Design Document

Version 1.1 - 2023.11.06
Created 2023.10.24

Project Name: FortCraft

Laith Alzoubi
Omar El Malak
Kaison Tran
Dan Duy Nguyen

Gitlab Repository: https://mcsscm.utm.utoronto.ca/csc207_20239/group_28

Project Identification

Why are you doing this project (i.e. what is the motivation?)

We want to make the existing Adventure Game from Assignment 2 more dynamic. We will do this via introducing turn-based combat with trolls, along with implementing new design features to improve accessibility for those with impaired vision or hearing.

How will it enhance or add to functionality that already exists?

We will enhance the functionality that already exists by adding new features using our implementations of the design patterns of State, Command, Memento, and Singleton. We will use these design patterns to help design solutions to the user stories in order to satisfy their respective acceptance criteria. This will improve the users' experience visually and will provide new game components that allow the game to be more engaging with a focus on troll battles. The following list is a mix of general accessibility and game features we plan on implementing, which are organized and expanded on in the User Stories and Acceptance Criteria sections…

- We will add a slider so the user can pick the font size they want.
- We will add a text-to-speech feature for anything in the scene.
- The user will be able to see the game information before loading a game.
- The user will be able to open the save/load window through commands.
- There will be a light and dark color theme.
- There will be 4 different kinds of trolls (Easy, Medium, Hard, Final).
- The player and the troll will have their HP displayed on the screen.
- There will be a settings window so that the user can choose font size, color theme, difficulty, and save/load games.
- The user will be able to attack the troll, and deal damage to it.
- When the user is in battle with a troll, they have the option to escape to the previous room.
- There will be weapons (to fight trolls), potions (to heal up), and keys (to enter locked rooms).
- There will be sound effects for trolls, attacking and escaping.

User Stories

Name	ID	Owner	Description	Implementation Details	Priority	Effort
Settings	1.1	Kaison	As a player, I would like to be able to access a settings menu so that I can make adjustments to accessibility features (font size, color theme, text to speech) and difficulty.	Add a Menu button which opens up a pop-up view. Add buttons to toggle features, change difficulty, select color theme, and save or load the game. The state of each button should be saved throughout the entirety of a fame, until it is changed by the user.	5	5
Difficulty	1.4	Dan	As a player, I would like to choose the level of difficulty so that I can play at my comfort level (Easy, Medium, Hard).	Add a drop-down button into the settings menu to select difficulty level. Add an “Easy”, “Medium”, and “Hard” setting that has its state saved throughout the entirety of the game until it is changed by the user.	4	2
Memento	1.6	Kaison	As a developer, I would like to access the attributes of saved adventureGames without needing to load the serialized binary file	Create a serializable caretaker which stores adventureGames in the form of mementos. The mementos will encode each adventure game into a string based on the base64 encoding scheme. The memento also stores the player attributes as the singleton design pattern does not allow it to store a copy of the player object. The caretaker will be responsible for saving and loading games.	2	3
Load Game Details	2.1	Dan	As a player, I would like to know the details of a save without having to load the game so that I don’t waste	Add the details of the saved game when selected, such as the last	3	4

			extra time. (It is quicker to view the save details instead of having to load each one until you get the one you want)	room name player was in, room number, inventory.		
Load/Save Command	2.2	Dan	As a player who has peripheral vision loss, I want to be able to save and load the game without needed to see the buttons of the edge of the screen	Add a command to save the players current progress, or load previously saved progress.	3	2
Battle	2.3	Laith	As a player, I would like to have all outcomes of the player and troll's attacks on each other to be displayed on the screen/	Add an invoker that will execute all commands depending on their respective buttons (attack, heal and escape)	4	3
Attack	2.4	Laith	As a player, I would like to be able to deal damage to the troll to take a step towards being able to defeat it.	Add a button which, upon clicking, either deals damage (removes HP) of the weapon used to the Troll or "whiffs" (misses the Troll).	4	3
Escape	2.5	Laith	As a strategic player, I would like the ability to escape from a troll (disengage from combat) so that I do not die and am able to plan a strategy.	Add a button to the top left of the screen which reverts the player to the previous room. The button will be set disabled to true until it is time for battle, in which case it will be set to false, indicating it is usable.	4	3
Potion	2.6	Omar	As a strategic player, I would like to be able to use potions so that I can tactically heal myself during a battle.	Add a Potion abstract class that inherits from AdventureObject class. The AdventureObject class should become abstract. Implement three distinct, insatiable subclasses of potions that are recognizable to the user no matter the game loaded in. For accessibility purposes, implement the visualization of potions through the GUI	2	3

				as well as via the command line, providing support for those with peripheral vision loss to be able to view (or hear with TTS) their potions without needing to see any buttons.		
Weapon	2.7	Omar	As a player, I would like to use weapons so that I can deal more damage to trolls.	Add a Weapon class that inherits from AdventureObject class. Implement an instance attribute dealing with damage range, and implement all backend/frontend methods required to support the Battle class. For accessibility purposes, implement the visualization of weapons through the GUI as well as via the command line, providing support for those with peripheral vision loss to be able to view (or hear with TTS) their weapons without needing to see any buttons.	3	3
Key	2.8	Omar	As a player, I would like to be able to use keys so that I can enter locked rooms.	Add a Key class that inherits from AdventureObject class. Ensure that keys still function as the subclass of AdventureObject designated to being able to open doors (via instanceof implementation). For accessibility purposes, implement the visualization of keys through the GUI as well as via the command line, providing support for those with peripheral vision	3	3

				loss to be able to view (or hear with TTS) their keys without needing to see any buttons.		
Troll	2.9	Laith	As a player, I would like to be able to come across trolls to make the game have more depth.	Add an abstract troll class with health and the ability to attack. Ensure the correct image files are being loaded upon troll encounters and there are subclasses of trolls available based on the current state of the game difficulty (“Easy”, “Medium”, “Hard”).	4	3
Audio Enhancements	2.10	Omar	As a visually impaired player, I would like to hear the troll sounds, healing sounds, and text to speech across the whole game to help provide cues for what is appearing on the screen.	Add sound effects that play when encountering monsters and healing, along with applying TTS to all text-related nodes (labels) in the view. Also, add a delay and noises to help indicate when healing is taking place for greater immersion.	2	3
Combat Audio	2.11	Dan	As a visually impaired player, I would like to hear combat audio to be able to recognize successful and failed attacks, along with awareness of when my player dies.	Add sound effects that play during combat that symbolize the player hitting or “whiffing” (missing) the Troll. Play a dying sound if the player dies or ender dragon sound if the troll dies.	2	3
Player Health	2.12	Omar	As a hearing impaired player, I would like to be able to see the health of my player to be able to make decisions on how I further engage in battle.	Display the health of the player on the bottom left corner of the screen using a combination of a visual and text display. Implement this using a button that simply acts as a “widget” displaying key health information. Ensure the correct image files are being loaded that	4	1

				can be adapted based on the current state of the player’s health.		
Player Singleton	2.13	Omar	As a game developer, I would like to be able to access the Player class globally so that I can better work with one single instance of the player across the whole AdventureGame.	Replace the existing Player class with a new one that implements the Singleton design pattern and update the relevant usages across other files. A health attribute must also be included to support the battles (User Story 2.3).	3	2
Adventure GameLoad er	2.14	Dan	As a game developer, I would like the extra features (trolls, weapons and potions) to be read and set to the correct attributes.	Add new methods to the existing AdventureGameLoader class to parse weapons, potions and trolls and set them to the correct attributes.	5	3
Final Boss Animation	2.15	Laith	As a game developer, I want the screen to animate in a specific way when the player enters the room with the final troll boss, so that the player experiences an immersive and dramatic buildup to the final encounter.	Implement a screen shake effect as the player enters the boss room, followed by a fade-to-black transition. This sets the stage for the dramatic appearance of the final troll boss, marked by a unique and elaborate animation. Accompanying this, a special background music track will enhance the climax.	2	3

CSC207 Project Phase 1 Acceptance Criteria

Name	ID	Acceptance Criteria
Settings	1.1	● Easily visible and accessible settings menu button at any point in the game ○ Should be visible above the scene and remains unchanged throughout the game ○ Should be a clickable “gear” icon ○ Should be effective in opening up the settings menu in the middle of the screen, replacing the current scene (similar to how it is done when showing instructions) ○ Clicking the gear again must close the settings menu and re-reveal the scene ● Settings menu includes all relevant buttons for accessibility and game settings ○ All accessibility and game alterations can be toggled through the settings menu ● Settings menu adapts to the current state of button clicks and color theme states ● The color themes option should provide two different color themes (light theme and dark theme). ● Color theme adjustments should apply immediately across the entire game (all menus, adventure item buttons, and the scene) and should be saved across sessions ○ This includes any transitions from one room to another, saving and loading new games, and any other user engagements (displaying instructions, picking up and dropping items, etc…) ● If text-to-speech is enabled, any text clicked on will be played out loud. ● Articulation stops and starts the next articulation whenever text changes ○ This can occur when the player changes rooms or click the instructions button
Difficulty	1.4	● Difficulty levels should include Easy, Medium, and Hard, each providing a distinct gameplay experience ○ Specifically, each level should reflect the trolls’ health and possible damage capability per attack on the player ● The difficulty options should be available in the settings menu in the form of three clickable buttons ○ One for “Easy”, one for “Medium”, and one for “Hard” ● Difficulty adjustments should apply immediately across the entire game (all trolls in all Rooms) and should be saved across sessions ○ This includes any transitions from one room to another, saving and loading new games, and any other user engagements (displaying instructions, picking up and dropping items, etc…) ● When the difficulty is changed, all trolls across the game (in ALL rooms) should have their healths reset to a health within the range of the current selected difficulty ○ i.e. A random number between 100 to 150 HP would be generated for a Hard troll, and this would apply to one troll in the game, the same process would occur for the next troll, and the next troll until all trolls have been adjusted to the new difficulty with health reset to a randomized value in the range ○ Additionally, all rooms must reflect the new corresponding Troll as their troll attribute

Memento	1.6	● Details like player inventory and room can be read from mementos ● Mementos are immutable once created ● Mementos are stored in an arraylist in the order they were created ● Memento correctly restores the player attributes when loading ● Caretaker is serialized upon closing the game and restored upon opening ● The CareTaker should be able to save the current state of the game using the save method. This involves creating a new Memento object that captures the current game state and adding it to the saves ArrayList. The game state includes player attributes, current room, and inventory. The acceptance criteria here are: ○ When a game is saved, a new Memento object must be created and added to the saves list. ○ The saved game state must accurately reflect the current state of the game. ● The CareTaker should be able to load a saved game state using the load method. This involves finding the correct Memento based on the save name, extracting the game state, and updating the current game to match the saved state. The criteria are ○ The load function must be able to find and load the correct saved game based on the provided name. ○ Player attributes, inventory, and current room must be accurately restored to their saved state. ● The replaceSave method allows updating an existing save with a new state. The criteria are: ○ The method must locate the correct save by name and replace it with the updated state. ○ The updated save should reflect the current game state at the time of replacement.
Load Game Details	2.1	● A list of saved games should display the file names of all valid, loadable game files (ending in .ser, from the /Games/Saved folder) ● Selection of a saved game should provide a preview of game details underneath the load button… ○ Player’s current room name ○ Player’s current room number ○ Player’s current inventory ● The list of saved games should be easily navigable ○ The player should be able to scroll through available loadable games ○ Saved games should be in order of most recent to least recent saves ● Overwriting of saved games should prompt a confirmation message. ○ Saving a file with the name of an already existing file will overwrite it
Load/Save Command	2.2	● Commands for saving and loading games should be intuitive and easily accessible. ○ Saving the game can occur if the user enters the command “save” ■ If a user inputs a name (“save <filename>.ser”), the file should be saved as “<filename>.ser” in the /Games/Saved/ folder ■ The saved file should, by default (no indication of a name with solely the command “save”), be saved as the current date in the form “YYYY.MM.DD.HH.SS.MS.ser” ○ Loading a new game can occur if the user enters “load <filename>.ser” ● A confirmation message should appear on successful save or load

		○ A failed message should appear on an unsuccessful save or load ● The save command should capture the current game state accurately ○ This includes the current room of the player, their inventory, current objects available in each room, and trolls in each room (along with their respective current healths) ● The load command should restore the game to the saved state accurately
Battle	2.3	● The Battle class must implement the command pattern to manage combat scenarios, essentially acting as the Invoker of commands. ● It should allow setting player and troll commands via setPlayerCommand() and setTrollCommand() methods. ● The Battle class instantiation should occur within the AdventureGameView class when the player encounters a troll. ● Commands corresponding to player actions (Attack, Escape, Heal) must be instantiated in the AdventureGameView class based on player input. ● The Battle class should have three distinct methods to execute combat commands: executeAttack(), executeEscape(), and executeHeal(). ○ executeAttack() should invoke both PlayerAttackCommand and TrollAttackCommand. ○ executeEscape() should invoke PlayerEscapeCommand. ○ executeHeal() should invoke PlayerHealCommand. ● These methods should be triggered by respective player actions (clicking Attack, Escape, or Potion buttons). ● The game must provide clear feedback to the player on the outcome of each command executed (Attack, Escape, Heal). ● The UI should reflect the current state of the battle, disabling irrelevant options (like keys, and input textbox) during combat. ● The UI should update to show the effects of the player's actions in real-time. ● Player commands should only be available during combat scenarios. ● Commands like taking or dropping keys should be disabled during battle.
Attack	2.4	● The attack button should be easily accessible during combat ○ Appears at the top of the screen, along with the other combat-related buttons ○ Disabled until user enters a room with a troll ● Clicking the attack button should deal damage to the troll ○ The damage dealt will be based on the weapon with the highest damage range in the player’s inventory ○ The damage value per attack should be selected randomly from the damage range of the strongest weapon ○ If the player has no weapons, they will be fighting with their fists and will have a default damage range. ● The result of the attack should be clearly indicated ○ Center of the screen will display how much damage player dealt to troll, how much damage troll dealt to player, as well as both of their HPs ● There must be a delay during the process of attacking in which a combat audio sound is played (detailed in the combat audio acceptance criteria), and damage is dealt to the troll ○ Trivially, only the troll the player is in battle with in the current room will be engaged in such battle and be able to deal and take damage ● Player will not be able to drop or take items after either their death or troll’s death. ● If the trolls kills the player, two things can happen:

		○ If the player has any saved games, the most recently saved will be loaded. ○ If the player has no saved games, a new game will be loaded.
Escape	2.5	● The escape button should be easily accessible during combat ○ Appears at the top of the screen, along with the other combat-related buttons ○ Disabled until user enters a room with a troll ● Clicking the escape button should force the player to the previous room ○ If the previous room only has “FORCED” passages, the player will not be able to revert back to the previous room ○ If the player is in the room with the final troll, they are again forced to stay in the current room. ○ Otherwise, the player will be sent back to the last room they were in ● Escape functionality should be available in most combat scenarios ○ The only time it won’t be available is when previous room only has “FORCED” passages and when facing the final troll ● The game should provide a visual and auditory indication that the player has escaped combat ○ The center of the screen will display the image of the new room the player enters ○ The room description should still be read out loud using text-to-speech ● All disabled buttons (non-combat related) should be re-enabled upon an escape from the troll
Potion	2.6	● Potion disappears upon use, cannot be used, dropped, or picked up again; it is removed from the game ● Potion does not increase HP over the maximum capacity of the player ○ The player’s health is set to 100 at the start of the game, and any healing potions cannot heal above 100 HP ● Player should not be able to drink a potion that is not in their inventory ● Potions should restore a specified amount of health ○ Each potion will have a set amount of health that will be added to the player’s HP upon use ● Potion usage should be intuitive and easily accessible ○ Using a potion will be done by clicking on it from the scrollable “potions” side menu of the screen (detailed in the next criteria point) ● All potions and potion images will be displayed on the right side of the window, along with weapons and keys ○ This should be scrollable ○ There should be a header that says “Potions” above all the clickable potions ○ Potions should NOT be able to be dropped into a room ■ The player should only be able to pick up or drink the potion from a room ● Thus, upon clicking the potion, it should disable, disappear, be used up, and then removed from the game entirely ■ The potion pictures should be accessible and have a caption of the name of the potion underneath ● A player can only use potions during a battle, in which they may only use the potion when it is their turn ○ They may not use the potion while it is the troll’s turn ○ Potions should be disabled at any non-battle moment in the game ● Any one of the three potions should be able to be added to any room (ALL games should have these same three types of Potion, as they allow any user of the game to pick it up

		and recognize the effects of each, which provides a sense of familiarity for an experienced user) ○ Bandages (gray background) should heal 25 HP ○ Med kit (green background) should heal 50 HP ○ Chug jug (gold background) should heal to max HP (100 HP) and not exceed it ● Player health should be updated upon Potion usage in the frontend/backend ● Add methods to access potions in Potion format and String format respectively for easier development for other stories ● Implement command “POTIONS” to view weapons ● Potions should still be visible from an “INVENTORY” command entry ● Should be light/dark mode responsive
Weapon	2.7	● Weapons should have clear damage ranges ○ Each weapon will have a range of damage that will deal a random amount of damage within it upon an attack dealt by the player holding the weapon ● The game should provide feedback on the effectiveness of weapon attacks (detailed in the Attack acceptance criteria) ● All weapons and weapon images will be displayed on the right side of the window, along with potions and keys ○ This should be scrollable ○ There should be a header that says “Arsenal” above all the clickable weapons ■ Upon clicking a weapon picture, it should move between the “Arsenal” tab and the objects in room tab ● This movement between room and inventory should be reflected in the back end as well ● Weapons can also be picked up and dropped using the commands “TAKE <name>” and “DROP <name>” respectively” ■ The weapon pictures should be accessible and have a caption of the name of the weapon underneath ○ The number of weapons the player has should be clearly displayed (“Arsenal (x)”) ● A player can only use weapons during a battle ● Add methods to access weapons in Weapon format and String format respectively for easier development for other stories ● Implement command “WEAPONS” to view weapons ● Weapons should still be visible from an “INVENTORY” command entry ● Should be light/dark mode responsive ● Like Fortnite, the weapon rarity backgrounds should be adjusted based on weapon strength ○ Determined by the upper bound of the range of the weapon ● A game should dictate the weapons in the game and their images ● Should be able to dictate which weapon is the strongest from a player’s inventory ● A weapon should be able to return a random damage calculated within its range for a particular attack
Key	2.8	● Keys should unlock specified rooms within the game ○ A key should not be able to unlock a room that is not blocked by such key ● The game should provide feedback on successful or unsuccessful use of keys ● Keys should be reusable even if already used (unless dropped) ● All keys and key images will be displayed on the right side of the window, along with potions and weapons

		○ This should be scrollable ○ There should be a header that says “Keys” above all the clickable keys ■ Upon clicking a key picture, it should move between the “Keys” tab and the objects in room tab ● This movement between room and inventory should be reflected in the back end as well ● Keys can also be picked up and dropped using the commands “TAKE <name>” and “DROP <name>” respectively” ■ The key pictures should be accessible and have a caption of the name of the key underneath ○ The number of keys the player has should be clearly displayed (“Keys (x)”) ● Keys are disabled during battle and have no use ● Add methods to access keys in Key format and String format respectively for easier development for other stories ● Implement command “KEYS” to view weapons ● Keys should still be visible from an “INVENTORY” command entry ● Should be light/dark mode responsive ● A game should dictate the keys in the game and their images
Troll	2.9	● As soon as the player enters a room with a troll, troll image will be added to the center of the screen (cell (1,1)) ○ If the troll is being played against on the easy difficulty, a less-menacing looking troll visual will be played, getting increasingly menacing the more difficult the troll is ● Troll disappears after losing all of its health ○ Troll will die allowing the player to have access to the room ● Troll health does not restore after escaping from combat and re-engaging ○ Every troll dealt damage will have the same health throughout the game, unless the user changes the difficulty of the game ■ In which case, ALL trolls’ health in ALL rooms will be reset to the new health determined by the user’s selection of difficulty ● The type of troll will also be changed, altering the image of the troll that appears upon entering a room with a troll ● Trolls should have specified health and attack ranges depending on current selected difficulty ○ Trolls will deal a random amount of damage within the range per turn ● The troll’s health should appear beneath the picture of the troll. ○ This number should be visible and adjusted based on the damage taken by the troll from the player’s attacks throughout a battle ● The player’s text field should be disabled during an encounter with a troll ● All buttons will be disabled during combat except for the 3 related to the battle (Attack, Escape and Potions) as well as the menu button. ● All disabled buttons should be re-enabled upon the death of the troll ● Once a troll dies, it is gone and will not be encountered again.
Audio Enhancements	2.10	● Monster audio should play upon encountering trolls ● Monster audio should not overlap with other critical game sounds ○ Other articulation will stop before monster audio is played ● The monster introduction sound played will differ depending on the difficulty of troll being encountered ○ If the troll is being played against on the easy difficulty, a zombie sound will play

		○ If the troll is being played against on the medium difficulty, a creeper sound will play ○ If the troll is being played against on the hard difficulty, a herobrine sound will play ● Add delays for greater immersiveness when healing, accompany this with a visual and audio indicator of healing ● Implement text-to-speech across the entire view
Combat Audio	2.11	● Combat audio should play during attack actions ● Combat audio should not overlap with other critical game sounds ○ Other articulation will stop before combat audio is played ● Successful hits should have a distinct sound effect, and there should be enough time during the process of an attack ○ After the attack button is clicked by the player, there should be a delay, allowing for the relevant audio to play of the attack result ○ The process of the troll attack should take the same amount of time, allowing for the relevant audio to play of the attack result ○ The audio indication for a hit will differ from the audio of a missed attack, to help indicate for those who are visually impaired to allow them to better understand the combat scenario and make informed battle decisions
Player Health	2.12	● Player health is displayed at all times for the user to see ○ This should be displayed at the bottom right of the screen, right above the text field for inputs ● Player HP is updated after each turn ○ Taking a potion will increase the health immediately ○ Damage dealt from troll will decrease it and must be indicated immediately during the process of a troll's attack ● The visual and text aspects of the button should be clear and evident in helping aid a hearing impaired user have a complete visual and numerical view of their health at any point in the game
Player Singleton	2.13	● The Player class should be implemented using the Singleton design pattern ○ The single Player object should be able to be accessed from a static context across the codebase where needed (and will be needed when other user stories are being implemented that involve the Player object) ● The Player class should include the health attribute and its accessor/mutator methods found in the UML diagram for the Singleton design pattern ● Usages of Player from other files (specifically AdventureGameView as AdventureGame will keep the attribute for easy access) should reflect the changes made via the Singleton implementation
Adventure GameLoad er	2.14	● The class should parse all new text files that are in the same format as objects.txt that are needed because of the new features. ○ weapons.txt for weapons ○ potions.txt for potions ○ trolls.txt for trolls ● Each new feature will have a method: ○ parseWeapons() for reading weapons and adding them into the objectsInRoom attribute. ○ parsePotions() for reading potions and adding them into the objectsInRoom

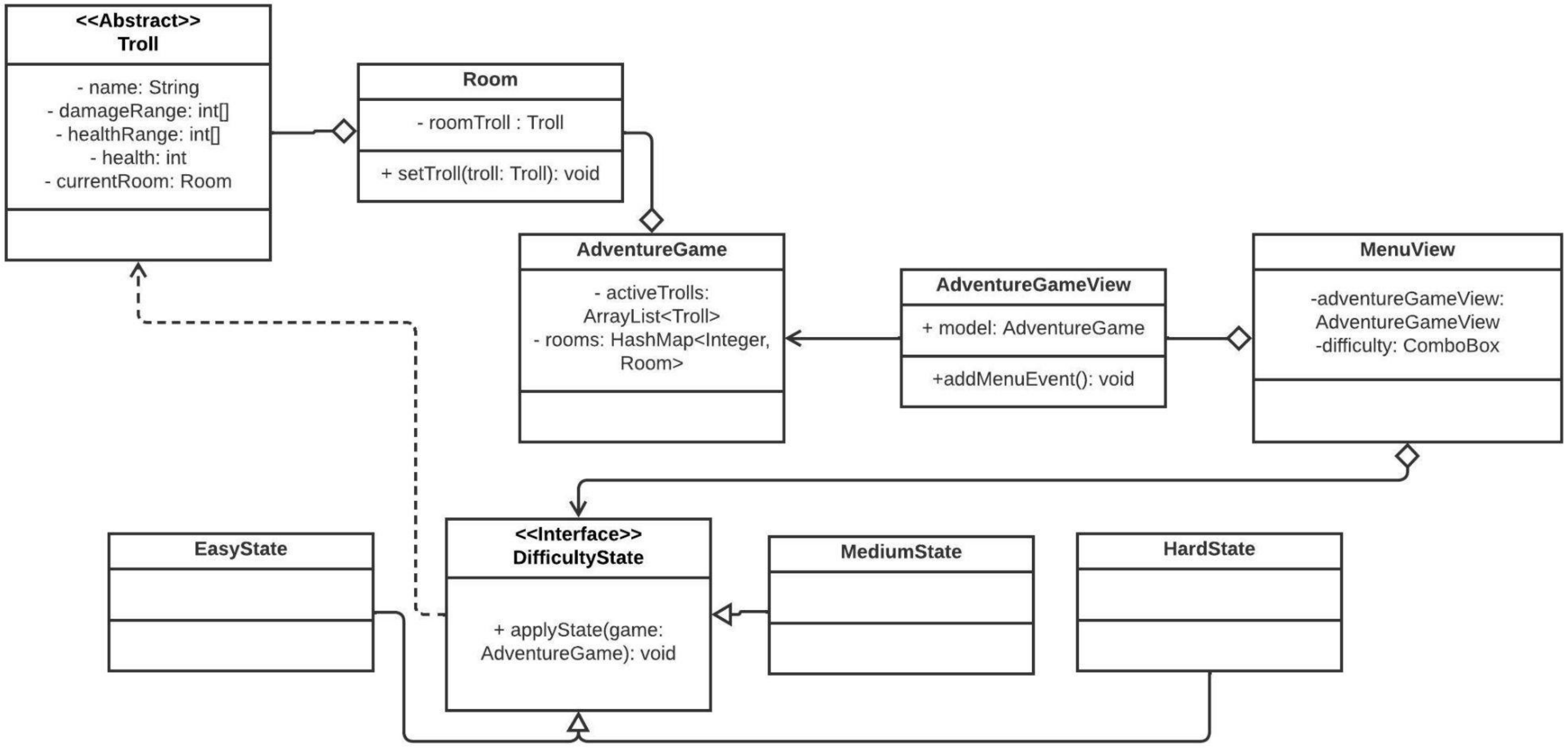
		attribute. ○ parseTrolls() for reading trolls and setting them into the roomTroll attribute.
Final Boss Animation	2.15	● The screen shake should be clearly noticeable but not overwhelming, adding to the drama without causing discomfort and should fit well with the stomping sounds. ● The fade to black and back should be smooth, without any visual glitches. ● The final troll boss's entrance animation should be visually striking and distinct, making the encounter feel special and different from previous battles. ● The background music should start precisely as the boss appears, enhancing the overall impact of the scene.

Software Design

DESIGN PATTERN #1: STATE DESIGN PATTERN

Overview: This pattern will be used to allow for state changes to be made globally and in the relevant areas of the codebase upon changes to difficulty and color theme in the settings of the game

UML:



<<Abstract>>
Troll
- name: String
- damageRange: int[]
- healthRange: int[]
- health: int
- currentRoom: Room
Room
- roomTroll : Troll
+ setTroll(troll: Troll): void
AdventureGame
- activeTrolls: ArrayList<Troll>
- rooms: HashMap<Integer, Room>
AdventureGameView
+ model: AdventureGame
+addMenuEvent(): void
MenuView
-adventureGameView: AdventureGameView
-difficulty: ComboBox
EasyState
<<Interface>>
DifficultyState
+ applyState(game: AdventureGame): void
MediumState
HardState

https://lucid.app/lucidchart/cd00591d-3979-42ed-bf90-3fa1a0763866/edit?viewport_loc=99%2C48%2C2937%2C1670%2CHWEp-vi-RSFO&invitationId=inv_2f2bb563-7d2a-4b15-b014-cabe8d24e647

Implementation Details: The UML diagram outlines these main components:

- The DifficultyState (Interface), which holds the method used by all implementing classes (EasyState, MediumState, and HardState) to execute a state change in all relevant areas
- The AdventureGame, which contains the ArrayList of active trolls in the game and a HashMap carrying the mapping from room number to the Room object, using these two variables to update the health and damage capabilities of all trolls in the current game
- The Room, which holds a troll (or null if the room is not blocked by a troll) and the method enabling the ability to assign a troll to a room
- The Troll (Abstract Class), which contains all relevant instance attributes of a troll of any difficulty (EasyTroll, MediumTroll, HardTroll, and FinalTroll)
- The Settings, which contains an AdventureGameView which contains the dropdown menu to select the difficulty state.

DifficultyState Implementation Details:

The DifficultyState Interface dictates the ability for an EasyState, MediumState, and HardState to be able to effect change in the game and gives each state a meaning: Upon clicking one of the difficulty buttons (easyDifficultyButton, mediumDifficultyButton, or hardDifficultyButton) in the Settings, the button handler will create a new State object (EasyState, MediumState, or HardState) and will have the AdventureGame passed in upon calling the state object's applyState() method. Two things should happen here…

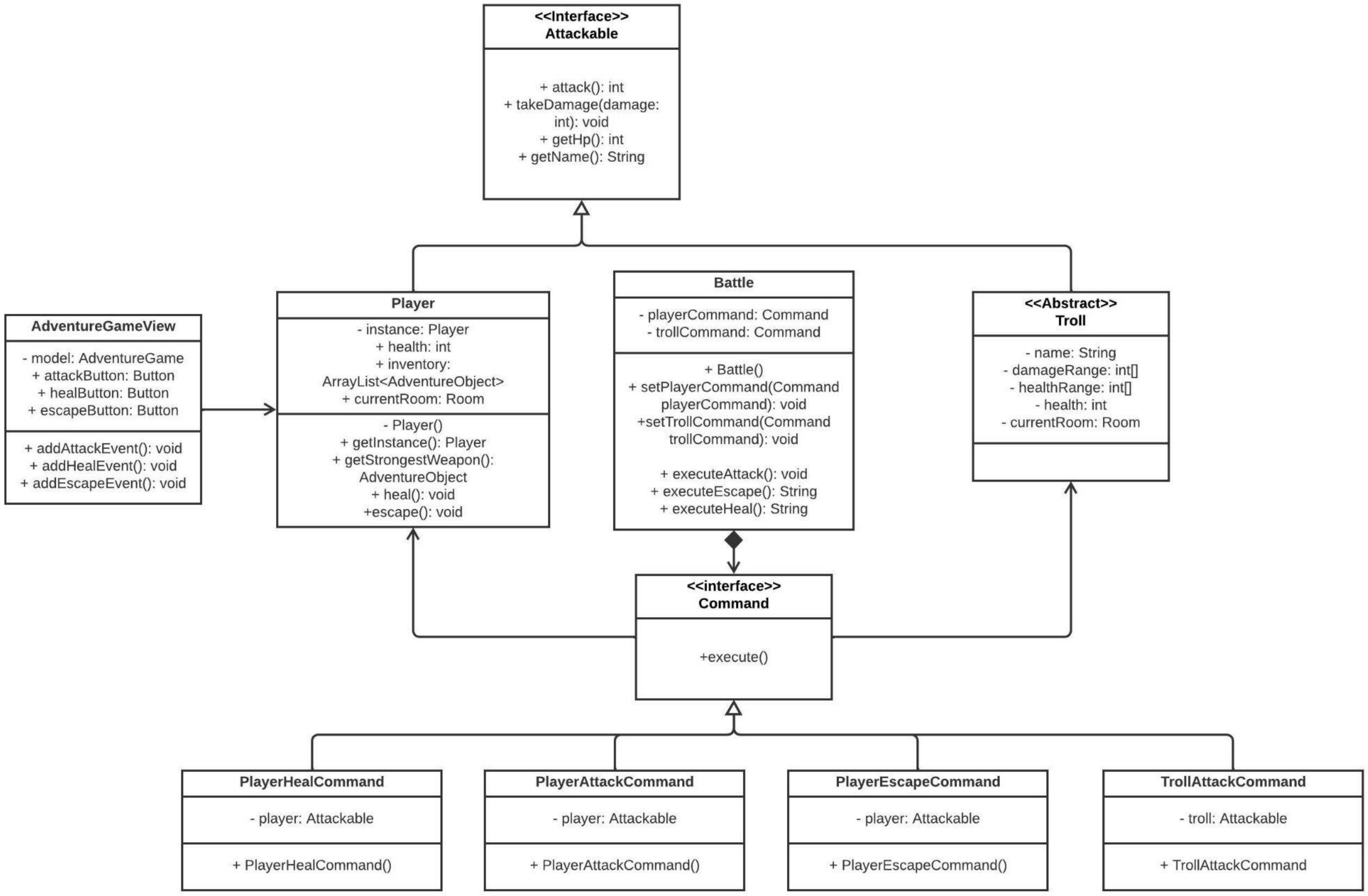
1. Each Troll (except the FinalTroll) in the ArrayList of activeTrolls in the AdventureGame object will be replaced with a new Troll that has the same room number, but a different difficulty (EasyTroll, MediumTroll, or HardTroll respectively) with attacks and health randomly ranging within a predefined domain that correspond to the difficulty of the troll (except the FinalTroll)
2. Each Room will have their "troll" instance attribute replaced with the new corresponding Troll object, accessible through the rooms hashmap in the AdventureGame object

This allows for the user to play the game with different difficulties, making the game more dynamic in terms of the combat experience.

DESIGN PATTERN #2: COMMAND DESIGN PATTERN

Overview: This pattern will be used to implement the Escape, Heal and Attack functionalities for the Player, and Attack for Trolls. In particular, requests for a Player to Escape, Attack and Heal and for a Troll to attack, will be defined by a *Command* interface. These Commands are then passed to an invoker object, in this case called Battle, which then handles the Command execution.

UML:



<<Interface>>
Attackable
+ attack(): int
+ takeDamage(damage: int): void
+ getHp(): int
+ getName(): String
AdventureGameView
- model: AdventureGame
+ attackButton: Button
+ healButton: Button
+ escapeButton: Button
+ addAttackEvent(): void
+ addHealEvent(): void
+ addEscapeEvent(): void
Player
- instance: Player
+ health: int
+ inventory: ArrayList<AdventureObject>
+ currentRoom: Room
- Player()
+ getInstance(): Player
+ getStrongestWeapon(): AdventureObject
+ heal(): void
+escape(): void
Battle
- playerCommand: Command
- trollCommand: Command
+ Battle()
+ setPlayerCommand(Command playerCommand): void
+setTrollCommand(Command trollCommand): void
+ executeAttack(): void
+ executeEscape(): String
+ executeHeal(): String
<<Abstract>>
Troll
- name: String
- damageRange: int[]
- healthRange: int[]
- health: int
- currentRoom: Room
<<interface>>
Command
+execute()
PlayerHealCommand
- player: Attackable
+ PlayerHealCommand()
PlayerAttackCommand
- player: Attackable
+ PlayerAttackCommand()
PlayerEscapeCommand
- player: Attackable
+ PlayerEscapeCommand()
TrollAttackCommand
- troll: Attackable
+ TrollAttackCommand

https://lucid.app/lucidchart/04c4387a-bbfb-4804-bfc3-848e79588755/edit?viewport_loc=-618%2C477%2C5328%2C2549%2CHWEp-vi-RSFO&invitationId=inv_ef8ce594-325c-4090-962b-d51e0c0ac399

Implementation Details: The UML diagram outlines these main components:

- The Command (Interface), which holds the method used by all implementing classes (PlayerEscapeCommand, PlayerAttackCommand, PlayerHealCommand, TrollAttackCommand) to implement the respective command's function (execute()).
- The Battle class acts as the "invoker" object by handling given Player/Troll Commands by setting them as the instance attributes of the object, then executing the respective command.
- An Attackable interface implemented by Player and Troll classes, providing attack(), getHp() and getName().
- The Player class which acts as a "Receiver", will contain the specific implementation of the healing, attacking and escaping methods.
- The Troll abstract class, which also acts as a "Receiver", will have 4 sub-classes (EasyTroll, MediumTroll, HardTroll, FinalTroll). This class will also contain specific implementation of the attack() method so it can be called in the Battle methods.
- The AdventureObject abstract class is the superclass of all objects in the game: Weapon, Key, and Potion.
- The Potion abstract class, which is the super class of all 3 types of potions: Bandage, MedKit, ChugJug.

Battle Class Implementation Details:

- The Battle class implements the command pattern to manage combat by assigning player and troll commands through setPlayerCommand() and setTrollCommand() methods, linking them to their respective attributes, then executing those commands through one of its 3 methods.
- The instantiation of a Battle object happens within the AdventureGameView class, which happens when the player enters a room containing a troll. Depending on the button clicked, a Command of that type is instantiated.
- Whenever one of the 3 combat related buttons are clicked (Escape Button to escape, Attack button to attack, potion to heal), the corresponding PlayerCommand (PlayerAttackCommand, PlayerEscapeCommand, PlayerHealCommand) and the TrollAttackCommad (for the Attack button only) is created in AdventureGameView class and set as the current command as the Battle class attributes.
    - The executeAttack() method is called when the Attack Button is clicked, and it invokes the attack commands (PlayerAttackCommand and TrollAttackCommand) for both the player and the troll by calling the commands' execute() method.
    - The executeEscape() method is called when the Escape Button is clicked, and it invokes the escape command (PlayerEscapeCommand) for the player by calling the command's execute() method.
    - The executeHeal() method is called when a Potion is clicked, and it invokes the heal command (PlayerHealCommand) for the player by calling the command's execute() method.

Player Class Implementation Details:

- The Player class has three command methods: attack(), escape(), and heal(). The attack() method deals damage based on the strongest weapon and is executed alongside the troll's attack. The escape() allows the player to revert to the previous room unless it's blocked by "FORCED" passages, and heal() restores HP up to the maximum, with potions being single-use. These are executed within their respective Command objects.

Troll Class Implementation Details:

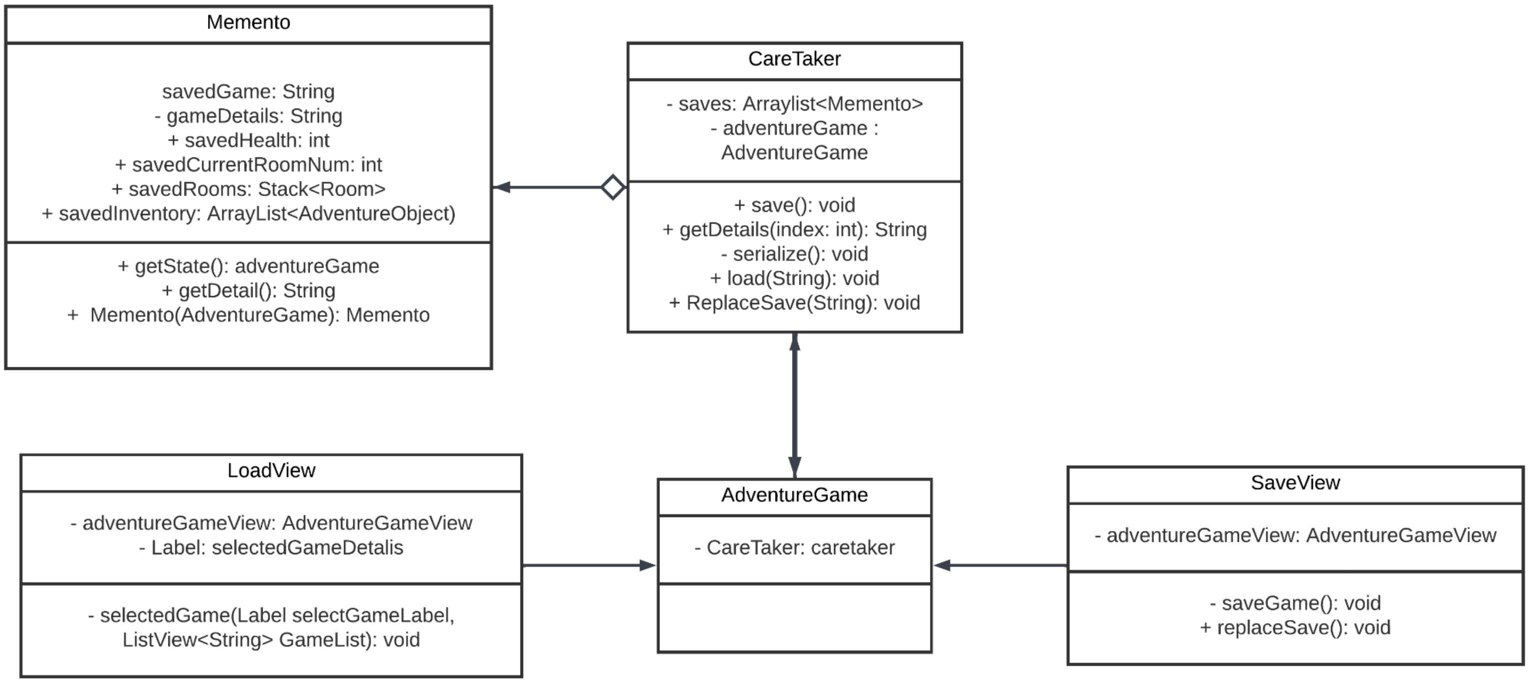
- The attack() method will be implemented so it deals damage to the player. Each troll class will have a set damage range, so that the damage amount is set by selecting a random number within that range. This method will then be called in the Battle object using the executeAttack() method along with the Player's attack() method.

In summary, the Battle class will have composition arrows to the Command interface, representing that it holds and manages Player and Troll commands. The concrete command classes will have implementation arrows to the Command interface and association arrows to their respective Player or Troll class. executeAttack will have association arrows to Player and Troll, and all the others will have association to Player only.

DESIGN PATTERN #3: MEMENTO DESIGN PATTERN

Overview: This pattern will be used to implement Save and Load functionality

UML:


Memento
savedGame: String
- gameDetails: String
+ savedHealth: int
+ savedCurrentRoomNum: int
+ savedRooms: Stack<Room>
+ savedInventory: ArrayList<AdventureObject)
+ getState(): adventureGame
+ getDetail(): String
+ Memento(AdventureGame): Memento
CareTaker
- saves: Arraylist<Memento>
- adventureGame : AdventureGame
+ save(): void
+ getDetails(index: int): String
- serialize(): void
+ load(String): void
+ ReplaceSave(String): void
LoadView
- adventureGameView: AdventureGameView
- Label: selectedGameDetalis
- selectedGame(Label selectGameLabel, ListView<String> GameList): void
AdventureGame
- CareTaker: caretaker
SaveView
- adventureGameView: AdventureGameView
- saveGame(): void
+ replaceSave(): void

The LoadView interacts with the CareTaker to obtain the necessary information to display saved games to the player. It lists the names and details of the available game states, each represented by a Memento. When the player chooses to load a game, the LoadView communicates the selected index to the CareTaker. The CareTaker then retrieves the corresponding Memento from its ArrayList. The memento then uses it's getState method to convert the string representation of the saved adventure game back into an adventureGame object. The caretaker then replaces the view's current model with the loaded model. The caretaker is also responsible for restoring the saved attributes of the player, which are also saved in the memento.

When a memento is created, the adventure game is converted into a string based on the base64 encoding scheme. This string is stored by the memento as a string attribute which can later be reconverted into an adventure game. Because mementos are stored in an indexed array, developers can easily retrieve the most recent save. The memento also stores the details of the saved game like current room and inventory. This prevents the need to load a game in order to see the details.

The serialization of the CareTaker class enables the state of the game saves to persist beyond the life of the program. When the game is closed, the CareTaker, along with its collection of Mementos, is serialized to a file. Upon restarting the game, the serialization file is read, and a new CareTaker is populated with the previously saved Mementos. This allows the game to restore the exact point where the player left off, including all the saved states available at that time. In the case that no caretaker serialized file exists when the game is created, the adventure game view will use an empty caretaker.

This mechanism ensures that the state of the game can be saved at any point and later restored, providing a robust undo/redo functionality and a way to manage game saves without violating the encapsulation of the game's internal state.

CareTaker to Memento:

- The CareTaker class is responsible for creating and managing Memento objects. This relationship can be represented as either an association or composition, depending on whether the Memento objects are considered to have their lifecycle tied to the CareTaker.

CareTaker to AdventureGame:

- The Caretaker interacts with the AdventureGame to save its current state into a Memento and to restore its state from a Memento. This is likely an association relationship if CareTaker maintains a reference to AdventureGame. It's a dependency if the interaction is only during the save/load processes.

SaveView to CareTaker:

- The SaveView class depends on the CareTaker to initiate the saving of the game's state. This is represented by a dependency relationship, as SaveView utilizes CareTaker for the save functionality.

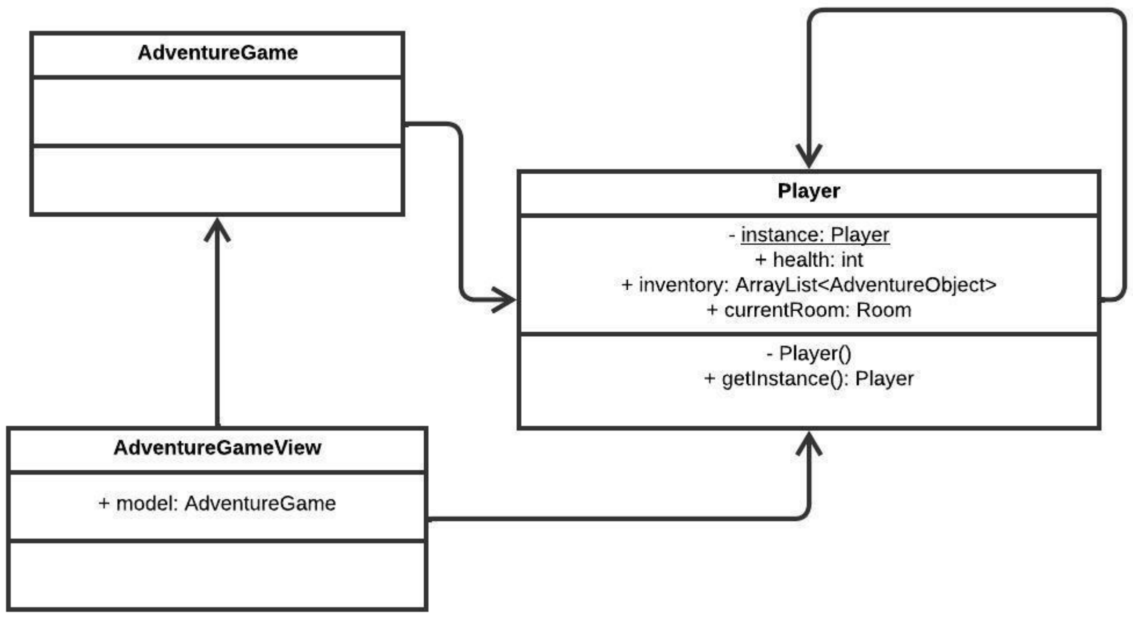
LoadView to CareTaker:

- The LoadView class also has a dependency relationship with the CareTaker. It relies on the CareTaker to load a previously saved game state.

<u>DESIGN PATTERN #4</u>: SINGLETON DESIGN PATTERN

Overview: This pattern will be used to standardize the number of instances of settings and players to strictly one respectively, allowing for global access to such instance across the codebase

UML:



https://lucid.app/lucidchart/32800f9f-0461-4116-a7a5-7ec78dcdb4c1/edit?viewport_loc=-207%2C494%2C1802%2C862%2CHWEp-vi-RSFO&invitationId=inv_d88377a3-3408-486c-8f0c-3f12b6516af6

Implementation Details: The UML diagram outlines these main components:

- The Player, which carries the single instance of itself
- The AdventureGame, which uses the Player singleton.
- The AdventureGameView, which also uses the Player singleton.

The Player class holds a private static "instance" variable to hold the single instance of itself. The Player's static getInstance() method checks to see if there exists a Player instance in the player instance variable of the class. If it does not, then it will create the single instance, and every time the getInstance() method is called from any class, it will return the single instance held in the Player class static "instance" attribute. This solidifies that there will only be exactly one instance of Player at all times. The constructor is private to ensure that new excess instances cannot be made from outside the Player class.

The AdventureGame class has a setupGame() method that calls Player.getInstance() to collect the single instance of Player (at this point, it will not have been instantiated yet, so the getInstance() method will create the one and only Player) and allow its player instance attribute to reference this Player.

The AdventureGameView contains an instance of AdventureGame to pull information from to display, along with a referenced Stage object, which is where all components to be viewed are rendered.

This mechanism ensures that the Player is only created once, and this single object is the only object of its respective class (Player) created throughout the entirety of a game. This enables us to work in a more organized manner as developers to streamline the way we implement the combat experience with Trolls.

Summary:

- The Player class has a self-association arrow to indicate the Singleton pattern through the getInstance() method.
- Since AdventureGameView and AdventureGame depend on each other, they have a 2-way dependency arrow.
- Both AdventureGameView and AdventureGame have an association with the Player singleton class, so they are a 1-way arrow from each one of them to the Player class.