.com/nlohmann/json> )

· jni.h (included in JDK)

· jni\_md.h (included in JDK)

**Technologies Used**

· Java: Main programming language used

· C++: To implement random character selection (needed to be quick)

· JNI: To bridge the main java program with C++ class

· JSON: Used to store anime character information

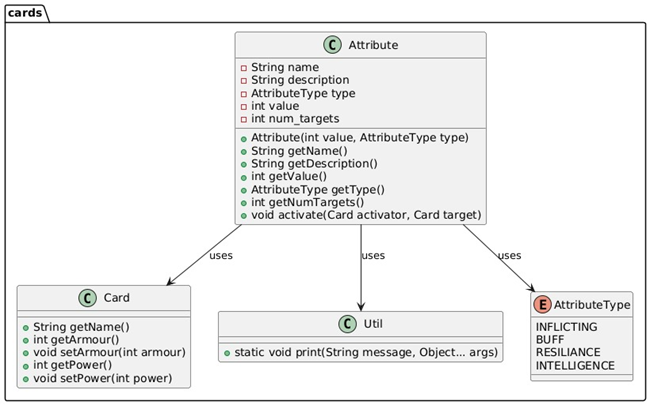
**Future Development**

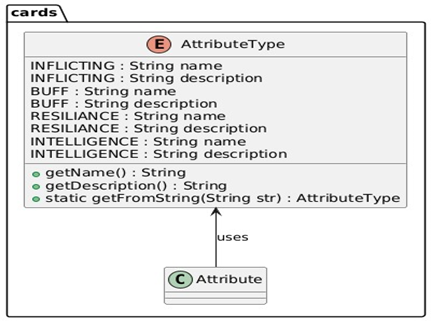
· computer bot therefore made player class and human extending class

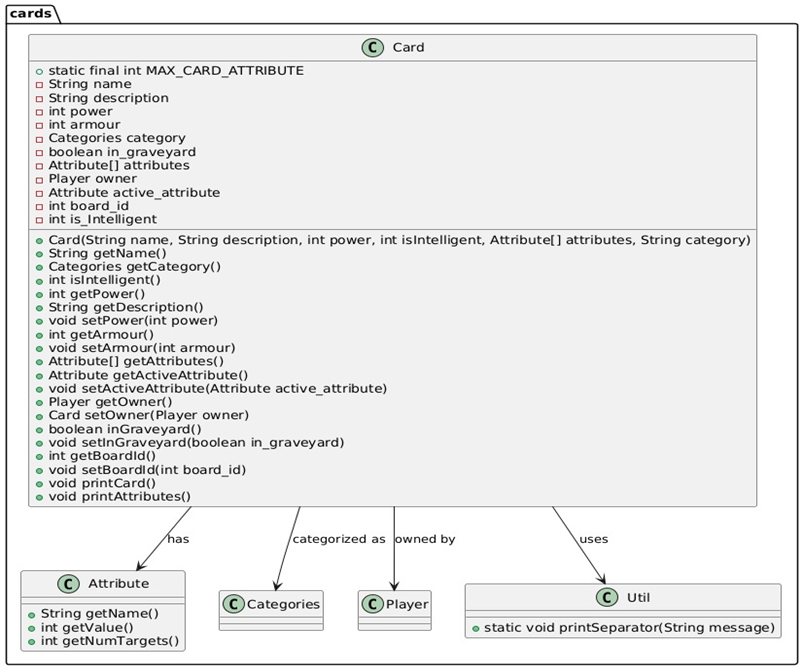
· multiple targets

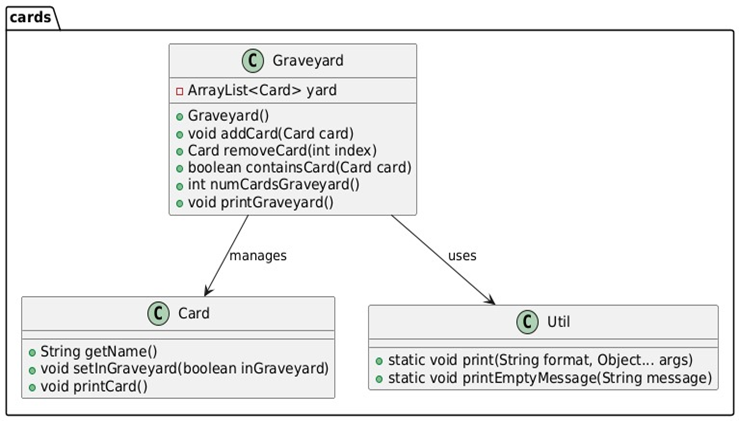
· timed attributes

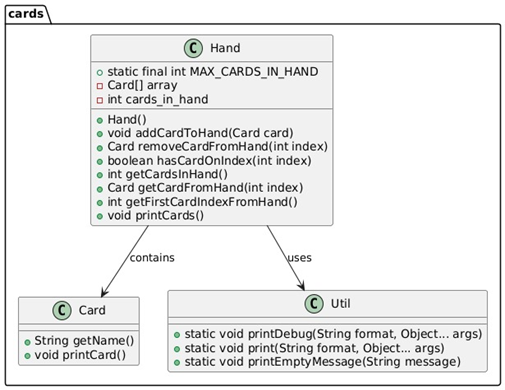
· online version

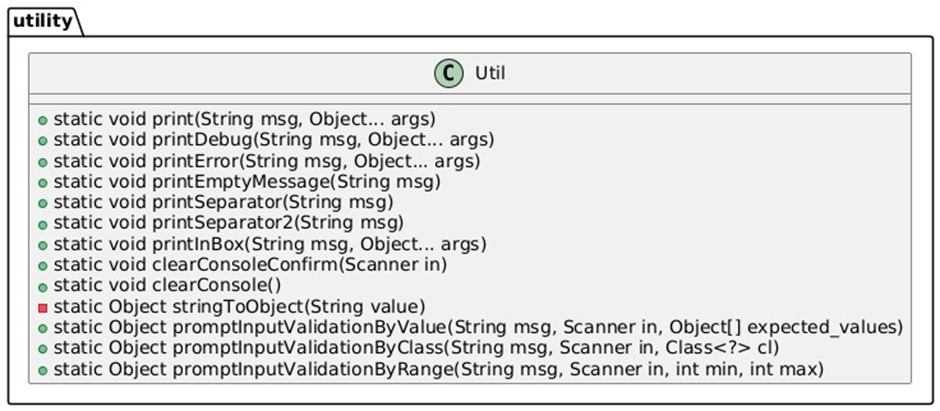
**UML DIAGRAMS**Attached below are the UML diagrams for all the backend codes. Here is the UML diagram for the cards  


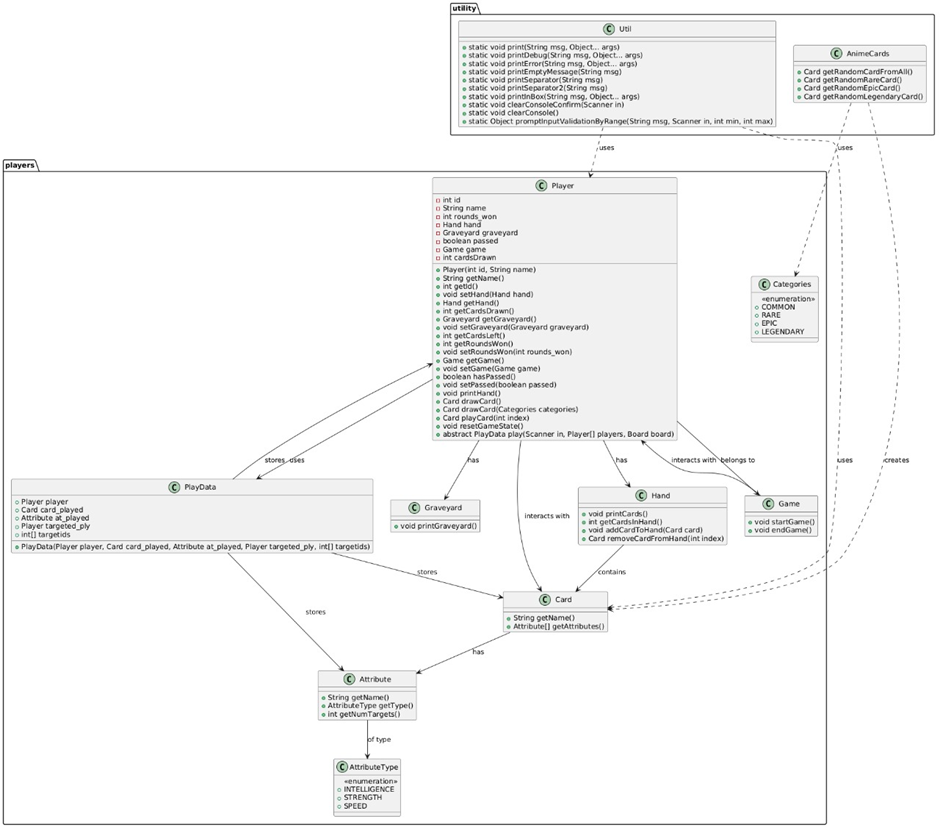
Attached below is the UML diagram for the attributes of the cards.  


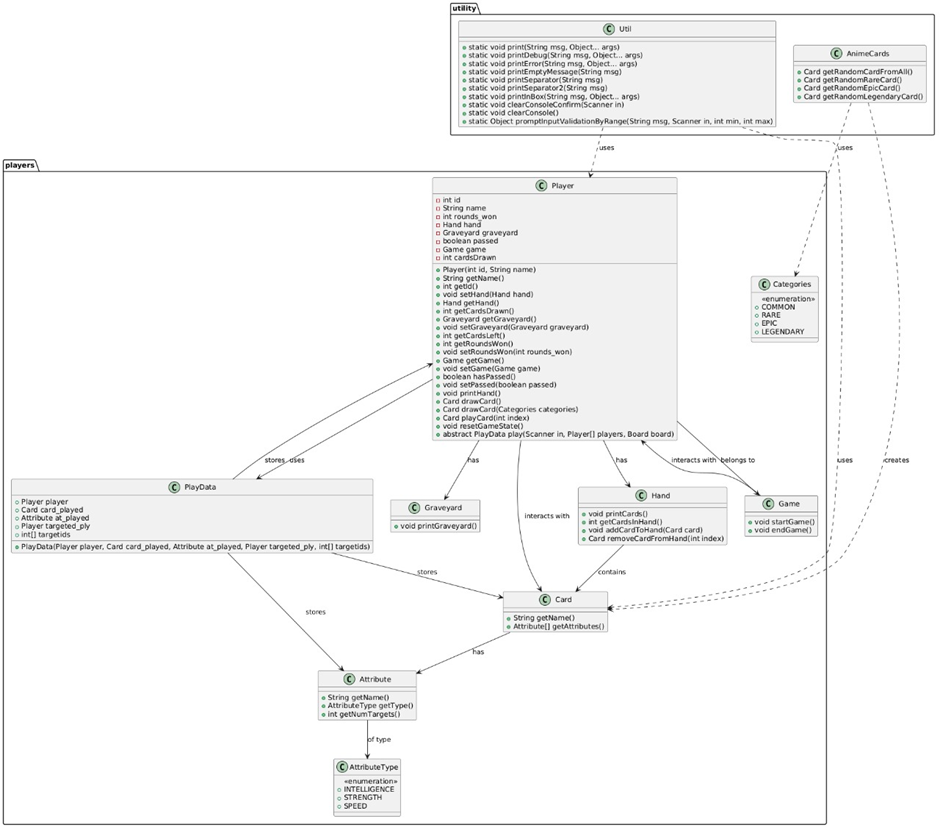
The overall UML diagram for cards  
  


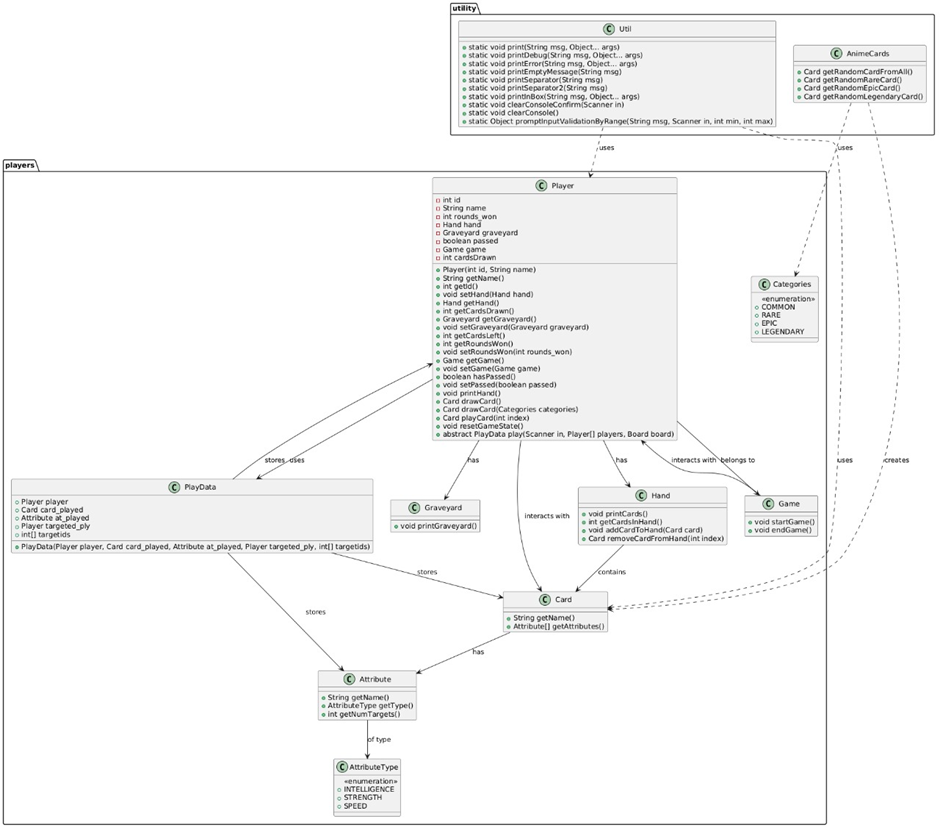
There are obviously various types of cards so below is the UML diagram for Graveyard class

Below is the UML for Hands card type.  
  


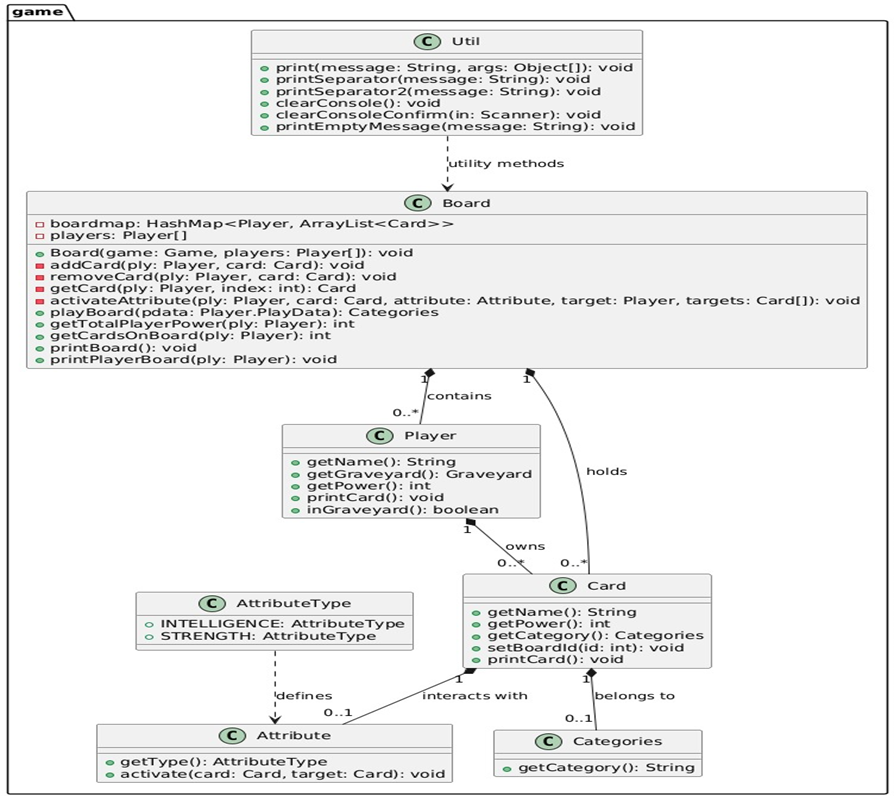
Below is the Utility class’ UML diagram  


Now onto the Player's section  
  


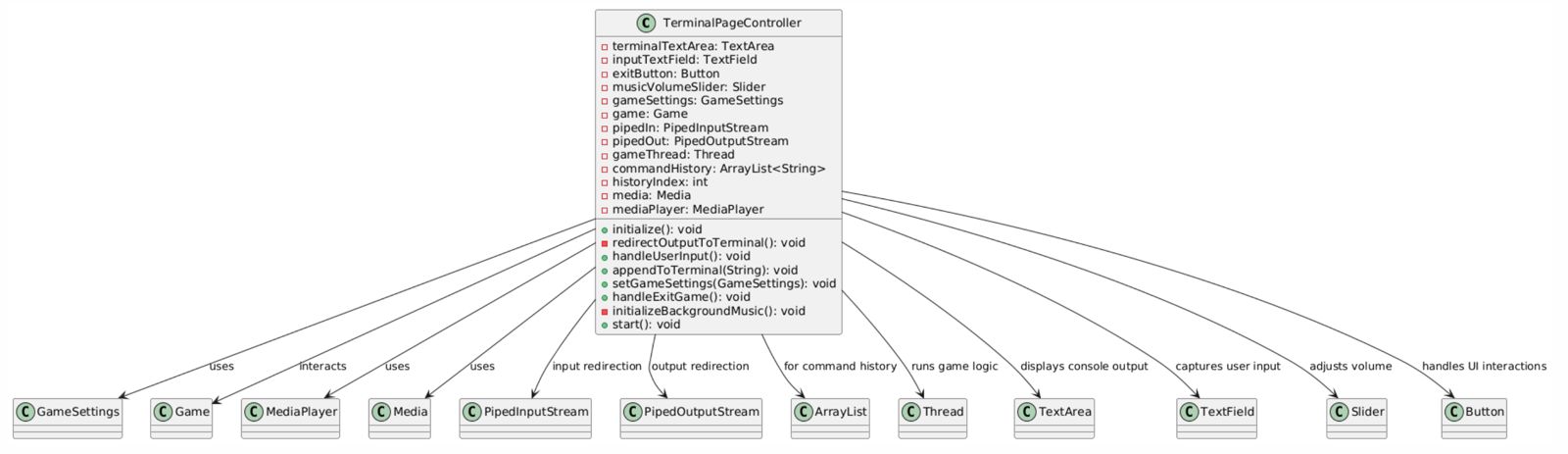
Players also have two Human categories and for the future scope of implementation we have Computer  
  


Utility UML for function  


Final UML



Front End UML DIAGRAMS

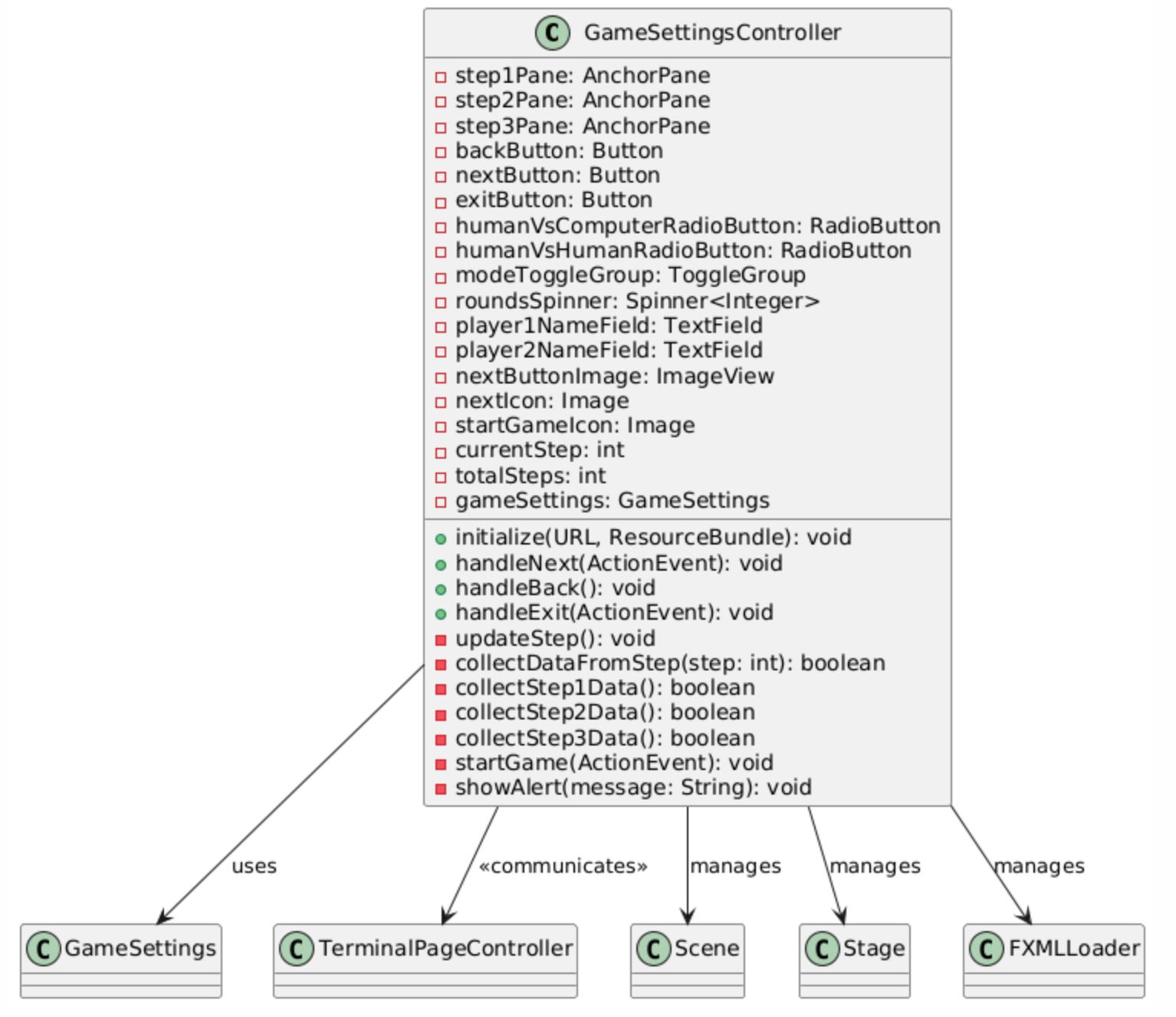


A diagram of a software

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated