Group 2 - *Angel Prisoner From Hell*

Dec 5, 2023

Aaron Li
Aidan Cullen
BangCheng Wang
Vo Khoi Nguyen Nguyen

This project is built upon Assignment 2. We added multiple functionalities to our project to make the game more interesting to play. The update includes:

+ Having a GUI based game
+ Using the keyboard to move the player around the screen (WASD or ↑↓←→)
+ Having NPCs that the player can interact with to give the game's story
+ Having enemies for players to defeat, as well as a final boss
+ Having a map as an item the player can use to navigate the game

Since we want our game to be played by anybody, we included accessibility features to accommodate users with different needs:

+ Text to speech description of each room
+ Adjust brightness of a room to lessen contrast, thus lessening eye strain
+ Volume adjustment feature for users who wish to lessen the game music

User Stories

Name	ID	Owner	Description	Acceptance Criteria	Implementation Details	Priority	Effort
Move player	1.1	Nguyen	As a user, I want to move my character around using keyboard keys WASD so that I can move my character to a new position	- Given that I am a user, when I press the correct key (WASD), then the character moves in the direction that I want. - Given that I am a non-sighted user, when I press the key, then the audio will be displayed so that I know which direction my character moves	-The player class will be responsible for reading the key event input. The position of the player is represented by x, y coordinate, moving will change the value of x and y. Check if x and y is out of bound -There will be another class which depends on the player class and this class will update the new position of the player. Delete the image of the	1	2

					old position and put an image of the new position. -There will be another class responsible for the audio part, whenever the key is pressed, this class will be called		
Menu	1.2	Nguyen	As a normal user and non-sighted user, I want a menu, which contains the information of the game as well as the settings, at the beginning of the game so that I know what the game is about, how to play the game and change the game's setting.	- Given that I am a normal or non-sighted user, when the game is displayed, then the first thing I see is the menu which contains the instructions and the game setting, and the button is accessed by either mouse or button	- Display the menu page with start button, instruction button and setting button. - Make all the buttons accessible by mouse and keys. - When the instruction button is chosen, show an instruction. An instruction has the return button to return to the menu - When clicking the start button, start the game. - When clicking the setting button, a setting will be displayed so that the user can adjust the display that they want. This widget have a return button to return to the menu	1	2
Enter room	1.3	Aaron	As a user, I want my character to move to a room that I navigate, so that I can move my character between rooms.	- Given that I am a user, when my character position is at the edge of the room, then my character enters that room.	- Keep track of the position of the player, when the position is at a certain value i.e middle top, middle bottom, middle left, middle right. The player will enter a desired room. - Change the current room of the player, reset the position of the player which respects the room that it enters. For	1	1

					example, when it moves to the left, in a new room image, the position of the character should be at the right of the screen		
Status	1.4	Aaron	As a user, I want my character to have status such as health, attack, money, inventory so that I can interact with the game features.	- Given that I am a user, when I interact with the game then my character status is updated according to what I have done	- In the Player class, create attributes of health, attack, money, inventory. - For each attribute, create a set and get method so that other classes can interact according to a specific design pattern.	2	3
Command Box	1.5	Aaron	As a user, I want a command box to execute the actions such as take objects, drop objects and buy items from the shop, possible alternatives could be one character input commands, such as t, d, and b, so that I can pick up objects, drop objects, and buy objects to perform a functionality with the objects that I have	-When a shortcut character is pressed, the action is performed -Audio is played for those who have visual impairments	- All the action take, drop, buy will depend on the Player class. - Using command design pattern to implement the take, drop, buy command. These commands will change the attribute of the class Player.	1	2
interactive NPCs	1.1	Aidan	As a gamer who prefers games in which the game's story is told through characters, I'd like to talk to other humans/entities in the game so the game feels more immersive.	The user is able to interact with an in-game entity by typing "TALK." The entity tells the player part of the game's story, or gives a hint on what to do next.	Design pattern: Visitor All NPCs have the same common base class. During interaction, a text box will appear at the bottom of the screen, and words will fill the text-box. When the player presses enter, the	2	3

					previous text will disappear, and the next portion of text will fill the box. The user can keep pressing enter until the NPC has nothing left to say. If the user decides to interact with the NPC again, this behavior will repeat.		
NPC sprites when talking	2.1	Aidan	As a dyslexic gamer, it is hard for me to read what NPCs say, and this diminishes my gameplay experience. I need a way to feel more connected to the NPCs other than reading their monologue.	When the user interacts with an NPC, an image of them appears in the chat box as they display their text.	When the user first interacts with an NPC, a sprite of the NPC will appear beside the text box. When the user presses enter, that same image of the NPC will keep appearing until the player exits chat mode.	3	1
NPC Spanish translation	3.1	Aidan	As a spanish-speaking user, I need a way to understand what the NPCs are saying to me so I can progress in-game.	A button to push so that the NPCs' language is translated into Spanish from English, and into English from Spanish.	When the user interacts with an NPC, there will be a button on the chat box that, when pressed, will translate the text into Spanish. If that button is pressed again, the language will be translated back into English. If the user interacts with another NPC while in "Spanish mode", that NPC will also speak Spanish.	2	2
AI-assisted NPC dialogue	4.1	(Whole group)	As a user who values variety in video games, I want NPCs to behave like real people: vary the way they speak each time	Every time the user starts interacting with the NPC, the dialogue will be different.	Store the variety of the dialogue in the NPCs class. Having a value to determine which dialogue to display. This value can be random or can be specific according to the player attribute.	3	3
Map	3.2	Nguye	As a user, I want to	- Given that I am a	- Create a scroll pane at	3	s

navig ation		n	see the map at the bottom of the screen, so that I know which rooms I have passed.	user, when the player goes to a new room, then the map at the bottom of the screen will be updated.	the bottom of the screen, then add the representation image of the room to the map whenever the player enters a new room. - Change the center of the map to the current room		
Inve ntory	3.3	Bang Cheng	As a user, I want to keep track of the objects in the inventory, so that I know what I have and I can drop them.	- Given that I am a user, when I click the inventory button, then all the objects that I have collected show up.	- Create a button for the inventory, on-click the grid pane will show the image of all the objects.	3	1
Save & Load	3.4	Bang Cheng	As a user, I want to save as well as load the game, so that I can play the game that I am left with.	- Given that I am a user, when I save the game and then load it, then the exact same game will show as the last time I save it.	-Serialize the Game into a .ser file and then load it back.	3	1
Basic Mons ter types	4.1	Bang cheng	As a player, I want to encounter various types of monsters so that the gameplay remains challenging and engaging.	Given that I am navigating through the game, When I encounter monsters, Then I should meet at least three distinct types of monsters with unique visual designs and attack patterns.	-Design a few basic types of monsters with different attributes (such as health, attack strength). These monsters can share some fundamental behavior patterns to reduce development complexity.	1	3
Final Boss Mech anics	4.2	Bang cheng	As a player, I want the boss to have two separate health bars, and upon depleting the first health bar, the boss should switch to a second, more	Given that I am in the boss battle, when I deplete the first health bar, then the boss transitions to a second phase with a new health bar and changes its	Develop the boss with two health bars, scripting the change in behavior and possibly the appearance to indicate the new phase. Include distinctive animations or effects to	3	3

			challenging attack mode, making the battle more dynamic and engaging.	attack mode. Tests confirm that the boss's health bars deplete correctly and that the transition to the second attack mode occurs seamlessly. Ensure that the second attack mode is distinctively more challenging and provides a visual and gameplay cue to the player that the boss phase has changed.	signal the phase shift to the player.		
battl e-ui	4.3	Bang cheng	As a player, I want to see a simple and clear battle-ui which has statuses of the player and monster, and buttons for the player to choose. As a player,if I defeat a monster , then I cannot fight again with it.As a player, if I am defeated by a monster, it will appear an image which means the player loses.	Given that I click the monster image and start-battle with a monster, there is a battle-UI which has statuses of the player and monster, and buttons for the player to choose. If possible, I will play the battle-bgm and implement the animation for battle-UI.	To implement it, writing methods in adventuregame.java and adventuregameview.java .In adventuregame.java, write methods which are named star battle,attack monster… In Adventuregameview.java , write methods like showbattleUI,updatebattleUI,checkbattleEnd,handlegameover…	2	3

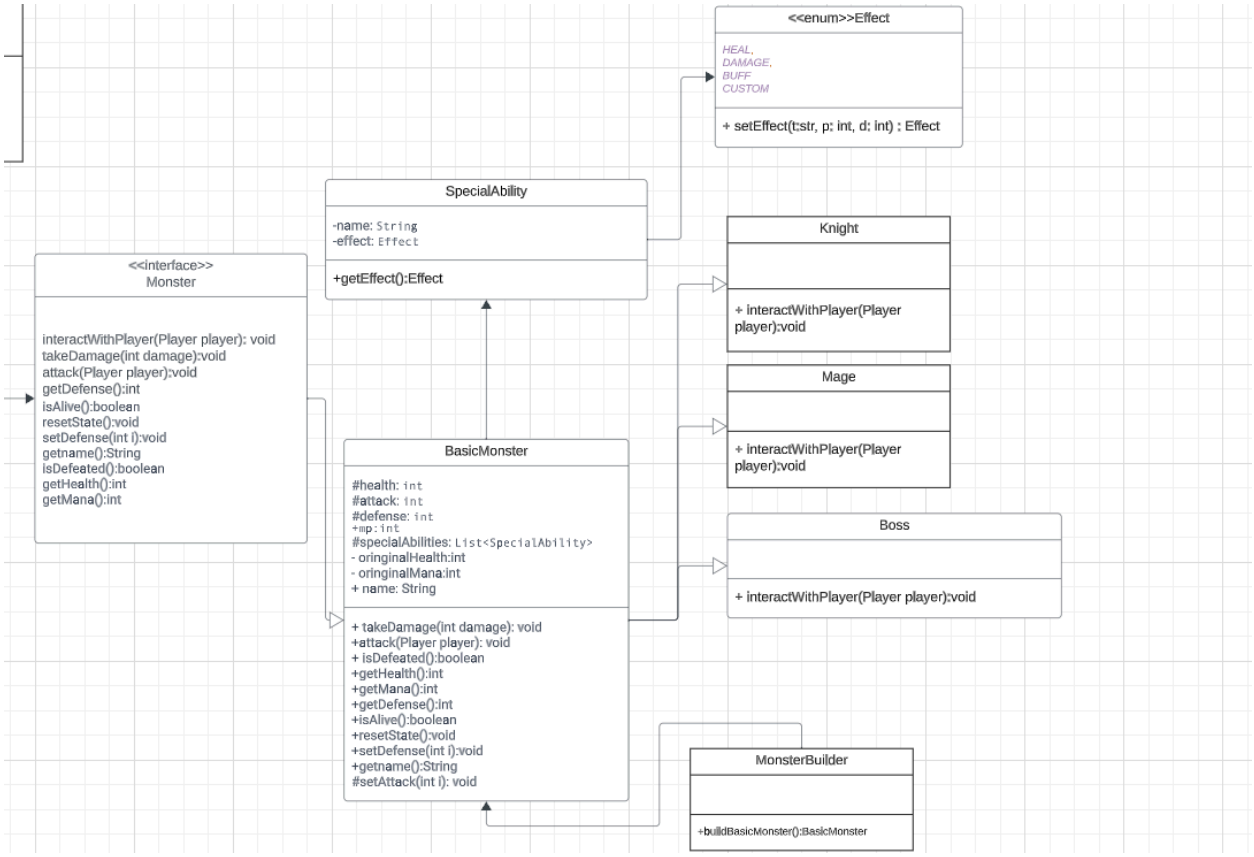
Design Pattern #1: Builder Pattern（Bangcheng)

Overview:

The Builder Pattern will be used to construct complex objects by separating the construction process from its representation. This pattern is beneficial for the

monster system, where each type of monster has different attributes and behaviors that need to be built step by step. (user stories 4.1-4.3)

UML Diagram:



Implementation Detail:

Monster Interface:

- Defines the contract for all monster types in the game, including methods for interaction with the player, taking damage, and performing attacks.
- Ensures that all concrete monster classes adhere to a common set of behaviors, promoting polymorphism.

BasicMonster Class:

- Implements the Monster interface, providing base attributes like health, attack, and defense.
- Can be extended by specific monster types, which suggests a hierarchy of monster complexity.

Boss Class:

- Extends BasicMonster, representing a stronger monster type within the game.
- Overrides methods to provide more complex behaviors, such as changing attack patterns when health is low, hinting at more advanced AI for boss types.

Knight Class:

- Another subclass of BasicMonster, tailored with custom behavior in interactions, such as increasing defense in certain conditions.
- This specialization demonstrates polymorphism and the open/closed principle, where classes are open for extension but closed for modification.

Mage Class:

- A subclass of BasicMonster with unique interaction logic, possibly related to mana and attack power.
- Showcases how different monster types can have distinct behaviors while still fitting into the game's overall structure.

MonsterBuilder Class:

- This class is responsible for constructing different monster types. It provides a fluent interface for setting various attributes, a typical characteristic of the Builder pattern.
- The buildBasicMonster() method compiles these attributes into a new BasicMonster object, encapsulating the construction logic within the builder and keeping the BasicMonster class free from complex initialization code.

SpecialAbility Class:

- This class represents a special ability that can be possessed by monsters or players within the game.
- Each SpecialAbility object has a name and an associated effect, which is defined by the Effect enumeration.
- The constructor SpecialAbility(String name, Effect effect) allows setting the name and effect of the special ability upon creation.
- The getEffect() method provides access to the special ability's effect, facilitating the retrieval of the effect's type for game logic.

Effect Enumeration:

- This enumeration defines the types of effects that special abilities can produce, such as HEAL, DAMAGE, BUFF, DEBUFF, and CUSTOM.
- Effect simplifies the implementation and management of special abilities by providing a predefined set of effects for different game mechanics.

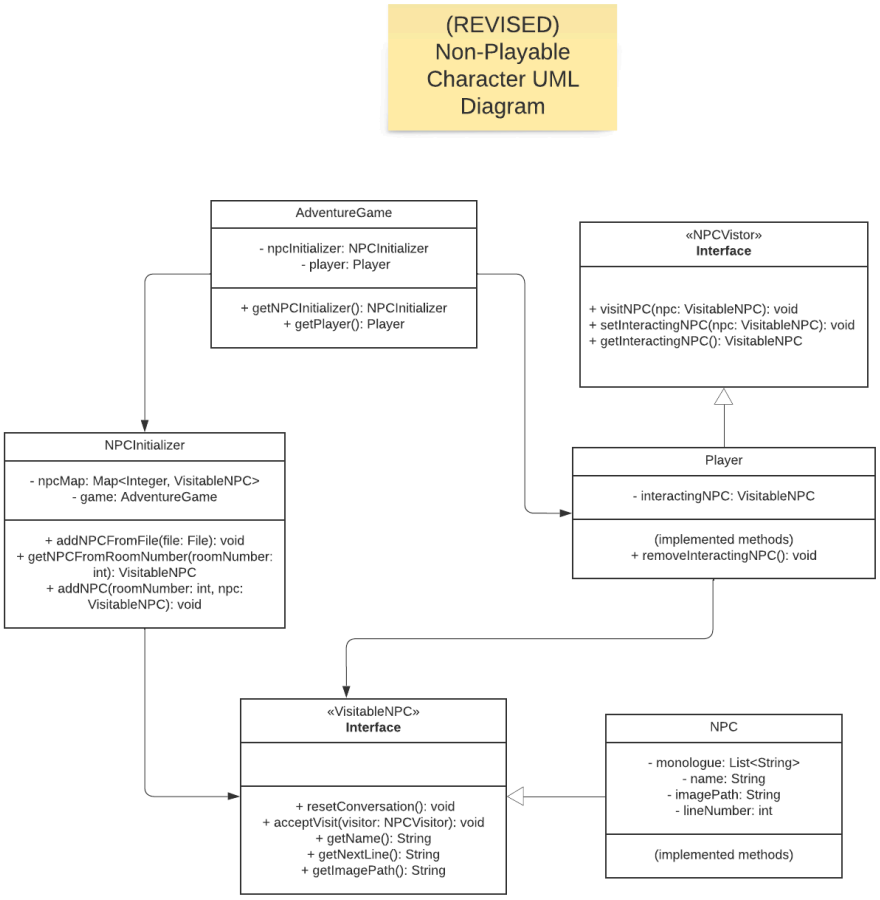
I changed the 4.3/MonsterDrop user-story to 4.3/battle-ui.

battle-ui	4.3	Bang cheng	As a player, I want to see a simple and clear battle-ui which has statuses of the player and monster, and buttons for the player can choose. As a player,if I defeat a monster , then I cannot fight again with it.As a player, if I am defeated by a monster, it will appear an image which means player lose.	Given that I click the monster image and start-battle with a monster, there is a battle-UI which has statuses of the player and monster, and buttons for the player can choose. If possible, I will play the battle-bgm and implement the animation for battle-UI.	To implement it, writing methods in adventuregame.java and adventuregameview.java.In adventuregame.java, write methods which named startbattle,attackmonster… In Adventuregameview.java, write methods like showbattleUI,updatebattleUI,checkbattleEnd,handlegameover…	2	3

Design Pattern #2: Visitor Pattern - Aidan Cullen
Feature: Addition of Interactive Non-Playable Characters (NPCs)

Overview:
NPCs are a foundational game mechanic in many modern story-based games, such as the Grand Theft Auto series, or The Legend of Zelda. Without NPCs, these games would be lacking several foundational narrative elements, such as exposition, plot, and narrative itself. These NPCs will serve to provide story and hints to the player, in order to help immerse them into the game's universe.

Explanation of design pattern
The visitor design pattern is used to separate the messy implementation details of visiting an NPC from the act of invoking it. Any visitable NPC must accept a visiting object through its acceptVisit() method. Since the visiting object is only aware of the implementation of the NPC it's visiting, the data of each individual NPC is encapsulated.



Relevant SOLID Principle(s)

Open-Closed Principle

If different types of NPCs are added in the future, they would inherit the behaviour of the NPC subclass without any modification to NPC or VisitableNPC.

Description of Classes and Interfaces

NPCInitializer

This concrete class is responsible for initializing all NPCs in the game by invoking addNPCFromFile() for each text file representing the NPC to create. The actual data stored in these text files include monologue, room number, name, and any other attribute to be used in future NPC variants.

<<NPCVisitor>>

An interface implemented by all NPCs visitable by the player. This interface allows NPCs to be sent a visit request by invoking the vistor's visitNPC() method.

Player

The agent controllable by the user, which also serves as the visitor in the visitor design pattern. The Player calls the NPCs visitNPC() method upon interaction in order to request to speak to the NPC.

NPC

The basic implementation of an NPC that is inherited by all future NPC variants. It has an acceptVisit() method which takes an NPCVisitor attribute, then interacts with the visitor.

<<VisitableNPC>>

This interface, implemented by all visitable NPCs, allows them to interact with the player through the acceptVisit() method.

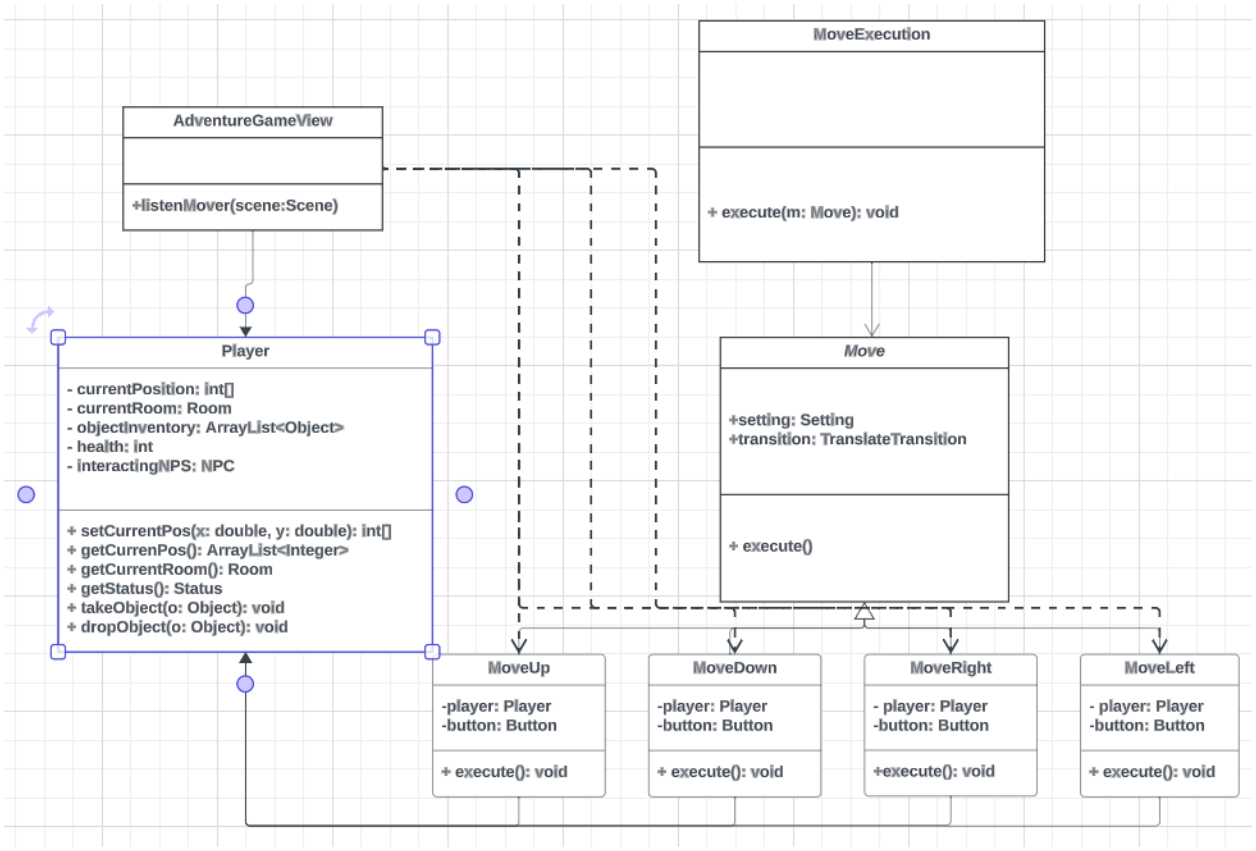
Game

The centerpiece of the Model-View-Controller system. This class invokes the NPCInitialization methods and holds the view for the user to interact with.

Design Pattern #3: Command Design Pattern (Nguyen)

Overview: This pattern will be used to implement Move player functionality in user story 1.1

UML diagram:



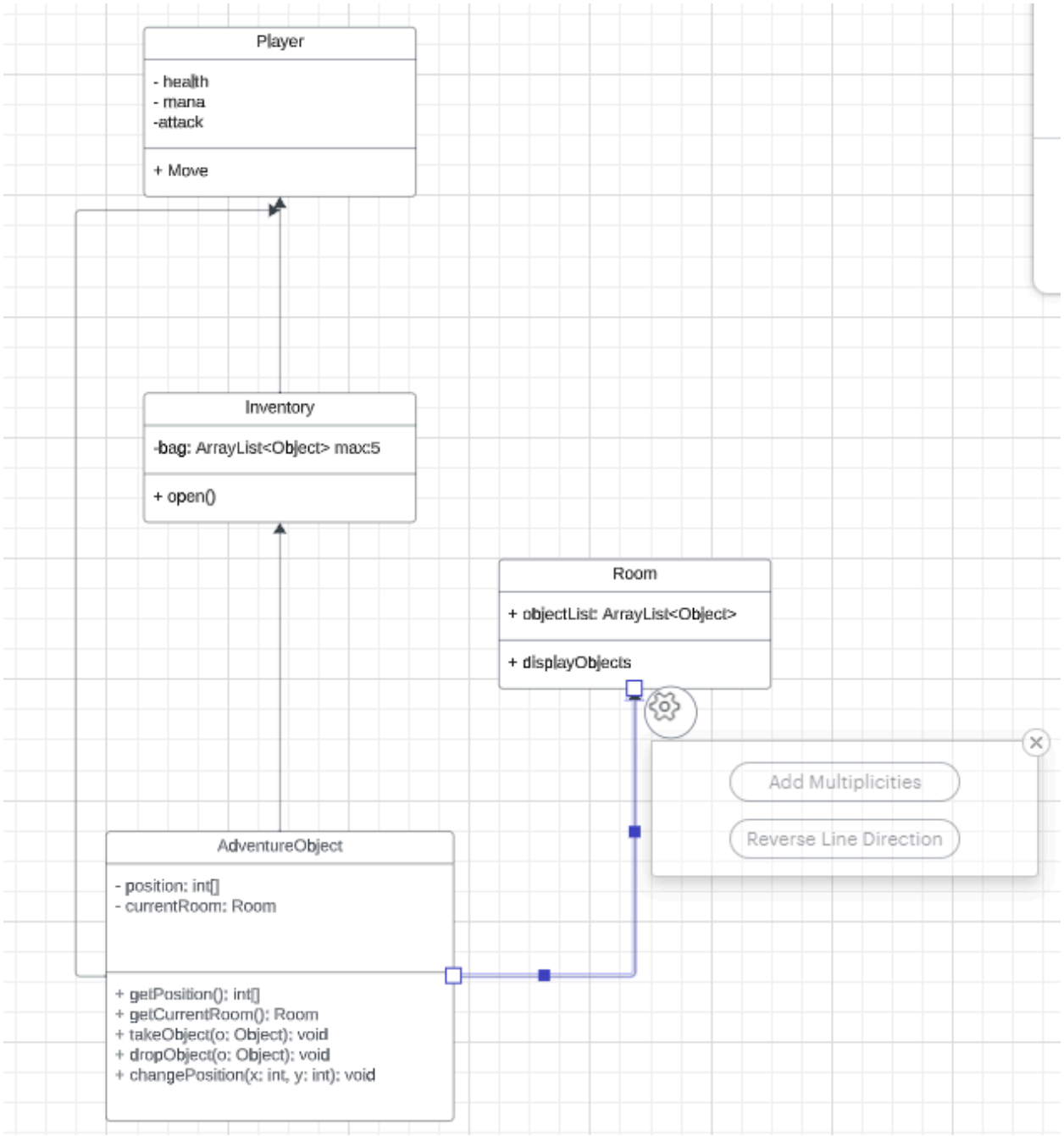
Implementation Details: The UML outlines these main components:

- The *Move* interphase which has an execute method. *Move* also has setting attribute which determine how fast the player move and transition attribute to perform the animation of moving the player
- 4 classes which implement the *Move* interphase are: MoveUp, MoveDown, MoveRight, MoveLeft.
- They will change the player position attribute. After changing the position of the player, they will use that to redraw the button.
- The MoveExecution receives Move as a parameter for the execute() method. This method will call the execute() method in Move.
- The Player class which is the Receiver and will be changed as the *Move* is executed.
- The Adventure Game class is used to instantiate MoveUp, MoveDown, MoveLeft, MoveRight in listenMove() method

The MoveExecution received only 1 *Move* because it is designed to execute 1 move at a time and it will execute the *Move* by calling execute() method. The implementation of *Move* will change the position of the Player by using the setCurrenPos(x, y) method in the Player class. *Move* also change the position of the player image in the game by using transition attribute. The AdventureGameView is

the main class for the game view and will create MoveUp, MoveDown, MoveLeft, MoveRight. The class does so in the listenMove(scene) method. I did not add many methods and attributes in it, but it will have many methods and attributes later on when we are working on it.

Design Pattern #4: Strategy Design Pattern (Aaron)
Overview: This pattern will be used to implement the adventureObject used throughout the game (user story 3.1-3.3)
UML diagram:



Implementation Details: The UML outlines these main components:

- The *Player* interphase which has status attributes, with health and attack
- 4 classes which implement the *Player* interphase are: AdventureObject, Player and Inventory and Room
- They will each operate to display objects in the room and inventory, as well as use them for their functionality
- Each AdventureObject can be picked up from the room and be placed in the players inventory, as can be used in their inventory

- The Player class which is the Receiver will be able to use/consume the adventureobjects throughout the game
- The AdventureObject class is implemented by getPosition, getCurrentRoom, takeObject, dropObject, changePosition. These are all to help the implementation of object

The AdventureOBject class serves as the main class for the game view and is responsible for creating instances of getPosition, getCurrentRoom, takeObject, dropObject, changePosition. These inventory is instantiated by the JavaFx gridpane, the player statuses are instantiated by normal java attributes, AdventureObjects classes are generated by the use of JavaFx buttons, and Room is instantiated by the use of the attribute scene from javaFX