Design Document (Revised)
Version 1.1 - 2023.12.05
Created 2023.12.04

Halloween Harvest: A Spooktacular Adventure

1. Garry Galsinh
2. Keith Tran
3. Mahmood Saifi
4. Sabrina Hirani

Gitlab Repository: https://mcsscm.utm.utoronto.ca/csc207_20239/group_62

Section #1: Project Identification

Halloween Harvest: A Spooktacular Adventure is a Halloween-themed role-playing game where players hunt for candy in an alternate Halloween dimension. Players navigate a series of neighbourhoods, collecting candy and forging new friendships, before eventually encountering (and hopefully defeating) the candy monster. Our motivation for building this game is to provide an entertaining experience and to spread the spooky Halloween spirit.

Section #2: User Stories

Name	ID	Owner	Description	Acceptance Criteria	Priority	Effort
Load Game	1.0	Sabrina	As a user, I want to be able to start the game with game data loaded correctly, including maps, costumes, NPCs and sprites.	Given that I am a user, when I start the application, the data for the game is loaded from the data directory, which includes map data, dialogues, and sprite images.	1	13
Implementation Details Read JSON game data using `org.json`. Within the Player class, set starting position and starting inventory (friends will always start empty). The map will be a list of paths within class `Neighbourhood` (class `Costume` and class `NPC` will be stored within class `Neighbourhood`, but can be transferred to the `Player` class as the game progresses).						
Handle Key Input	1.1	Sabrina	As a player, I want to be able to direct my character around the map using key input.	Given that I am a player, when I enter key input, then the input is recognized and handled appropriately.	1	1
Implementation Details Implement the `javafx` Key Event Handler. When the user presses an arrow key, the player will move according to the key mapping as follows:						

↑: Up
↓: Down
→ : Left
← : Right

Handle Mouse Input	1.2	Sabrina	As a player, I want to be able to interact with objects in the neighbourhood using mouse input.	Given that I am a player, when I enter mouse input, then the input is recognized and handled appropriately.	1	1

Implementation Details

Implement the `javafx` Mouse Event Handler.

When the user clicks on the costume in the neighbourhood, the costume will be added to their inventory.

When the user clicks on the costume in their inventory, the costume will be equipped.

When the user clicks on the npc, the game will enter combat.

Game Ending	1.3	Sabrina	As a player, I want to be able to trigger an ending event at the end of the game.	Given that I am a player, when I meet certain conditions, such as reaching a certain candy count, friend score, etc., then an ending event is triggered (where I face the candy monster).	8	5

Implementation Details

The player triggers an ending (involving combat with the candy monster) when certain player statistics conditions are satisfied. The specific ending is based on the position of the player, the friend score, the kill count, etc.

View NPC in Neighbourhood	2.0	Sabrina	As a player, I want to be able to see non-player characters situated in different	Given that I am a player, when I am situated in a specific neighbourhood, I can see a list of or images	2	1

			neighbourhoods.	of the NPCs within that neighbourhood.		
Implementation Details The `Neighbourhood` class will store a `HashSet` of `NPC` objects, representing the non-player characters situated in the neighbourhood.						
View Costume in Neighbourhood	2.1	Sabrina	As a player, I want to be able to see costumes situated in different neighbourhoods.	Given that I am a player, when I am situated in a specific neighbourhood, I can see a list of or images of the costumes available in that neighbourhood.	2	1
Implementation Details The `Costume` class will store a `HashSet` of `Costume` objects, representing the costumes situated in the neighbourhood.						
Player Statistics	3.0	Garry	As a player, I want to be able to view information such as kill count, friend score, amount of candy collected, and inventory so that I can keep track of my progress.	Given that I am a player, when I am playing the game, I am able to view relevant statistics such as kill count, friend count, and the amount of candy collected. Given that I am a player, I can view properties of the costume that I have equipped. Given that I am a player, I can view a list of items in my inventory.	5	3
Implementation Details A row of key value pairs (ex: Kills: 0) at the bottom of the GUI that display the relevant statistics to the player.						
Player	3.1	Garry	As a player, I want	Given that I am a	2	3

Interaction with Costume			to be able to interact with costumes.	player, if a costume exists in my inventory or in the neighbourhood, then I can equip the costume.		
Implementation Details The player interacts with costumes by either clicking on a costume in the neighbourhood to add the costume to inventory or clicking on a costume in their inventory to equip the costume.						
Player Inventory Remove	3.2	Garry	As a player, I want to be able to remove items from my inventory when available.	Given that I am a player, when I view my inventory, I can enter a command to drop the item from the inventory.	4	1
Implementation Details If the item is in the player's inventory, add it to the `neighbourhood`'s hashset and remove it from the player's inventory, otherwise nothing is done.						
Player Inventory Add	3.3	Garry	As a player, I want to be able to add items to my inventory by picking them up from the neighbourhood, if available.	Given that I am a player, when I am in a specific neighbourhood, I can enter a command to pick any of the items available in the neighbourhoods.	2	1
Implementation Details If the item is available in the neighbourhood, add the item to the player's inventory, and remove it from the `Neighbourhood`'s hashset. Otherwise, we do nothing.						
Switch Costumes	4.0	Garry	As a user, I want to be able to switch costumes in my inventory.	Given that I am a player, if a costume is already equipped and another exists in my inventory, I can choose to "switch" the equipped costume between	2	1

				them.		
Implementation Details A field in the Player class will store the currently equipped costume, and when the user chooses to switch two costumes, the field is updated to the newly equipped costume.						
Upgradable Costumes	4.1	Garry	As a user, I want to be able to upgrade costumes to receive more/ better boosts by spending candy.	- Given that I am a player, when I have enough candy I can choose to upgrade my costume. - Given that I am a player, when I upgrade a costume, its stats such as attack damage are improved.	7	5
Implementation Details When a costume is clicked, it opens a `javafx` node that allows the player to upgrade the costume. If it is possible to upgrade the costume (meaning that the player has enough costume and the game data indicates that the costume can be upgraded), then update the costume information and reduce the candy count by the upgrade cost. Otherwise, inform the player that they are not able to upgrade the costume.						
Costume Abilities	4.2	Garry	As a player, I want to be able to have different abilities based on the currently equipped costume.	Given that I am a player, when I am wearing a costume, I gain specific abilities and boosts	3	2
Implementation Details After equipping a costume, transfer the abilities of the equipped costume as player boosts and update player statistics. After unequipping a costume, remove the effect of the previously equipped costume. Reset the player statistics to the 'base' level.						
Combat Scene	5.0	Keith	As a player, I want to be able to engage in combat with a NPC.	Given that I am a player, when I decide to engage in combat with a NPC, then I enter into a combat view with the NPC	3	7

				(where I can see my opponent, combat details (i.e. health, attack details, etc.)).		
Implementation Details Implement a `CombatView` class that uses a javafx scene with GridPane for a “combat screen” to facilitate combat between a player and a NPC. Use javafx nodes like labels or shapes to display information such as player HP, NPC HP, etc. on the GridPane.						
Attack NPC	5.1	Keith	As a player, I want to be able to attack an NPC and deal damage.	Given that I am a player, when I press the attack button, then I will deal a specified amount of damage to a NPC that I am fighting.	3	3
Implementation Details We will implement a JavaFX button whose event handler will call an attack method to deal damage to the NPC. The `attack` method checks for info such as NPC stats and Player stats and decides how much HP the NPC should lose. We will also implement a turn tracker to change the turn to the NPC’s and a check to see if the NPC’s health has dropped to zero. If so, we set the `isAlive` attribute to false and end the combat.						
Hit By NPC	5.2	Keith	As a player, I want a NPC to be able to attack me back.	Given that I am a player, when I am fighting an NPC, then I want the NPC to attack me, causing me to lose a specified amount of health.	3	5
Implementation Details This is implemented similarly to 5.1, but with the roles of the NPC and Player reversed. Given the NPC’s turn to attack, an attack method checks for info such as NPC stats and Player stats and decides how much HP the player should lose. We also implement a turn tracker as before, to change the turn to the player’s afterwards. During this we also check if the player’s health has dropped to zero, in which case we set the player’s `isAlive` attribute to false and end the combat scene.						

Combat Result	5.3	Mahmood	As a player, I want to be able to see a screen displaying the result of combat so that I know if I've won or lost the fight.	- Given that I am a player, when the NPC that I am fighting loses all their health, then I get a reward. - Given that I am a player, when I lose all my health, then the game ends and I should see a "Game Over" screen.	3	2
Implementation Details Each character (Player or NPC) has an 'isAlive' attribute. A character or NPC is "defeated" when their HP is non-positive. At this point, 'isAlive' is set to *false*, and the combat is ended when either the NPC or the player is defeated. Based on the player's 'isAlive' attribute, we decide whether to switch to the "Game Over" screen (if the player is not alive), or continue with the game (if the player is alive).						
NPC Info	6.0	Keith	As a developer, I want an NPC to contain a name, a type, position in neighbourhood, etc.	- NPC info and data can be accessed by other methods and classes. - The player can see NPC names. - The player can see dialogue options. - The player can see combat information during combat. - The player can see combat patterns during combat. - The player can see act options.	1	1
Implementation Details The NPC class will contain instance variables for information such as: name, mapPosition, possibleDialogue, etc.						
NPC Attack Patterns	6.1	Keith	As a developer, I want an NPC to have the ability to have different attack patterns.	- During a fight, the NPC can randomly choose between multiple attack patterns (rather than just damaging player HP)	1	5

				- These attacks can affect the player's stats, such as lowering the player's attack damage. - When these attacks are used in a fight, the changes are immediately reflected in the combat view.		
Implementation Details We will add an "attackPatterns" instance variable in the NPC class, which will be a list holding instances of subclasses of the `Attack` class. Each of these subclass attacks in the list is an implementation of an abstract `Attack` class, such that while they are similar in structure, they each affect the player/NPC a tad differently.						
Options View	7.0	Keith	As a player, I want to be able to interact with an NPC	Given that I am a player, when I visit an NPC, I will see a small window of options to choose from.	2	5
Implementation Details Upon a player "visiting" an NPC (by being in the same "neighbourhood" as them), an "interact" button becomes visible, and on click, we will display the interaction options available through the GUI (using the *possibleOptions* array in the NPC class).						
NPC Talk	7.1	Keith	As a player, I want to talk with the NPC	Given that I am a player, when I select to talk with an NPC, I will see a display of a randomly chosen message from a pool of messages the NPC contains.	2	2
Implementation Details Via a button, upon a player trying to "talk" with an NPC, a small label appears in the GUI (representing the NPC talking) with text pulled from a random pool of dialogue options in the possibleDialogue array (in the NPC class).						
NPC Done Talk	7.2	Keith	As a developer, I want the NPC to stop talking back	Given that I am a player, when I select to talk to an NPC too	10	1

				many times, a default message will be displayed instead.		
Implementation Details After the player reaches a threshold amount of “talks”, have a *finishedTalk* boolean that becomes true and the NPC either stops talking back or repeats a “leave me alone” message.						
NPC Act	7.3	Mahmood	As a player, I want to perform specific actions with the NPC to help befriend them.	- Given that I am a player, when I select to act with an NPC, a list of “act” options are displayed for me to choose (give item/start a dialogue/etc.). - Given that I am a player, I want specific interactions to increase or decrease friendliness with the NPC.	1	3
Implementation Details Via a button, upon a user “interacting” with an NPC, a list of more specific interactions with the NPC appears, such as: cracking a joke, saying a rude comment, exchanging items, and so on. This list is implemented as a small popup window containing buttons for each specific interaction. Additionally, each interaction has a “rating” attached to it which will determine how the friendship (stored as an int from 0 to 100) between the player and NPC is affected after the interaction.						
NPC Fight	7.4	Mahmood	As a player, I want to fight with the NPC.	Given that I am a player, when I select to fight with an NPC, I’ll be moved into a combat view with the NPC.	2	2
Implementation Details Invoke the combatView class, passing the selected NPC as a parameter.						

NPC Natural Languag e Dialog	7.5	Sabrina	As a player, I want to be able to interact with an NPC that generates unique natural language dialogue.	Given that I am a player, when I interact with an NPC, then I am able to engage in a natural language conversation with the NPC.	10	13
Implementation Details Use the OpenAI API to generate NPC dialogue.						
GUIThem eChange	8.0	Mahmoo d	As a user, I want to be able to change the theme of the UI between light and dark mode so that the UI is more accessible.	- Given that I am a user, when I open the game's settings, I can see an option to toggle between light and dark mode - Given that I am a user, when I choose to toggle the theme, it changes from light to dark and vice versa	1	3
Implementation Details A "toggle theme" button is displayed on the GUI sidebar. The theme begins in light mode. When the toggle button is pressed, the event handler checks the stylesheets in the root node, and if there is a stylesheet for the dark mode theme, then the "light mode" theme is removed from the root node; otherwise it is added to the root node.						
GUI Alt Text	8.1	Mahmoo d	As a user who relies on assistive technologies like screen readers, I want important image and button elements within the game to have descriptive alt text, so that I can understand the content and purpose of images when using the application.	- Given that I am a user, when I play the game using assistive screen reader technology, I am able to hear descriptive alt text associated with images and buttons.	1	3

Implementation Details Use the `setAccessibleText` and similar methods on JavaFX elements.						
GUI Sidebar Layout	8.2	Mahmood	As a user, I want to be able to access a sidebar next to the game view, which shows options for viewing game information, game settings, and exiting.	- Given that I am a user, when the game window loads, I can see a sidebar next to the game view. - Given that I am a user, I can access a sidebar with multiple buttons for viewing game information, settings, and exiting.	2	5
Implementation Details The scene graph can be constructed with an HBox that contains two nodes: a VBox representing the sidebar, and another node for storing the main game view. The VBox will contain a list of buttons that are shown on the sidebar (such as exiting the application, saving/loading the game, etc.)						
GUI Responsive	8.3	Mahmood	As a user, I want the application window to be responsive, so I can adjust the size of the main window.	- Given that I am a user, when I resize the application, the content of it responds accordingly and resizes as well. - The window should have a minimum size, so that I can't make it too small.	5	5
Implementation Details Make use of JavaFX layout containers that automatically adjust their size, such as HBox, VBox, GridPane, etc. to ensure the app is responsive. Set minimum height and width properties for the Stage node to ensure a reasonable minimum window size.						
Game Shop GUI	9.0	Mahmood	As a user I want to be able to see a "Shop" section in the GUI, with buttons I can click	- Given that I am a user, I can see a section in the sidebar that shows available items to buy from the	7	3

			to buy items.	shop - I should be able to see the name of the item, it's price, and a button to buy the item. - When an item is bought (with candies) it should be added to my inventory.		
Implementation Details Add a VBox for the "shop" section in the left sidebar, inside which is a list of HBoxes containing a label for the item name and price, and a "Buy" button on the side. When the "buy" button is clicked, an event listener is triggered that will handle buying the item.						
Game Shop Buy	9.1	Mahmood	As a player, I want to be able to buy items from the shop in exchange for candies	- Given that I am a user, when I enter a command like "Buy <item>" in the shop command box, I can exchange my candies for the specified item. - I can only buy the item if it is available in the shop, and I have enough candies. - If bought successfully, the item is added to my inventory and removed from the shop's stock.	7	2
Implementation Details When the "buy" button is clicked, an event handler is triggered which will check the user's amount of candies and if there are enough candies to purchase the price of the item, the candies will be deducted from the player and the item will be added to their inventory.						
Game Shop Sell	9.2	Mahmood	As a player, I want to be able to sell items to the shop in exchange for candies	- Given that I am a user, when I enter a command like "Sell <item>" in the shop command box, I can exchange my candies for the specified item, if it is available in my	7	2

				player inventory.		
Implementation Details Inside the inventory list, there will be a button to “Sell” items. When it is clicked, an event handler is triggered which will remove the item from the inventory, and add a certain number of candies to the user based on the original price of the item.						
Potion Item	9.3	Mahmood	As a player, I want to be able to buy potions from the in-game shop, which I can consume to positively affect player stats like health.	- Given that I am a user, I can choose to consume a potion item if it is available in my inventory. - When consumed, the potion will have a positive effect on my character, depending on the type of the potion (health, attack, etc.)	7	2
Implementation Details A potion will be an instance of an item class, which will store attributes like the name of the item and methods like what effects the item will have when consumed. When the potion is bought, it will be stored in the player’s inventory, and can be “consumed” at any time the player wishes. When consumed, the potion will have positive effects on the player’s stats, like increased health. Once consumed, the potion is removed from the player’s inventory.						
Invisibility Cloak Item	9.4	Mahmood	As a player, I want to be able to buy the “invisibility cloak” item from the in-game shop, which I can consume to escape NPC fights.	- Given that I am a user, I can buy an invisibility cloak from the in-game shop. - The cloak can only be consumed during an NPC fight. - When consumed, the cloak will allow the player to “escape” the NPC fight early. - Depending on the difficulty of the fight, the effect of the cloak may fail (for example during boss fights).	7	2

Implementation Details

When consumed, the invisibility cloak item will check if the user is in a combat state. If not, the item will not be consumed.

If the item is consumed, the difficulty of the opponent is checked (each opponent will be assigned a specific difficulty).

We will use randomness, biased according to the difficulty, to determine if the cloak effect succeeds or fails.

If it fails, the player continues the fight. The item is removed from the inventory.

If it succeeds, the combat ends early. The item is removed from the inventory.

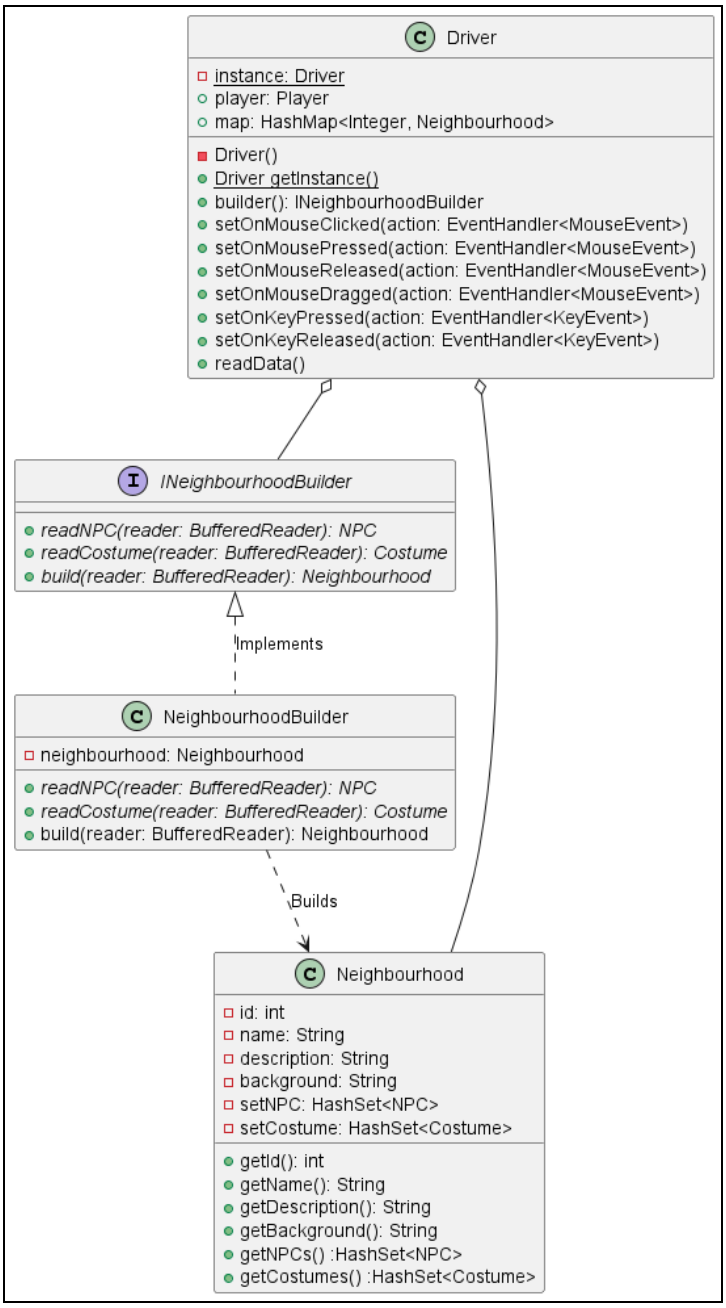
* We select user story 1.2 (the simplest user story) as a base user story for generating estimates. All other user stories are estimated relative to the base user story in terms of effort and complexity.

Section #3: Software Design

Builder Design Pattern

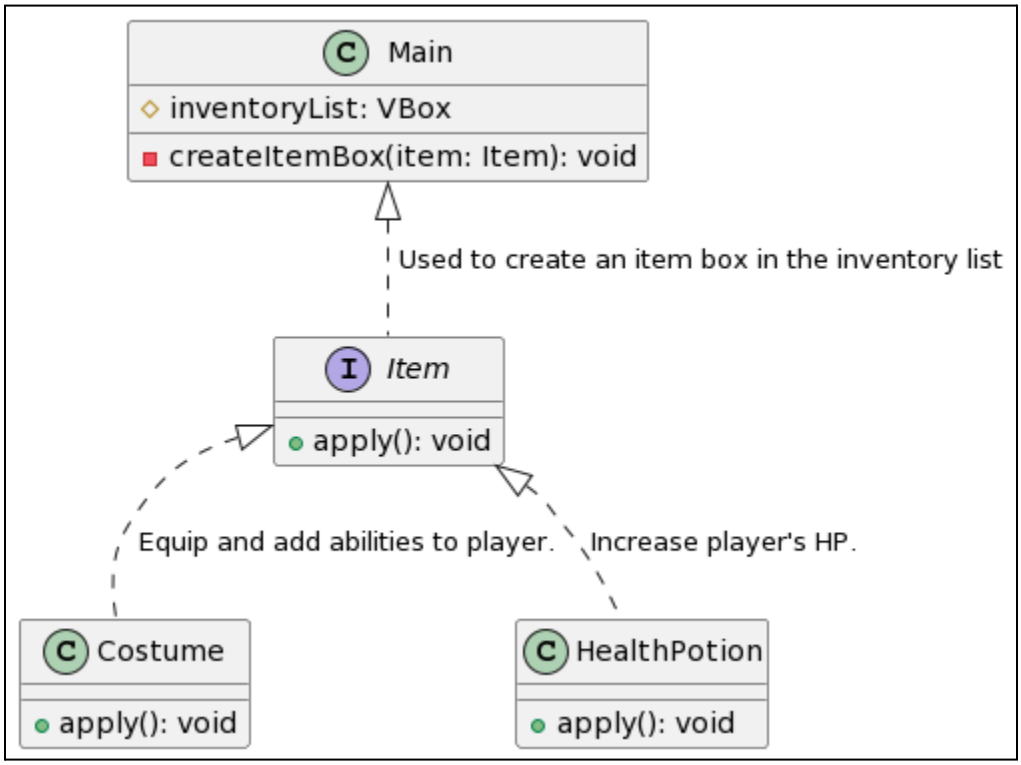
This diagram represents the builder design pattern, which is used to construct complex objects step by step. In this diagram, the pattern is applied to the construction of the `Neighbourhood` object. This construction is facilitated by the `NeighbourhoodBuilder` class which implements the `INeighbourhoodBuilder` interface to separate (for the sake of simplicity and readability) the logic of constructing a `NPC` from the logic of constructing a `Costume` from the logic of constructing a `Neighbourhood`.

The `NeighbourhoodBuilder` class contains methods designed to construct smaller objects that are stored within the more complicated `Neighbourhood` object. The `parseNeighbourhood` method constructs a `Neighbourhood` object from the game data and handles the parsing of smaller objects stored within the `Neighbourhood` object through methods such as `parseCostume`, `parseNPC`, `parseAttack`, and `parseItem` as required. This systematic process employs a top-down approach to progressively and methodically assemble `Neighbourhood` objects.


Driver
instance: Driver
player: Player
map: HashMap<Integer, Neighbourhood>
Driver()
Driver getInstance()
builder(): INeighbourhoodBuilder
setOnMouseClicked(action: EventHandler<MouseEvent>)
setOnMousePressed(action: EventHandler<MouseEvent>)
setOnMouseReleased(action: EventHandler<MouseEvent>)
setOnMouseDragged(action: EventHandler<MouseEvent>)
setOnKeyPressed(action: EventHandler<KeyEvent>)
setOnKeyReleased(action: EventHandler<KeyEvent>)
readData()
INeighbourhoodBuilder
readNPC(reader: BufferedReader): NPC
readCostume(reader: BufferedReader): Costume
build(reader: BufferedReader): Neighbourhood
Implements
NeighbourhoodBuilder
neighbourhood: Neighbourhood
readNPC(reader: BufferedReader): NPC
readCostume(reader: BufferedReader): Costume
build(reader: BufferedReader): Neighbourhood
Builds
Neighbourhood
id: int
name: String
description: String
background: String
setNPC: HashSet<NPC>
setCostume: HashSet<Costume>
getId(): int
getName(): String
getDescription(): String
getBackground(): String
getNPCs() :HashSet<NPC>
getCostumes() :HashSet<Costume>

Strategy Design Pattern

The UML diagram below represents the Strategy design pattern as used in our game, when integrating items (such as potions and costumes) into the GUI. Inside the GUI, we create “item boxes”, which are essentially HBoxes containing Label and Image elements for showing item or costume details, along with a “Use” button to use the item, or in the case of costumes, equip them. However, each item box may contain a different type of item, each of which will have its own effects on the player and game state. Implementing all of this logic inside the GUI controller would violate the Single Responsibility principle and also make for some very messy code. We use the strategy design pattern here, where each item will have its own implementation of the `apply` method, that specifies the strategy used to apply the effects of that item. This behaviour is abstracted into the `Item` interface. Then, this `apply` method is called when handling the click event for the “Use” button in the item boxes, within the GUI.

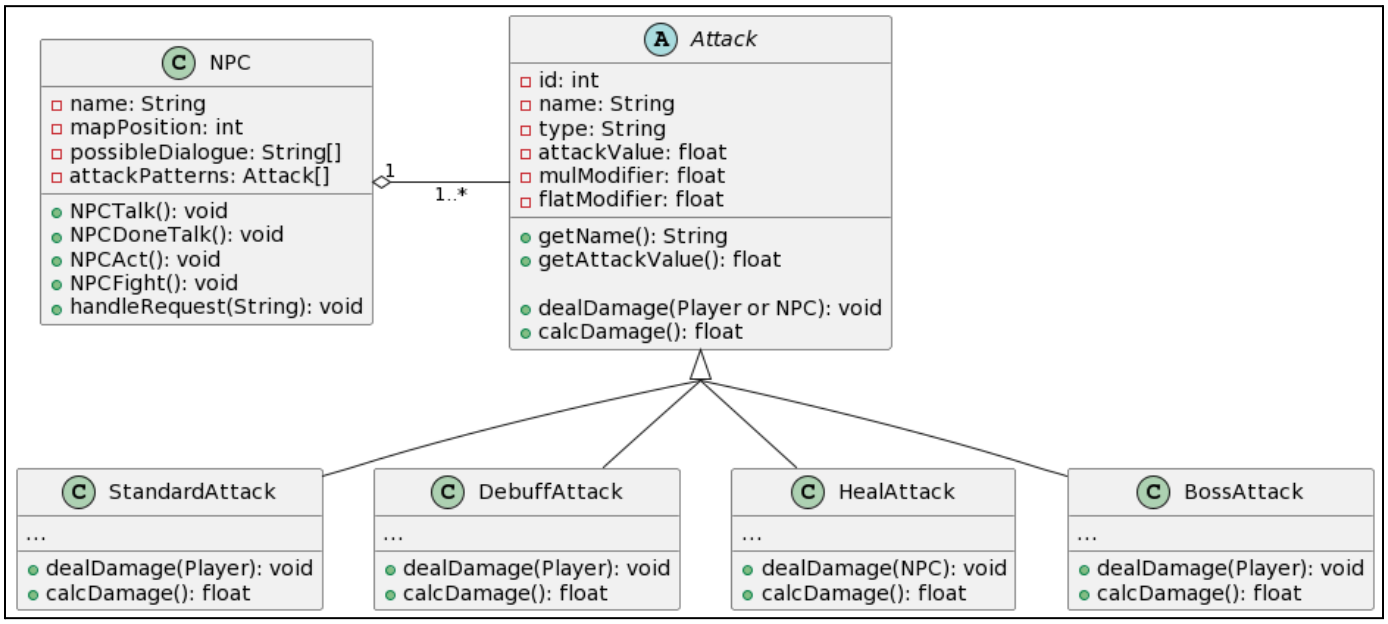


Template Method Design Pattern

In order to make combat more interesting, NPC's are designed to be able to have multiple "attacks" that are stored in "attackPatterns". The thought is that attacks should be different enough so that it encourages Player strategy, but not so different that combat is silly and outrageous. Thus, an NPC's "attacks" are designed with the *Template Method* design pattern, which can be seen below. The *Attack* class is a class with abstract methods: **dealDamage** and **calcDamage** and serves as a template for "attacks". In essence, *calcDamage* calculates how damage an attack should do based on factors such as *mulModifer* and *flat Modifier*, and *dealDamage* takes the calculated value and applies it to (or "damages") a Player or NPC.

The *subclasses* (StandardAttack, DebuffAttack, etc.) implement the *Attack* class by overriding the abstract methods, while retaining instance variables and non-abstract methods. Each subclass may override and implement the abstract differently. For example: the **StandardAttack** subclass will simply calculate damage based on the modifiers and subtract the damage from the player's health; the **DebuffAttack** will be of the same vein but *dealDamage* will also apply a debuff; the **HealAttack** will instead heal the NPC on a *dealDamage* call instead of the damaging the player; and so on and so forth.

And thus, each *Attack* subclass is different, yet follow a common *template*.



Singleton Design Pattern

The Singleton design pattern is used here to ensure that there is only one instance of the `Driver` class in the application. This is important because the `Driver` class appears to serve as a central controller for input events from the GUI. Having a single instance ensures that event listeners and input handling are coordinated consistently and avoids the potential issues that can arise from multiple instances of the `Driver` class.

To implement this pattern in the `Driver` class, we marked every variable and method in the `Driver` class as static, including the Player instance and map data, and made the constructor private. This ensured the Driver class could not be instantiated, and only the singleton Driver class object was available globally for access.