GROUP 41 CSC 207 DESIGN DOCUMENT

Created: 2023/11/05

Project Name: *Survive the Night!*

-Jonathan Chun Hei Sin

-Shah Mir Zafar Ahmad

-Saurabh Nair

-Kenson Jia Jie Guo

Gitlab Repository: https://markus.teach.cs.toronto.edu/git/utm-2023-09/csc207h5/group_2021

SECTION 1: PROJECT IDENTIFICATION

- ***Why are you doing this project (i.e. what is the motivation?)***
  - We are undertaking this project to create an RPG adventure survival game with a strong focus on accessibility features, driven by our motivation to make gaming more inclusive for users with visual impairments.
  -
- ***How will it enhance or add to functionality that already exists?***
  - By incorporating optical-impairment friendly visual and audible sensory features, we aim to transcend the range through which the user can interact with the game.

SECTION 2: USER STORIES

NAME	*ID*	*OWNER*	*DESCRIPTION*	*ACCEPTANCE CRITERIA*	*IMPLEMENT DETAILISATION*	*PRIORITY*	*EFFORT*
TrollClass	1.1	Kenson	As an adventure player, I want the process of fighting the troll player to be more enjoyable.	Given that I am an adventure player, when I engage in a battle with the troll player, the battle should include the following elements: ● The troll player should have health (HP). ● The player should be able to attack the troll player. The player should be able to see the troll player's HP.	Create a Troll class as an enemy, assign HP attributes to the Troll, and implement the combat mechanism, allowing the player to interact with the Troll.	2	3
Collect	4.1	Kenson	I want to collect all the game pieces for the satisfaction it brings so I can feel accomplished.	Given that I am a collection player, when I complete the game with all objects collected, I want to receive an achievement.	1. Implement a list variable containing all the objects, check at the end game if the player contains all the objects. 2. At the end of the game, check all the rooms to see if there is any object remaining in the room.	1	2
Random	2.1	Kenson	As a rouge player, I want the game to introduce more randomness so I can have more interest on playing	Given that I am a rogue player, I want the game to include random events to increase unpredictability and excitement.	Random events could include the introduction of random paths, random objects in rooms. Set random int (1-100), And random.randint(1-100). if the result is less than the random attribute of the event, it will happen.	2	3

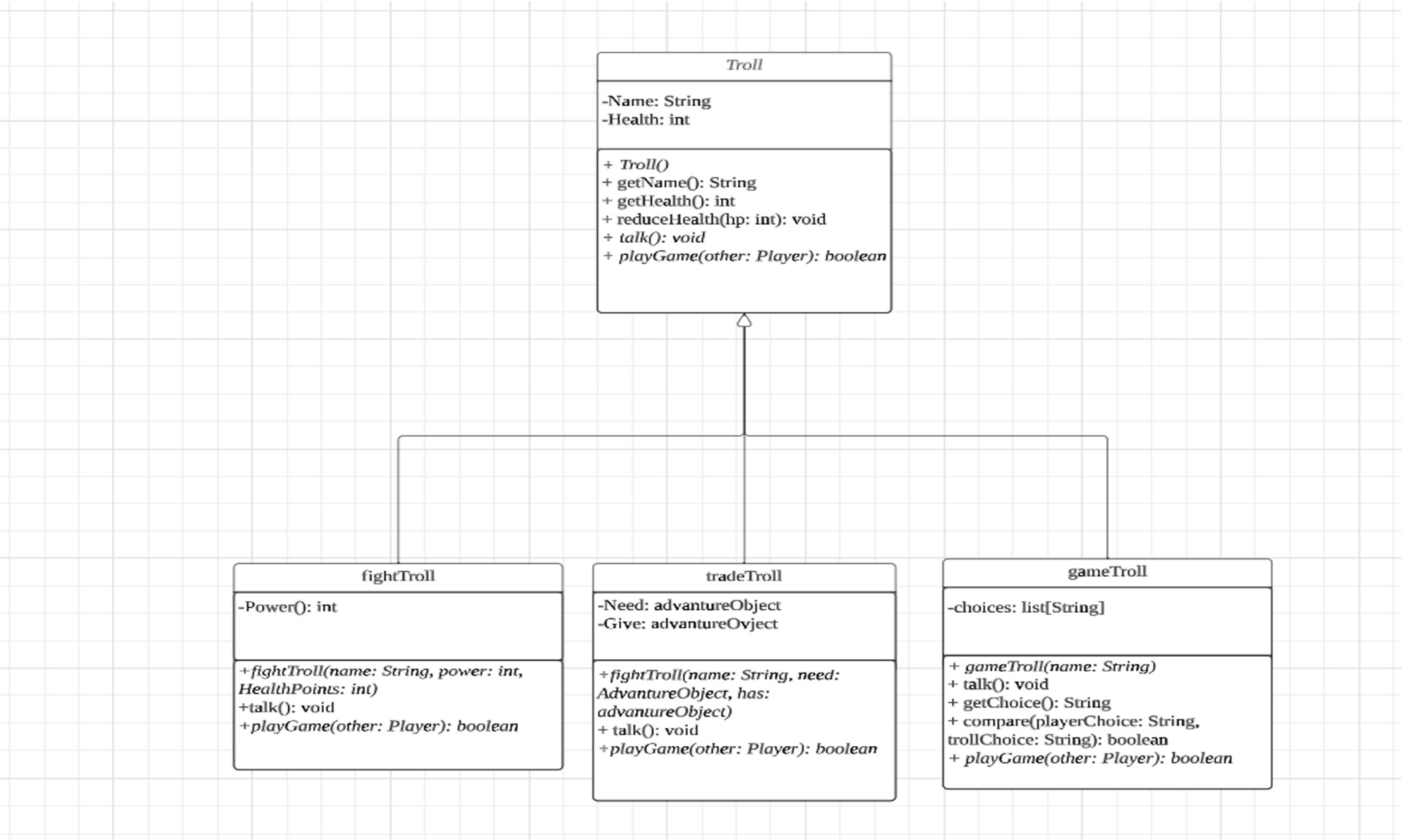
Rock, Paper, Scissor	3.1	Kenson	As a user, I want to be able to play rock, paper, scissor during the game	Users are able to choose one of the choices and the Troll will choose one and decide who is going to win, first one that reach three points win the game	Troll will randomly choose from Rock, Paper, Scissor and decide who win, whoever reach three points will win the game.	1	2
State DP	1.2	Shah	As a user, I want to be able to choose the accessibility level of the game, whether I need visual/audible assistance or not, and the GUI elements should adapt accordingly.	Given the user wants to pick a level of accessibility for the game, a select menu pops up and presents options. Given the user chooses an accessibility option (blind, color blind, other visual impairments) GUI elements must be equipped with a accessibility features to aid the user. Given the user chooses the regular option, the GUI elements should only support their basic functionality.	Implement a state design pattern: Turn adventureGameView into a Parent class. Implement child classes which inherit AdventureGameView's basic functionality and offer more features. The child classes wil catter to different accessibility levels of the game (E.g: Color-blind mode, Black and White mode, Regular mode). Also Implement a SelectGameView class, a small pop-up screen for the user to pick their accessibility options.	1	2
Visual Spacing	2.2	Shah	As a user with any kind of visual impairment, I want the game screen to not be congested with clusters of different GUI features at any given part of the screen.	GUI elements must be sufficiently spaced. Only GUI elements of similar roles/purposes should be grouped together. User must have whitespace between elements to hover the cursor without unnecessarily employing any game features.	Implement a sufficiently large grid pane, with enough panes to house groups of GUI elements with similar roles and have empty panes where the user then let their cursor hover without initiating any game feature involuntarily.	1	3
Hover Button Feature	3.2	Shah	As a visually impaired (blind) person, I want to be able to navigate/interact with the GUI elements on the screen without having to be able to see them.	Given I am a non-sighted user, if I hover the mouse cursor over a GUI element for over 3 seconds, audio plays highlighting the button name. If it's the first-time hovering over the button, the button description will also play. If it is not the first time, the button name is sufficient.	Create a Boolean attribute which keeps track of whether it is the first time or not the user is hovering over the button. Use JavaFX library tools to detect cursor over objects, subsequently begin playing the appropriate audio.	2	3
Collections	2.3	Shah	As a player, I want to be able to see all the items and buffs my player has collected, and the items to be collected in a setting within the game.	Given I am a player entering a new setting in the game and there are objects and buffs in the room. If/When I click on the object it should automatically go to the correct inventory space (either the object inventory or buffs inventory).	Create a new String [] attribute for Player class to hold their active buffs. Create four scroll pane objects that can house multiple objects and buffs, respectively. 2 will be on the left showing the objects and buffs in a new game setting, 2 will be on the right showing the player's inventory.	1	2
Audio Playback	4.2	Shah	As a focused, visually impaired player I want to be able to carefully listen to the audio playback of the text-to-	Given that I am visually impaired and cannot keep up with the speed of the audio files, I should be able to adjust the speed of the playbacks from a range playback speed option.	Import the necessary Java libraries to appropriately interact with Audio input files and manipulate the audio input. Create a game attribute which keeps track of the user's preferred playback speed.	3	3

			speech and other audio features of the game.	E.g.: x1.0, x0.75, 0.5.	This attribute will be used as an input to the methods which will play audio files, so that they play at the chosen speed.		
Survival element	1.5	Jonathan	As a survival game player, I want the game has more survival element so it will be more challenging and suits the lore of story	Given that I am a survival player, when I walk in the game aimlessly, I will die in [20] turns.	1. Player has hunger stat [100], which will decrease when moving room to room 2. Player has infection stat [0->100], which will increase when defeat to troll 3. Player has an energy stat [100], which will decrease when moving room to room. [resting] will be an option in the available command.	2	2
Different endings	2.5	Jonathan	As a non-linear game player, I want the game to have different endings so it will be a more interesting story-wise	Given that I am a non-linear game player, when I am near the ending of the game, I will have at least 5 ways to pass the game.	1. There will be at least 4-5 room that can achieve ending	2	3
Randomness in enemy stat	3.5	Jonathan	As a RPG player, I want the game to have different battle so it won't be boring in repeating battle	Given that I am a RPG player, when I encounter a battle, I will have a different battle every time.	1. We will encounter [1d10] infected(troll) each time, the difficulty of the battle depends on this	3	3
Dynamic room stat	4.5	Jonathan	As an RPG player, I want the stat of each room will change in runtime so it will be more dynamic	Given that I am an RPG game player, when I kill an infected person in the game, it will be removed in the destination.	1. After battling the infected(troll) in the room, the troll will disappear the next time we arrive. 2. Every time we arrive at a new room, it may add a new option/destination to the room (by randomness), which will not stay the same as the room.txt forever. 3. Every time we arrive at a new room; it may add forced encounter to infected(troll)	2	2
Visual element	3.4	Jonathan	As a Dyslexia/reading disorder person, I want to see more visual element for stat so I can read the info	Given that I am a Dyslexia/reading disorder person, when I look at my health, I can know my health approximation immediately without reading.	1. Every piece of info should have a visual presentation related, and text. 2. Use energy bar for stats	3	3
Audio battle for enemy	4.4	Jonathan	As a low vision person, I want to have battles based on sound so I can battle easily without reading.	As a low vision person, when I start a battle, I can finish a battle quickly without reading.	1.Create an Audio_battle class 2.Next_turn() method for every turn 3.Every turn, the zombie will come close/go away from the player, the player needs to guess the distance and direction from the volume and sound direction 4.Player can choose a direction to attack the zombie, the range of the attack depends on the weapon range	1	1

Object	1.3	Saurabh	As an adventure game player, I want to be able to collect both objects and consumables while playing the game to aid my progress through the game and skip through stages.	Given that I am an adventure game player, when I progress through the game, I should be able to skip stages with certain objects and gain power-ups with consumables.	1. Create a general Object Interface, which will contain methods and attributes for any object no matter the type. 2. Create specific classes that implement this interface for specific object types Including generic objects that can be picked up, consumable objects that can be eaten. Further classes can be created as desired on objects wanted (Eg:-Weapons object)	1	2
Scene pictures	2.4	Saurabh	As an adventure game player, I want a visually appealing game scene and animations when I am on the move.	Given that I am an adventure player, I want to be visually able to see my current position in the game and the setting that I am in to help make my decisions easier. Given that I am an adventure player, I also want to know when I am stationary and when I am moving from scene to scene using static and dynamic images(animations).	1. Create visually appealing images that are accessible in the game by a player. 2. Use bright colors to be able to visually depict the scenario and to make scenes easier to see, which will be fed to the software in terms of picture files added to the game using JavaFX. 3. Use online software tools to create animations that indicate the player is moving.	2	3
Audio Descriptions	3.3	Saurabh	As a colorblind/visually impaired person I want to be able to hear all the information needed about the game as I play it.	Given that I am a visually impaired game player, on the change of scene or on request, all information about a scene must be read out at a moderate pace with proper enunciation and in a loud voice.	1.Use software to aid in my creation of audio files specific to each object and room/scene which will be fed to the software in the form of audio files which can be read out using the third-party java screen reader.	1	2
Game story	1.4	Saurabh	As an adventure game player, I want to have an interesting and engaging experience while playing the game.	Given that I am an adventure game player, the game must have multiple pathways which each choice I make affecting my chances of winning/ending the game. There must be multiple paths that lead to success(ending) and likewise some that lead to failure(death).	1. Do research on adventure/survival games of similar kinds and create stories and pathways for the game and how it can be played which will feed to the software in terms of text files.	1	3
Alt text	4.3	Saurabh	As a visually impaired game player I want accessible text for each visual picture and visual scene animation in the game.	Given that I am a visually impaired game player, if I come across an image that provides visual users with information, when I navigate to the image/animation, I should be able to hear the alternative text describing the image.	1. Write accessible text for each image and animation provided which includes the description and options available in that scene, and use the java.accessibility library to incorporate third party screen reader hooks that includes the object description.	2	1

SECTION 3: SOFTWARE DESIGN

Kenson: - Design Pattern 1: *Strategy Design Pattern*



Implementation Details:

Related user stories:

The UML graph represents the user stories (ID) 1.1 and 3.1, which involve the Strategy design pattern and the SOLID principles.

SOLID principles related:

1. Single Responsibility Principle:

Each class seems to have a well-defined responsibility. Troll class is responsible for the basic attributes and behaviors of trollsfightTroll and tradeTroll classes are responsible for specific types of trolls, and the gameTroll class is responsible for the rock,paper,scissor behavior. This helps ensure that each class has a single responsibility.

2. Open/Closed Principle (OCP):

It is always able to exhibit a degree of extensibility; different types of trolls can always get added to the subclass of Troll without having to modify the existing Troll class. It is open to extension but closed to modification.

Key concepts outlined in the UML Diagram:

- Description about UML:

  The Troll class is an abstract class that defines the basic attributes and common behaviors of trolls. The Troll class includes abstract methods that need to be implemented by different types of troll subclasses to customize their behavior based on their differences.

  The fightTroll, tradeTroll, and gameTroll classes represent different types of trolls. They extend the Troll class and implement the specific behavior of each troll type. This aligns with the core idea of the Strategy pattern, which encapsulates different strategies (troll types) into separate strategy classes.

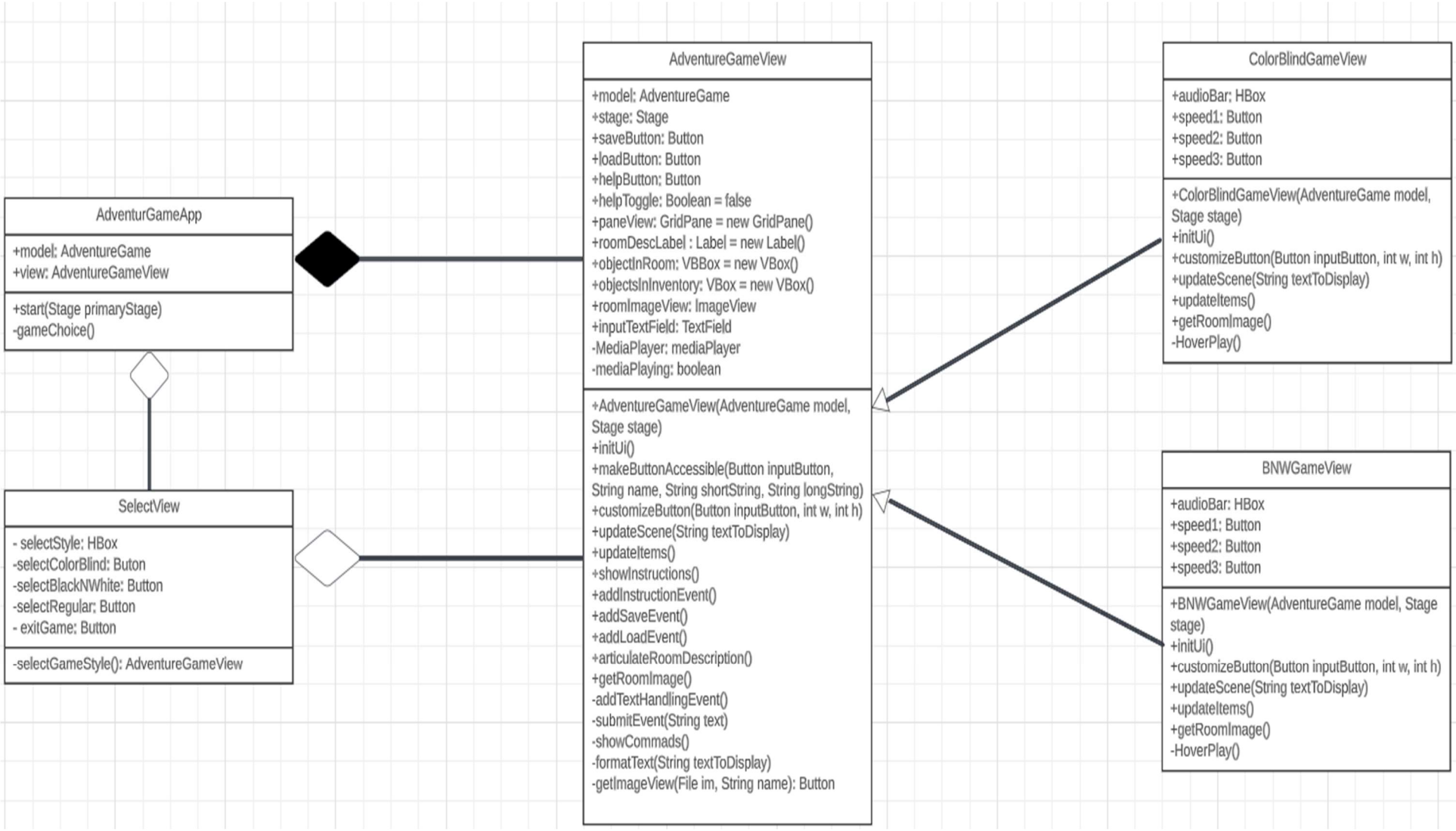
  All troll subclasses implement a common playGame(other: Player) method, which can be considered part of the strategy interface. This method defines how trolls participate in the game, but the specific behavior is determined by each troll type.

- This UML diagram represent the Strategy Pattern:

  playGame(other: Player) method can be seen as part of the strategy interface. Different troll subclasses implement this interface to define their specific game-playing strategies.

  Concrete Strategy Classes: Represented in the UML diagram as fightTroll, tradeTroll, and gameTroll classes. They are implementations of concrete strategies, with each class representing a different behavior strategy for trolls.

Shahmir: - Design Pattern 2: *State Pattern*



Implementation Details: The UML diagram represents the user stories (ID)1.2, 2.2, 2.3, 3.2, 4.2.

Key concepts outlined in the UML Diagram:

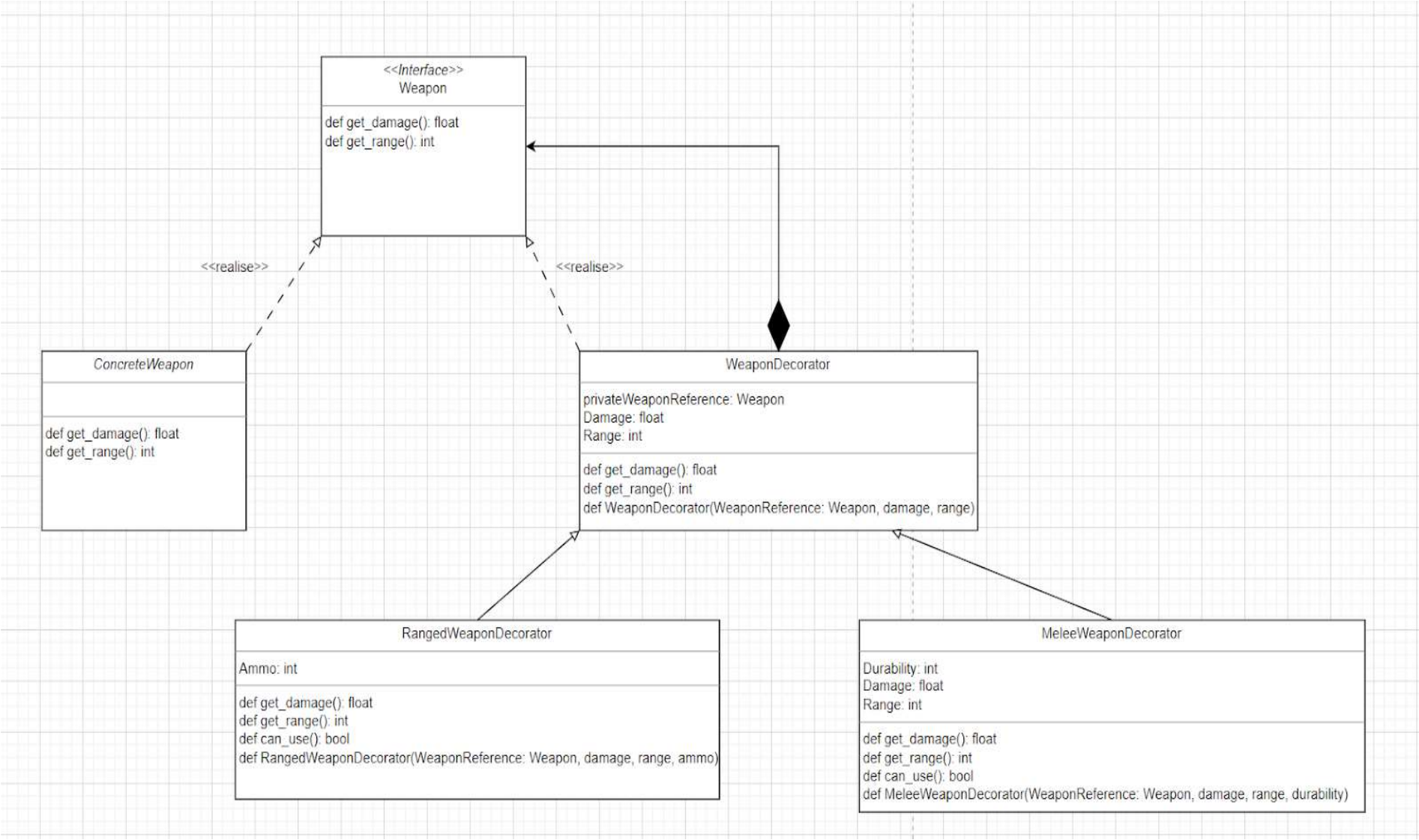
- State design pattern: *AdventureGameView* (Parent class), *ColorBlindGameView*(Child class) and *BNWGameView*(Child class).
    - These classes represent the different version of the game a user can play depending on the type/level of accessibility the user may desire to play with.
- *SelectView* Class: Used to select which game style (type of accessibility features) the user wants to play with, sets the state of the *view* attribute of *AdventureGameApp*, to determine which class of type *AdventureGameView* to play with.
- The *gameChoice()* method calls upon an instance of the *SelectView* class and records the the return value of the *selectGameStyle()* method.
- The *AdventureGameApp* class has an "has-a" relationship with the *AdventureGameView* class, and subsequently its child classes.
- The *SelectView* class is associated by aggregation to the *AdventureGameView* class (and its subclasses), as the method *selectGameStyle()* has a return type of *AdventureGameView*, as it returns a new instance of a class of type *AdventureGameView*.

- The user picks the style of game they want to play using the *selectGameStyle()* method in the *SelectView* class. This sets the state of the type of *AdventureGameView* object for the *AdventureGameApp* class, and accordingly instantiates the appropriate class of type *AdventureGameView* (*AdentureGameView*, *BNWGamView*, *ColorBlindGameView*).
- Each class has relatively similar behaviours but with different implementations to cater to different types of visually impaired users:
    - All the reimplemented methods in the child classes of *AdventureGameView* are JavaFX visual object creation methods, each subclass's implementation of these methods will change the visual of the GUI elements (different background and text colors for higher contrast, color blind friendly palette, larger button sizes, etc.).
    - Another different behvaiour between the *AdventureGameView* and its child classes is the *HoverPlay()* method, when the user's cursor hovers over a button, and audio file stating the button's name and description is played.
- With specific reference to the state design pattern, the state changing is the preference of the type of accessibility the user sets, for which multiple classes of type *AdventureGameView* were created. So that they behave according to the type of accessibility (state) set by the user.

SOLID Principles adhered in UML Diagram:

- Single Responsibility Principle:
    - Each class (*AdventureGameView* and its child classes) all cater to one and different levels of accessibility the game has to offer.
    - It is also easier as a developer to implement the State design pattern by differentiating the roles/responsibilites of each class.
- Liskov's Substitution Principle:
    - Every Child class of the AdventureGameView class can be substituted everywhere AdventureGameView is used, as it has the same methods and attributes, and more. This allows the subclasses to interact with the program as its parent class would break/crashing the application.
- Open/Closed Principle:
    - We implemented new subclasses to cater to different levels of accessibility of the game, rather than manipulating the main, parent class. As that allows us to preserve the way the parent class and the subclasses interact with the program and add new functionality and behaviours.
    - In context of *ColorBlindGameView* and *BNWGameView*; they house all the same features and behaviours as *AdventureGameView*, but also have additional accessibility features such as the *HoverPlay()* method, the audio playback control and GUI elements of different properties. All whilst preserving the way *AdventureGameView* interacts with other elements of the game.

Jonathan: - Design Pattern 3: *Decorator Design Pattern*



Implementation Details

Related user stories:

- The user story pertains to the "Audio Battle for Enemy." Within the Weapon Class, the functionality includes the implementation of the get_damage and get_range() methods, specifically designed for the audio battle feature. During the audio battle, players are tasked with discerning the range of the enemy based on sound cues in each round. Subsequently, they must strategically initiate attacks when they believe the enemy is within range. Upon executing an attack, the get_range() method is employed to ascertain whether the weapon's range encompasses that of the enemy. If a match is found, the get_damage() method is invoked to retrieve the weapon's damage output, which is then applied to the targeted enemy.

SOLID principles related:

- 1)Single Responsibility Principle
  Each class has a single responsibility.
  The Weapon interface: responsible for defining the basic behavior.
  ConcreteWeapon: is responsible for implementing the basic weapon behavior.
  decorators: responsible for adding a type of weapon.
- 2)Open/Closed Principle
  Using the decorators, we can add new weapons without altering the existing code (Weapon or ConcreteWeapon)

Explain of UML graph:

-The solid line with an arrowhead represents the inheritance relationship.
-The dashed line with an arrowhead represents the composition or aggregation relationship.

- 1)Interface: Weapon
  Base interface for all weapons.
- 2)ConcreteWeapon
  Concrete implementation of the Weapon interface.
- 3)WeaponDecorator
  Decorator class that also implements the Weapon interface. It has a reference to a Weapon object. Concrete decorators will extend this class.
- 4)RangedWeaponDecorator
  Concrete decorator of RangedWeapon
- 5)MeleeWeaponDecorator
  Concrete decorator of MeleeWeapon

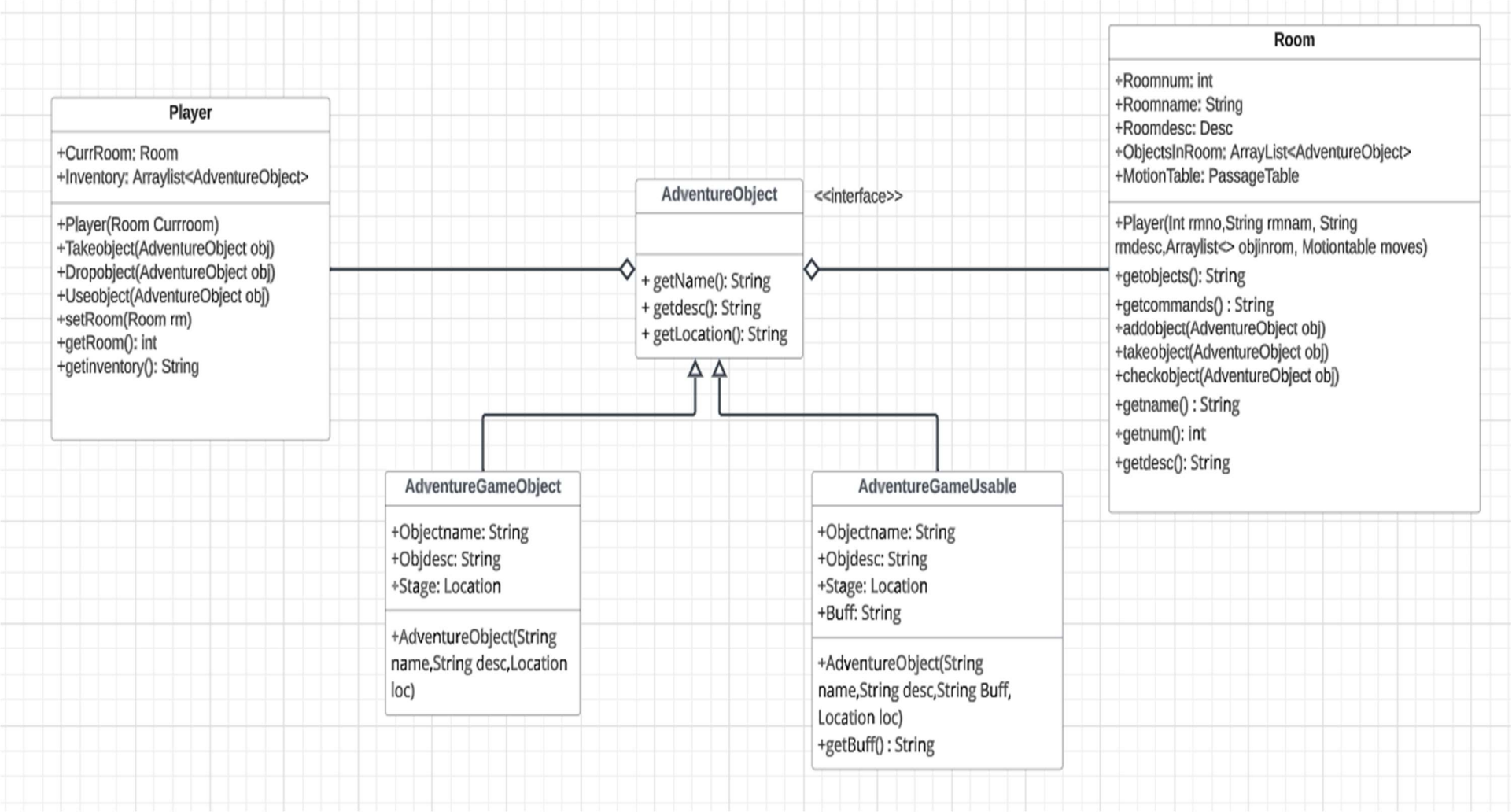
Implemented detail:

- Weapon will be inherited from the inventory item class.
- Weapon will be created by reading the txt files, the value of the weapon may have randomness.

Example Usage:

- basic_weapon = Weapon() # Creating a basic weapon
- gun = RangedWeaponDecorator(basic_weapon, 10.0, 5, 20) # Define a gun with 10.0 damage, 5 range, 20 ammo
- pipe = MeleeWeaponDecorator(basic_weapon, 3.0, 1, 10)   # Define a pipe with 3.0 damage, 1 range, 10 durability

Saurabh: - Design Pattern 4: *Factory Design Pattern*



Implementation Details:

The UML diagram represents the User stories for Saurabh: ID – 1.3, 1.4, 2.4, 3.3, 4.3.

Key Concepts outlined in the UML Diagram:

- Factory Design Pattern: *AdventureObject()* : Parent Interface & *AdventureGameObject(),AdventureGameUsable()* => Child classes
  - These classes represent the different types of objects that a player can pick up: One is a usable type which can be used once and is then lost, one is a simple game object that can be picked up and stored in player inventory and used to unlock game paths.
- *Player* class which maintains a has-a relationship with *AdventureObject()* as it stores list of adventure Objects in inventory of player, and uses *Adventure Objects()* as parameters in *Take, Drop* and *Use* methods.
- *Room* class maintains a has-a relationship with *AdventureObject()* as it stores a list of Adventure Objects as objects in room and uses *Adventure Objects()* as parameters in *Take, Add* and *Check* methods.
- *AdventureGameObject* class and *AdventureGameUsable* class are specific kinds of Adventure Objects that have different implementations and usages thus they are child classes that implement the parent interface *AdventureObject()*.

***Factory Design Pattern* explained in the UML Diagram**

- The *Take* and *Drop* object method of *Player* class will access the adventure object specified and either remove object from *inventory* of player or add object to *inventory* of player. The *Use* object method of the *Player* class will check if the object specified is of the *AdventureGameUsable* class and only then will consume it, else it will return specified error.
- The *Take, Add* and *Check* object method of *Room* class will access the adventure object specified and either remove object from *objectsinroom* or add object to *objectsinroom* or check if object in *objectsinroom*.
- The *Factory design pattern* is most usable for this scenario as it is a design pattern used for object creation, as rooms are read from text files and data is created about each room, we need to be able to create separate type of objects as needs are facilitated so that the player may be able to access these objects as the game progresses. The *Factory design pattern* allows us to specify a general adventure object overview but leaves room for modification and additional implementation, this includes additional object types that can be used in the future.
- The *Factory design pattern* specifies the interface for an adventure object in the game and allows developers to implement this interface and add it on to it by modification and extend this design pattern to further child classes.

***SOLID* principles adhered to in the UML Diagram:**

- Single Responsibility Principle: Each class also has a specific responsibility it caters to, and thus we create specific classes for each responsibility we need including the adventure object class and the usable object class.
- Dependency Inversion Principle: We reduce chains of dependence between classes hence we can change an individual piece by modifying just that part of code. We create abstract interfaces (AdventureObjects) to create adventure objects, adventure usable objects and more in the future.