CSC207 DESIGN DOCUMENT TEMPLATE

Design Document

Version 1.0 – 2023.12.05

Project Name: RealmQuest

Link To Demo:
https://www.youtube.com/watch?v=CQViQ_DSMws

Project Members

David Kochanski
Habeeb Fadak
Devesh Aggarwal

GitLab Repository:

https://mcsscm.utm.utoronto.ca/csc207_20239/group_12

Revisions from Previous Version:

- Assign user stories
- “Crossed out” some unused user stories
- Revised the UML diagrams to reflect how the code actually turned out
- Updated the Implementation Details
- Add Singleton Pattern example instead of duplicate Factory

SECTION 1: PROJECT IDENTIFICATION

This project will be an expansion on the choose-your-own-adventure genre by incorporating top-down adventure game elements in a fully interactive world. Instead of picking up items by entering a command, the user will be able to freely move along a grid to pick up items and weapons, and to progress through the game. The user will also encounter enemies and will be able to use potions, weapons, and armor to defeat them.

This game will have the same premise as A2, being GUI based and being based around dialogue and interacting with key items to progress. But, the player will have a much higher degree of movement, more in depth interaction with the world, and random variance to keep gameplay fresh.

SECTION 2: USER STORIES

NAME	ID	OWNER	DESCRIPTIO N	ACCEPTANCE CRITERIA	IMPLEMENTATION DETAILS	PRIORITY (1 is highest)	EFFORT (1,2,3, 5,8,13)
GUI Layout	1.1	David	As a user, I want a clearly defined GUI with defined sections to facilitate a seamless gaming experience.	- User is presented with a clear display of the current HP and inventory. - Player character and environment interactions (such as obstacles, interactable objects) are visually distinct. - The game initiates	- Make the environment as a JavaFX GridPane, where each cell represents a different tile in the game world. - Make a Tile class that has a texture, the current item or enemy on it, it's behaviour when stepped on etc.	1	5

				from a menu screen that transitions to the game environment. - The game environment is designed as a grid with varied elements on each tile to represent different terrain or items.			
Item Usage	2.6	David	As a user, I want to use items to enhance my power in the game for a richer experience.	- Items are discoverable in the world and can be interacted with for pick-up. - Users can equip items like weapons and armor to alter damage output and resistance. - Consumable items are usable and have immediate effects such as healing or buffs. - Certain items are key to the story, like to access certain rooms.	-Item class with a pickup method that adds items to the player’s inventory - Define weapons and armor and consumables as subclasses of Items with methods to equip/unequip affecting player stats. - Add checks to certain doors requiring key items - Items will have a use() method	1	3
Enemies and combat	3.3	Habeeb	As a user, I want to defeat enemies to progress and test my strategy and skills.	-Users can attack enemies within their range. -Enemies will attack back and lower the user’s HP -Some enemies drop items or keys when defeated.	-The game will be turn based. On a turn, the user will move 1 space or attack. Then the enemy will move 1 space or attack. -There will be updatePlayer() and updateWorld() methods to the game that will be	1	5

					called handling all of this. -Create an enemy class that has attributes, drops, etc. Make the damage amount slightly variable/random		
Room Exploration and Dynamics	3.4	Habeeb	As a player, I want to explore different rooms in the dungeon to uncover treasures and battle enemies.	-Rooms have doors that act as tile-based transitions. -Doors are two way. -Some doors may require keys in the inventory to unlock and pass through. -Each room contains items on the floor, enemies, and maybe obstacles	-When the user goes on a tile, if it's a door tile, move to that door tile's connecting room.	2	3
Inventory Management	3.5	David	As a player, I want an intuitive inventory system that I can navigate easily using both mouse and keyboard.	- Users can select items in the inventory using mouse clicks or number keys on the keyboard. -The selected item is highlighted in the inventory bar. -Inventory can be toggled to show or hide with a keyboard shortcut. - The inventory has a limit on how many items can be	-Bind event listeners to item icons for mouse clicks and number key presses to select items. -Create a keyboard shortcut using JavaFX event handlers that toggles the visibility of the inventory. -Design an inventory UI using JavaFX HBox that holds item icons. -Create a keyboard shortcut (1,2,3,4,5,..) using JavaFX event handlers that toggles the visibility of the inventory.	1	2

				carried.			
Home Screen Navigation	1.3	Devesh	As a player, I want to start on the home screen that allows me to start playing or adjust settings before I begin.	-The home screen displays a 'Start Game' button to enter the main gameplay. -The home screen includes an or 'Settings' button that opens the game settings.		3	2
Character movement	2.3	David	As a player, I want to move my character in cardinal directions to navigate through the game world seamlessly.	-The player character can move up, down, left, and right using the WASD or arrow keys. -The player is constrained to move within the limits of the walkable environment.	-Capture key press events using setKeyOnPressed and update the player's position accordingly. -When the player is about to move, check if it's valid (no wall or enemy currently on that tile)	1	2
World Engagement	2.4	Debesh	As a player, I want to interact with objects and the environment to enrich the gameplay experience.	-The player can pick up items by moving onto the tile that the item is on. -The item goes into the inventory. -The player can drop items in the inventory (maybe) -Certain doors look in the inventory for a specific item.		3	3
~~World Variance~~	3.2		As a player, I want to encounter randomness in item availability and attack outcomes to	-The items that a player can find in the game world spawn at random locations each game session.	-Use the Random package to dictate some of these things	4	5

			make each playthrough different!	-The attack damage has a bit of variance, for the player and the enemies. Some of the environment can be randomized too			
Textures	2.2	Habeeb, David, Devesh	As a player, I want each room to have unique textures so that the game environment more not dead	-Rooms can have different elements like maybe lava pools, rivers, decorations.	-When going into a room, make some tiles have a small chance of having a decoration instead of the basic tile	4	1
Instructions/ HelpText	2.7	Devesh	As a player, I want to be able to view the instructions for the game so that the game can be played without guessing controls.	- A pop-up button can be available at the bottom right of the screen for info/instructions - Clicking on the button toggles the help text to the screen via A textView	- Use JavaFX to manipulate the game GridPane object to display help text	2	2
~~Difficulty/Accessibility~~	3.5		As a player, I want to be able to change the size/difficulty of the board so that the games take longer, and more effort to complete	- under the settings view, a number selector is visible and updates the game's difficulty level - A pop-up button can be available at the bottom right of the screen for settings	- Use text input to get difficulty number desired by user - display error message if the number is out of bounds or invalid - get the value of the textbox, OnClickoff, so user does not have to click save	3	8
~~Sounds~~	3.1		As a player, I want to be able to change the	- under the setting view, a scrollbar is visible indicating the	- Under GridPane of settings, use JavaScrollPane (or	3	3

			music/sounds so that	current sound of music, sound effects, and other sound related items - A pop-up button can be available at the bottom right of the screen for settings	equivalent) to create a scrollable number input for volume - set the value of the game volume to the value inputted by the user once clicked off		
~~Save/Load~~	1.2		As a player, I want to be able to save/load the current game so that I can play later	- Under settings, a save/load game button can be found - A pop-up button can be available at the bottom right of the screen for settings	- write the game object to a serialized binary file that ends in .<GAME NAME SUFFIX>	3	3
Visual Accessibility	1.3	Devesh	As a player, I want to be able to change the font size/colour of the text displayed so that I can read the text better.	- use real-time text updating, so that the user gets a feel for how the text will look before committing their choice - A pop-up button can be available at the bottom right of the screen for settings	- Set an on keydown listener for the input field, and update the font size based on the input size - display error message if input is invalid, or is out of bounds	3	3
Enemy	2.1	Habeeb	As an enemy character, I want to be able to chase a player so that they are eliminated	- Calculate the most efficient path from the enemy to the player, and take that path	- Create a helper method which calculates the distance from the enemy to the player	5	5
~~Multiplayer~~	4.1		-As a player, I want to be able to play with friends in	- Create a server which hosts games in which players can join and play.	- Use Java ServerSocket to establish Server - Client connection, and	10	13

			real-time so that we can have a shared experience.	- Server and client updates at 60fps	use it to communicate changes/updates		
Health change/Accessibility	4.2	David	As a player, I want to be able to tell when I have gotten hit by an enemy and have lost hp. and when I have gained hp via potions.	- Use an audio cue similar to articulating the room in A2. - There are clear audio indications of whether a player has healed or has taken damage by an enemy. - If a player has healed it could be a cheer or if they have taken damage. it could be an “ouch” sound.	- Create a folder with sound files within the game files folder to be accessed. These can be sounds for when the player has gotten hit, or has healed hp. - Use a media player object to access and play the mp3 files from the sound files within the game files. - Use different audio articulation methods similar to a2. When the player is in an idle state(no sound will be playing), and when they get hit or healed, different articulation methods will be called. - Can be a part of an event handler for any combat events.	5	2
Boss Enemy	4.3	Habeeb	As a player, I want to be able to tell when I am facing a boss enemy and an indication to be displayed when I am encountering one.	- The boss enemy has a bigger hp bar and is bigger in size - Could have a special/unique name - Only one boss enemy in the entire game	-Utilize an inheritance relationship from the enemy class and create a final boss enemy subclass - Its Hp bar will look slightly different and can be placed underneath the player's hp to differentiate it from normal enemies	2	4

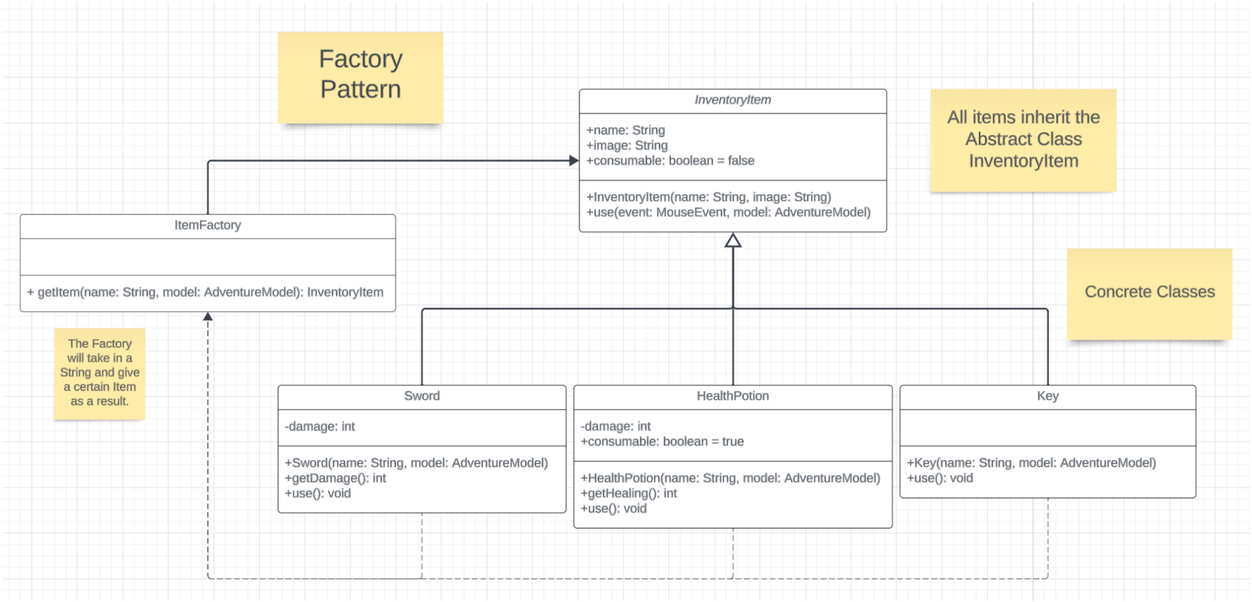
					as normal enemies would have their hp bar above the enemy itself -Utilize an associated textlabel object to display that the player has encountered a boss.		
End Game	4.4	Devesh	As a Player, I want a clear indication of when I have beaten the game or have lost the game.	- Clear the defeat screen when the player has lost all HP. - Defeat screen will prompt the user to load a saved game or start a new one - Clear winning screen when the player has defeated the final boss	- Change the state of the game upon player hp attribute <0. and Visualize a defeat screen. With the load game button and new game button prompts. - Change the state of the game upon boss enemy hp attribute <0. and Visualize a victory screen.	2	4

SECTION 3: SOFTWARE DESIGN

Below are some of the portions of the UML that represent design patterns that we will be using in this project.

Here is a link to the entire UML: https://lucid.app/lucidchart/2202a0d5-98fe-4c7f-82f9-c9ab40a73af9/edit?viewport_loc=-73%2C-85%2C1740%2C996%2C0_0&invitationId=inv_c162754f-e93e-4476-8248-174a0893294

#1 Item Spawning and Interactions


Factory Pattern
InventoryItem
+name: String
+image: String
+consumable: boolean = false
+InventoryItem(name: String, image: String)
+use(event: MouseEvent, model: AdventureModel)
All items inherit the Abstract Class InventoryItem
ItemFactory
+ getItem(name: String, model: AdventureModel): InventoryItem
Concrete Classes
The Factory will take in a String and give a certain Item as a result.
Sword
-damage: int
+Sword(name: String, model: AdventureModel)
+getDamage(): int
+use(): void
HealthPotion
-damage: int
+consumable: boolean = true
+HealthPotion(name: String, model: AdventureModel)
+getHealing(): int
+use(): void
Key
+Key(name: String, model: AdventureModel)
+use(): void

Implementation Details:

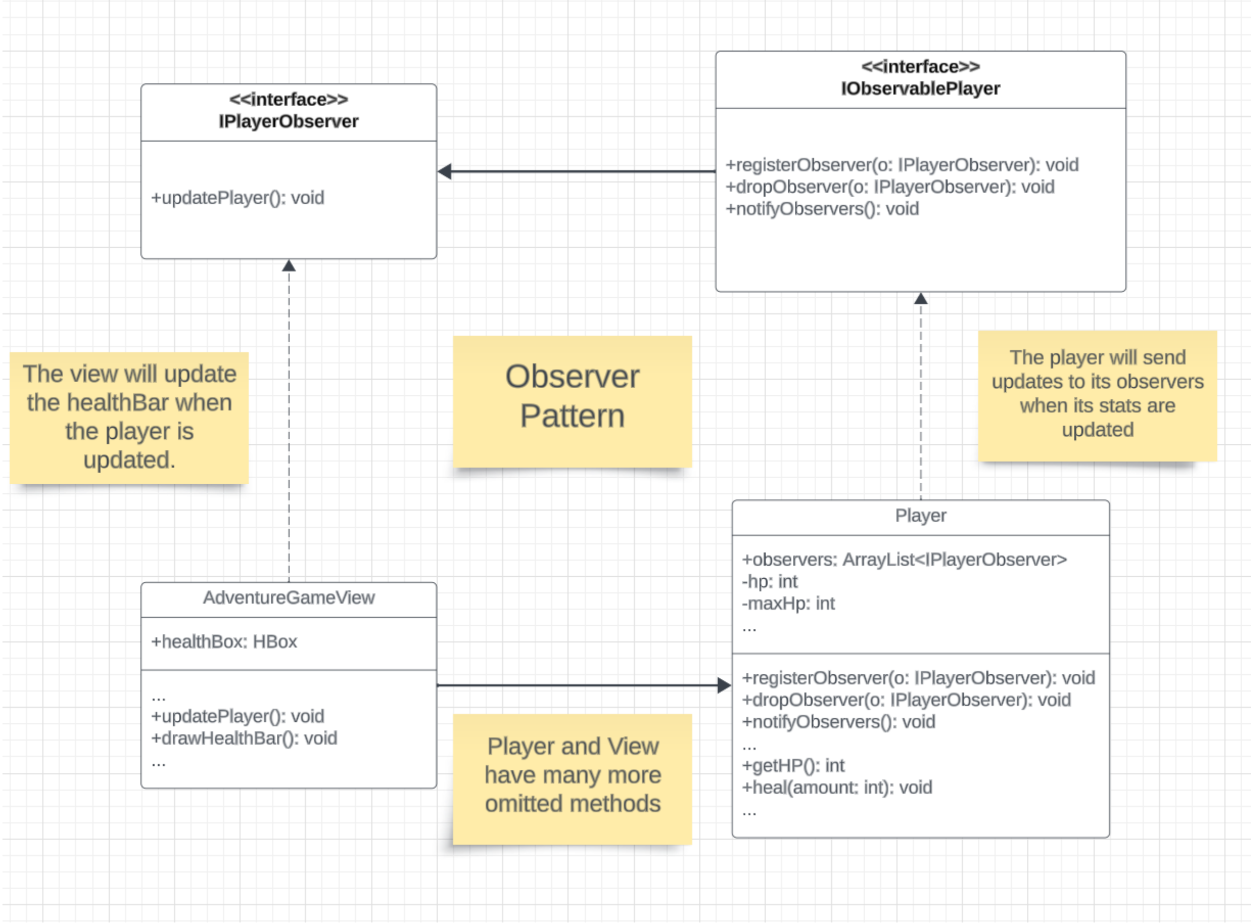
Item Interface: use() will apply the effect of the item into the game. getName() and getDescription() will get data of the item to display in game and for

accessibility reasons. getTexture() will be the sprite used for this specific item.
ItemFactory class: Features createItem(String item, int value) method to return an Item object. It uses a string for item type selection and an integer for attribute values, applicable to different items.

Concrete classes *HealthPotion, Weapon,* and *Key*: Implement the Item interface with distinct attributes. HealthPotion includes a healAmount for its healing effect, Weapon contains attackDamage for inflicted damage, and Key has a keyForDoor attribute for unlocking doors. Each class's use() method is unique to its function in the game.

When a tile in the world is chosen to spawn an item, a random dice will be rolled for which item it has. This will likely be done in a switch statement. An exception is with the Key item, which is integral for story progression, in that case, a key will have to spawn before a locked door in the story so that the player can use the key to progress.
The reason for using the Factory Pattern is that should more items be chosen to be added to the pool of loot in the future, it will allow for very easy expansion to where items already generate in the program.

#2 Health Bar


<<interface>>
IPlayerObserver
+updatePlayer(): void
<<interface>>
IObservablePlayer
+registerObserver(o: IPlayerObserver): void
+dropObserver(o: IPlayerObserver): void
+notifyObservers(): void
The view will update the healthBar when the player is updated.
Observer Pattern
The player will send updates to its observers when its stats are updated
Player
+observers: ArrayList<IPlayerObserver>
-hp: int
-maxHp: int
...
+registerObserver(o: IPlayerObserver): void
+dropObserver(o: IPlayerObserver): void
+notifyObservers(): void
...
+getHP(): int
+heal(amount: int): void
...
AdventureGameView
+healthBox: HBox
...
+updatePlayer(): void
+drawHealthBar(): void
...
Player and View have many more omitted methods

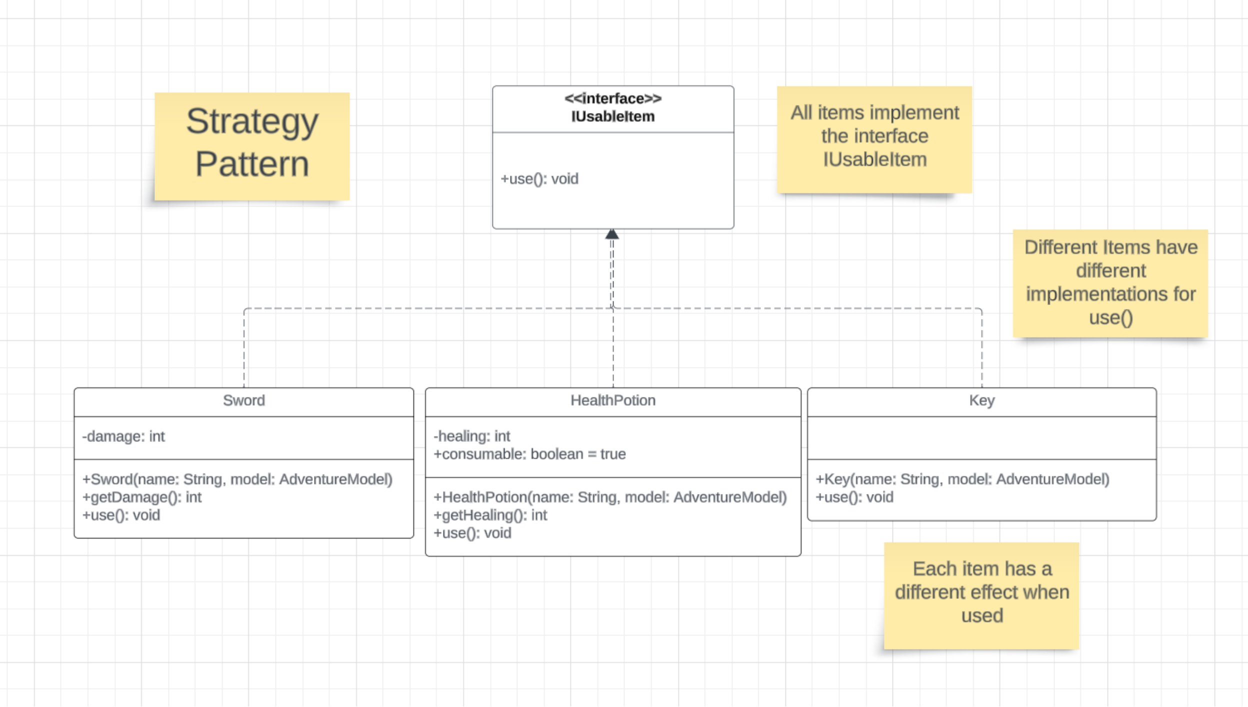
Implementation Details

The AdventureGameView implements the IPlayerObserver, while the Player implements IObservablePlayer.

The View subscribes to the player so that on any health change, then the UI can be updated. When the health of the player changes through the heal() method, then a signal is sent to all of the Player’s observers. Then, the View will respond to the change by drawing the updated amount of hearts in the player’s health bar.

The reason for using the Observer Pattern here is so that the player’s health is visible in real time. When the player gains health from a potion or is damaged by an enemy, then it is immediately reflected in the UI.

#3 Usage of Items



Strategy Pattern
<<interface>>
IUsableItem
+use(): void
All items implement the interface IUsableItem
Different Items have different implementations for use()
Sword
-damage: int
+Sword(name: String, model: AdventureModel)
+getDamage(): int
+use(): void
HealthPotion
-healing: int
+consumable: boolean = true
+HealthPotion(name: String, model: AdventureModel)
+getHealing(): int
+use(): void
Key
+Key(name: String, model: AdventureModel)
+use(): void
Each item has a different effect when used

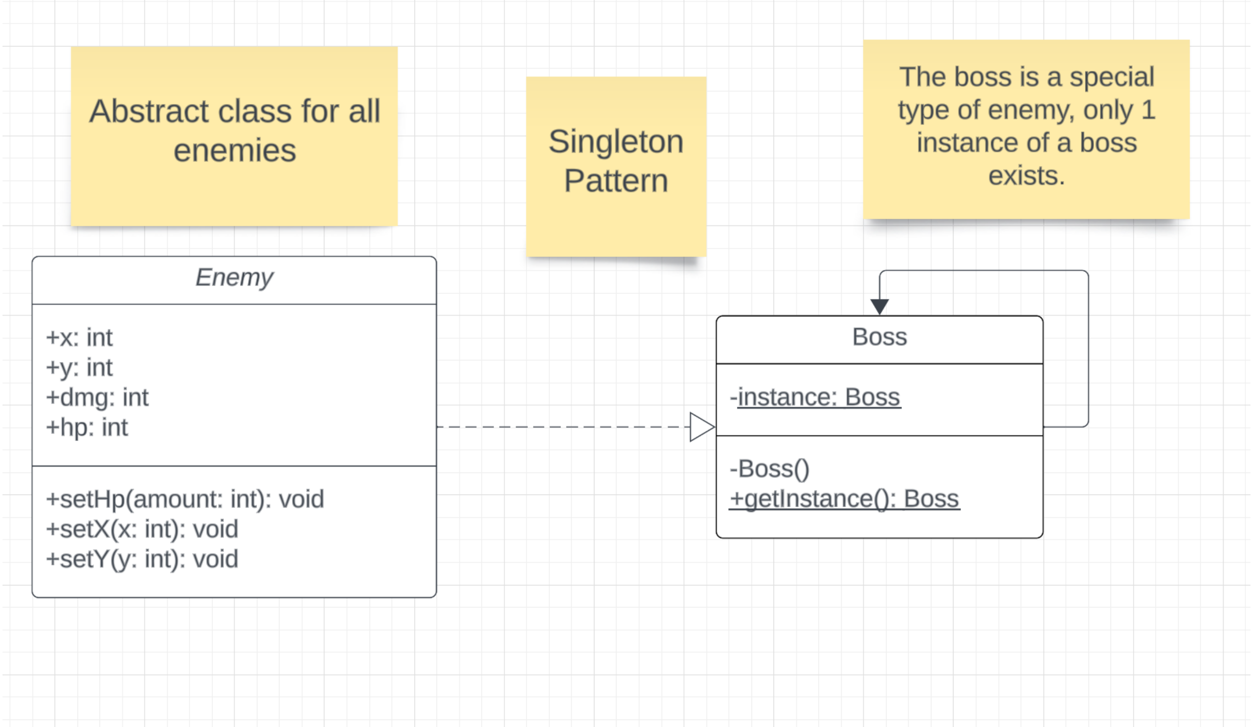
Implementation Details:

Every item implements the IUsableItem interface. Different items have different attributes, for example the sword has a damage attribute and the health potion is consumable and has a healing amount.

Whenever the player uses an inventory item, then that item's respective use() method is called. Each item does something different, but uses the same overridden method use(); the implementations of use() are different depending on the item.

The reason for the Strategy Pattern being used here is so that each item has one method that is called which responds to it being used. Although each item needs a different implementation, there is a common agreement between all items and the function can be called on any inventory item.

#4 Final Boss



Implementation Details:

There is an abstract Enemy class for all enemies in the game (not just the Boss), and also a Boss class. The Boss will inherit attributes like x,y, health, etc. from the abstract class. The Boss has a static method getInstance() that gets the instance of the singleton Boss.

The reason that the Singleton pattern is used here is because the Boss in the game is pivotal for the game’s progression. When the boss is slain, then the game will complete. Thus, there may only be one instance of a boss at any given time.