CSC207 PROJECT PHASE 1 DESIGN DOCUMENT

Design & User Stories Document

Version 1.0 – 2023.11.04

Created 2023.11.04

Project Name: Touch Grass

Moeez Omair
Adam Osama Ebid
Youssef Samaan
Nikhil Iyer

GitLab Repository:

https://mcsscm.utm.utoronto.ca/csc207_20239/group_76

SECTION 1: PROJECT IDENTIFICATION

- Our goal is to gain experience of collaborating in a group towards a common goal. Additionally, we want to show off and improve our JavaFX skills, and make a game that we find enjoyable. We all enjoy Tycoon style games, so we decided to make our game a Tycoon.
- Our strategy is to take some of the key functions from Adventure Game, and add our own functionally and GUI on top of it. It will essentially be a new game, but still have some of the core mechanics that Adventure Game had.

SECTION 2: USER STORIES

Name	ID	Owner	Description	Acceptance Criteria	Implementation Details	Priority	Effort

Game grid	1.1	Nikhil	As a player, I want a visual interactive game grid that updates real-time with each of my moves, allowing me to strategize my next actions quickly and effectively.	Given that I am a player engaged in the game, when I perform any action, such as making a move, planning a strategy, or interacting with game elements like grass or poison ivy, then the game grid should immediately update to reflect my action, allow for strategic planning, and show visual changes corresponding with the specific gameplay activities I engaged in.	-Develop a 9x9 grid mechanics with responsive tile states. -Ensure grid updates are visually indicated.	1	3
Move Player	1.2	Adam	As a player, I need the ability to move to different tiles so that I can progress through the game	Given that I am a player who has selected to move using keyboard input, when I input a command to change my position, then the model updates my current position, the player sprite moves to the new tile, and the scene updates to reflect this new status.	-Call the MoveTable class that describes all the current moves (up to 4 cardinal directions) -If a valid keypress is detected given the MoveTable from that tile, the currentTile will be updated to the new one	2	3
Cut Long Grass	1.3	Youssef	As a player, I want to cut long grass automatically when I move onto a spot containing long grass, so that I earn money that I can spend.	Given I am a player with the ability to move, when I move onto a spot containing grass, then the grass should be automatically cut, and I should earn money for each long grass cut.	-Integrate automatic grass cutting when player is on a tile that contains long grass -Update UI to reflect currency changes. -Program currency accrual per grass tile cut.	3	2
Random long grass Regeneration	1.4	Nik working on this	As a player, I want the grass to regenerate after a random amount of moves so that I can continue to earn more money.	Given that the player has cut the grass, when a random amount of moves have occured in game, then the grass will begin to regenerate and they can cut it again.	-If the grass is cut, then a random integer will be generated, and a counter will be initialized at 0. -After every player movement, the counter will be incremented, and when it is equal to the	4	2

					random number value, the grass will regrow. -The change will be reflected in UI, and the scene will be updated to display the new grown grass tile.		
Random Poison Ivy Generation	2.5	Moeez on standby and waiting for Nik	As a player, I want random poison ivy generation throughout the game, so that I have to come up with new strategies which will make the game more interesting and exciting.	Given that the player is actively exploring the game, when the player moves through the environment, then the poison ivy tiles should be randomly generated within the game grid at various locations. The poison ivy generation should ensure a reasonable challenge without overwhelming the player, and the poison ivy tiles should be visually distinguishable from the regular grass.	-Poison Ivy will spawn as another sprite instead of grass (but 1/10 chance) -After the tile cooldown is complete, there can be a 1/10 chance for the grass to generate as poison ivy instead	5	2
Player HP	2.6	Youssef	As a player, I need a certain amount of health points so that I can strategically manage my decisions and actions, rewarding me to succeed in the game, or punished with losing otherwise.	Given that the player is actively engaging in the game, when the player purchases a health potion from the item store, then they should see an HP status indicator that changes based on the players health conditions. If the players health decreases the HP status indicator should visibly decrease. If the players HP reaches zero, the game should end and show a game-over screen.	-Develop visual representation of HP -Implement logic to increase HP -Implement logic to decrease HP -Set game conditions when HP reaches zero	6	2
Poisoned State for Characters	2.7	Adam	As a player, I want my character to enter a poisoned state when they come into	Given I am a player with a character at risk of environmental hazards, when my character is poisoned by contact with poison ivy,	-Implement distinctive visual effects on characters (such as a greenish tinge or a bubbling effect) to	7	2

			contact with poison ivy, so that the game is more challenging and requires me to avoid or mitigate the effects of hazards in the environment.	then they enter a "Poisoned" state with visual indicators, suffer health loss over time, and experience gameplay changes until the condition is cured or wears off, with immediate feedback provided.	indicate the "Poisoned" state. -Create a health reduction mechanism that triggers a steady decrease in the character's health once they come into contact with poison ivy. -Design and integrate an alert system that combines visual cues (like a status icon) and audio signals (such as a distinct sound effect) to provide immediate feedback to the player upon entering the poisoned state. -Develop a system for the poisoned state to impact gameplay by taking 1 hp of damage every second for 5 seconds -Ensure there is a method for the poisoned state to be cured, either over time or through player action, such as using an antidote item.		
Multiple plant sprites	2.8	Nikhil	As a player, I want the different foliage types like cute grass, long grass and poison ivy to look graphically appealing, so that I can distinguish between them.	Given a game environment with a single grid displayed from a bird's eye view, when the player explores this map, then they should encounter multiple plant sprites scattered throughout, each representing different plant species and contributing to the visual richness and diversity of the in-game world.	Implement different models for different plant types.	8	2

Save Game	2.9	Moeez (working on this right now)	As a player, I want to be able to save my game progress so that I can continue my gaming experience from where I left off later.	Given the player is actively engaged in the game and has made progress, when they initiate the save game function, then their current game state should be saved, allowing them to continue their gaming experience from the same point later on or recover their progress if the game is closed or restarted.	Develop a save game function that serializes the current game state and writes it to a designated file, allowing players to resume their progress at a later time by deserializing the data.	9	1
Load Game	2.10	Moeez (working on this right now)	As a player, I want to be able to load a saved game so that I can resume my gaming experience from a previously saved point. So that I can revisit and continue my adventures, ensuring that I don't lose any of the progress I've made in the game.	Given that the player wants to continue a previously saved game, when they initiate the load game function, then the game should retrieve and load the saved game state, allowing the player to resume their gaming experience from the exact point where they left off, preserving their progress and achievements.	Implement a load game feature that reads and deserializes the saved game state from a designated file, restoring the player's progress and game state to the point where they previously saved.	10	1
Gaining money	3.11	Youssef	As a player, I want some incentive to cut long grass, so I would like to be able to earn money to buy other in-game products.	Given the player goes on a tile that contains long grass, when the action is successful, then their in-game currency should increase and the new total should be displayed, allowing them to track their financial progress.	-Develop in-game economy logic to reward players with currency for cutting long grass. -Introduce tool-based modifiers affecting the amount of currency gained from cutting long grass, dependent on the tool in the inventory.	11	3
On screen money display	3.12	Adam	As a player, I want to keep track of my finances and	Given that the player is actively engaging in the game, when the player is progressing through	-Implement a label that stays on screen and updates when user gains or loses money.	12	1

			make sure I can always see how much money I have so that I know how much I need to buy items from the shop.	the game and earning money, then the player should have a real-time, visible representation of their wealth, readily available on the game interface.			
Player Inventory	3.13	Moeez	As a player, I want to view my inventory of items so that I can manage and keep track of the items I've collected during the game.	Given the player is in the game and has accessed their inventory screen, when they view the inventory, then they should see a list of items with names and descriptions.	Implement a javafx GUI that displays all player's current items with their respective names and descriptions.	13	2
Item shop	3.14	Youssef	As a player, I want to press a button that will open a virtual in-game shop so that I can buy items using in-game currency to enhance my gameplay.	Given I am a player with in-game currency and the item shop is accessible, when I press the designated button to open the shop, select an item, and complete the purchase, then the shop should display with items and their details clearly described, and my inventory should update to include the purchased item with the in-game currency adjusted accordingly.	-Design and integrate an item shop button in the main game interface. -Create UI for the item shop with a clear layout of items, descriptions, and prices. -Program the functionality for currency exchange and inventory updates.	14	2
Tool upgrade system	3.15	Adam	As a player, I need different tools that will make cutting grass more profitable, so that I can purchase items from the item store faster.	Given that I have access to different tools in my inventory, when I cut grass with these tools in my inventory, then my earnings should be greater compared to using basic tools, which will allow me to purchase items from the store faster."	-Design a variety of tools with increasing earning potential. -Implement tool-specific bonuses for cutting grass. -Adjust in-game economy to reflect the increased earnings from upgraded tools.	15	3

					-Integrate visual and functional feedback in the UI when using different tools.		
Player time	3.16	Moeez	As a competitive player, I need to see how much time I've spent on each level, so I can work on beating my own best times.	Given I am playing the game, when the level starts, the system should begin time tracking, and it should stop when the game ends, so that I can view my total completion time post-game to work on beating my own best times.	-Develop and integrate a timer mechanism within the game logic that activates upon game commencement and deactivates upon game completion. -Ensure the timer accuracy aligns with the game's tick rate for precise measurement. -Design a user interface component to display the elapsed time in real-time during gameplay for player reference.	16	2
End of Game Stats	4.17	Nikhil	As a player, I want to view a summary of my game stats when I complete a game, so that I see my achievements and areas for improvement.	Given that the player has successfully completed the game, when the game concludes, then a stats summary screen should be displayed showing items picked up in the game, amount of money collected, time taken to complete the game, total HP lost.	-Create an end-of-game trigger that brings up the stats screen. -Collect and display relevant game statistics. -Design UI for clarity and ease of understanding.	17	1
Game Audio	4.18	Nikhil	As a casual game enthusiast player, I want to have an engaging and fitting background music along with responsive game audio, so that it keeps me stimulated and interested as I navigate	Given a player starts a new game or continues a game session, when they navigate through the game and interact with specific scenes/objects (i,e. Main game grid, shop, player inventory, game sprites etc..), then the appropriate background music and game audio snippets should be played to engage the player.	-Initialize a JavaFX MediaPlayer for managing background music, ensuring it plays the right sound for the right scene and can be controlled by the player via in-game settings. -Initialize a JavaFX MediaPlayer for managing the dynamic playback of audio snippets tied to in-game actions, ensuring a responsive and	18	2

			through the game.		immersive experience for the player.		
Text-to-Speech	4.19	Moeez	As a visually impaired user, I need audio descriptions of onscreen text, the current menu displayed, and objects in the room, so that I can understand my onscreen environment effectively.	Given that a visually impaired user is using the application, when they navigate to a screen with onscreen text (e.g., instructions, labels, notifications, menus), Then the application should provide their descriptive audio descriptions of the onscreen text using text-to-speech technology.	Create an attribute associated with every object description, using java.accessibility library to incorporate screenreader hooks to activate when called.	19	2
Color blind Filters	4.20	Youssef	As a user with color blindness, I want to apply a color-blind filter to the application's text, so that I can perceive and distinguish different colors more accurately.	Given the user enables a specific colorblind filter, when they view the applications text, then the text colors should be adjusted to make them more distinguishable to the user, by applying the appropriate color transformation.	Transform and apply all of the text colors within the application to the desired color.	20	3
Keyboard only functionality	4.21	Nikhil	As a user using keyboard only, I want to navigate and interact with the application using just the keyboard, so that I can efficiently progress through the game without relying on a mouse or other pointing device, enhancing my	Given the user is navigating and interacting through the application using the keyboard, when they press the Tab key or input text using Enter key, then the focus should move logically through the interface elements in a consistent and intuitive order and be able to input text into the application, respectively.	Implement a keyboard event handler for the Tab key and Enter key. The Tab key event handler should focus the next interactable element. The Enter key event handler should input text into the TextField.	21	1

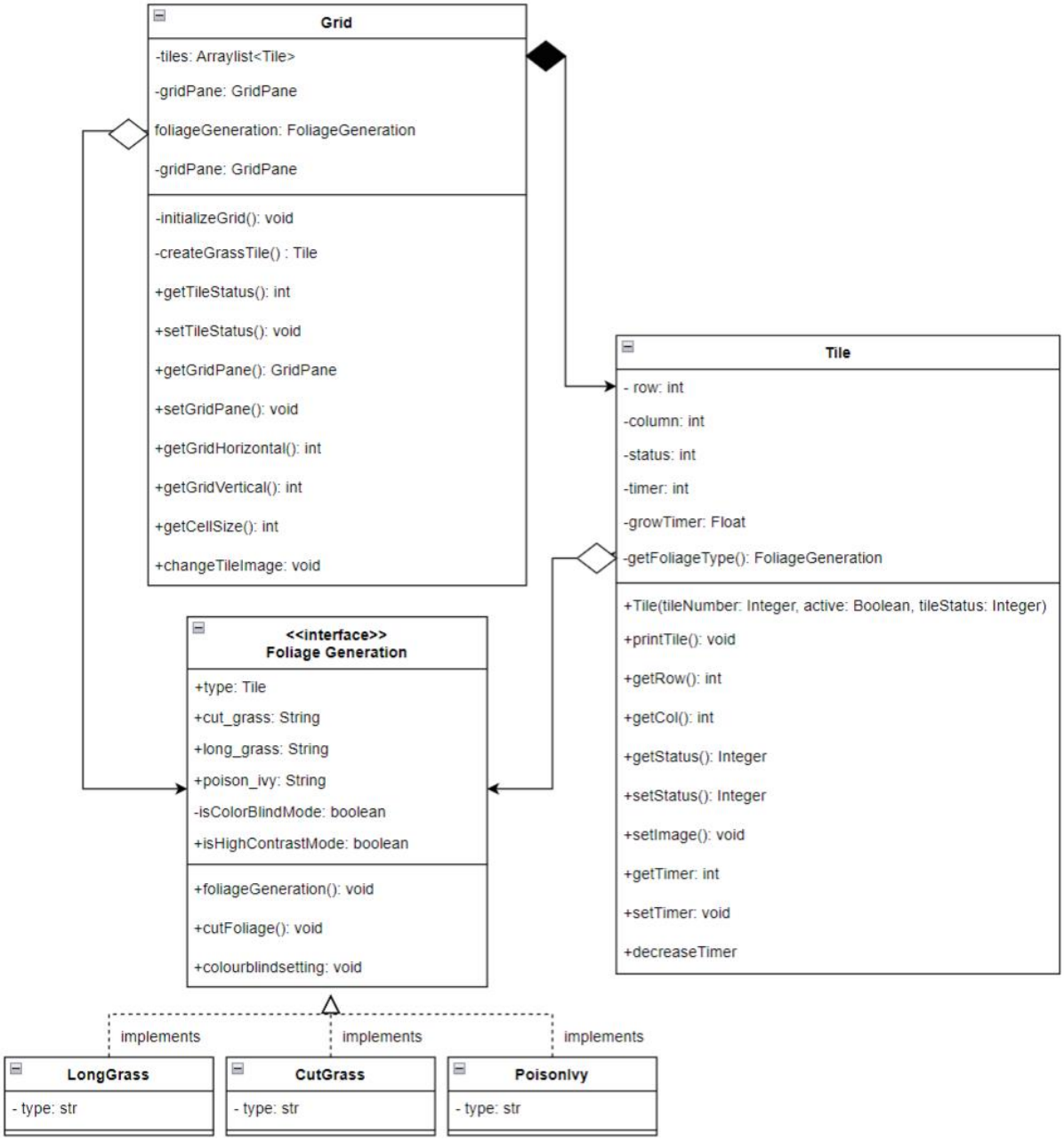
			accessibility and productivity.				
Treasure progress bar	4.22	Adam	As a player, I want to be able to see how much progress I have made, so that I know if I am close to completion or not	Given that the player is on a quest or exploring the game world, when they collect treasure or valuable items, then they should see a treasure progress bar that fills up as they accumulate these treasures, giving them a visual indication of their progress in amassing valuable items or wealth within the game.	Create a progress bar widget that remains visible on the screen and updates in real-time as the player collects or loses treasure items.	22	2

SECTION 3: SOFTWARE DESIGN

Design Pattern #1: Builder and Decorator

UML

Moeez – Builder and Decorator



Implementation Details

Revising my original UML, I have realized that a “builder” and “decorator” patterns reflect my design choices best after implementing everything. The UML diagram above explains the implementation of the grid’s appearance.

Grid:

The Grid class is responsible for grid where all Tile objects are presented, collected within an ArrayList. There will be an active tile attribute and a method to extract it.

Tile:

The Tile class represents each individual tile within the grid. The tile class has multiple attributes related to its implementation as well as methods to extract it. There are also additional methods required internally to keep the method functional. FoliageGeneration is also used to further make this function complete.

Foliage Generation:

The FoliageGeneration interface provides a contract for different types of foliage that can decorate a tile, such as long grass, cut grass, and poison ivy. Moreover, when implementing the accessibility features, this allows us to apply the filters to each of the tile’s photos with ease.

LongGrass, CutGrass, PoisonIvy:

These classes implement the Foliage interface, each representing a different type of foliage that can exist on a tile. Each of these have unique properties, such that:

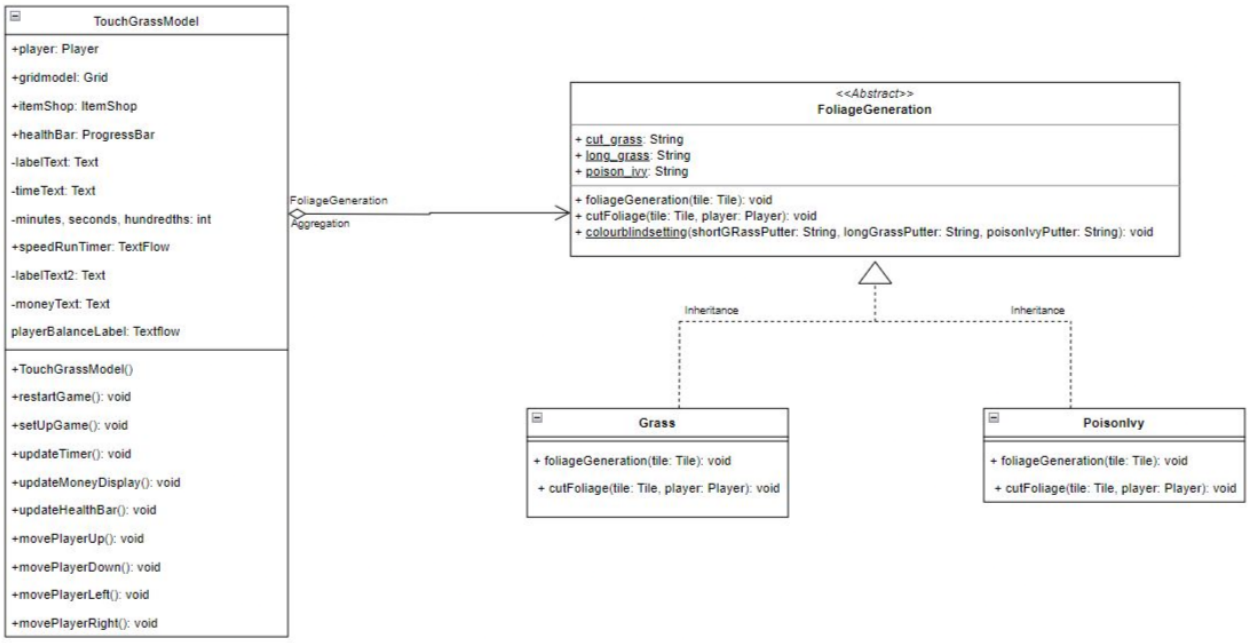
- LongGrass can be cut by the player and provides them money.
- CutGrass replaces LongGrass when it is cut, it also provides no money.
- PoisonIvy is unique such that it provides the largest sum of money.

These are further developed in TouchGrassView where we replace the sprites for other assessibility features.

Design Pattern #2: Strategy

UML

Nik - Strategy



Implementation Details:

The UML diagram uses the 'Strategy' design pattern and outlines these main components:

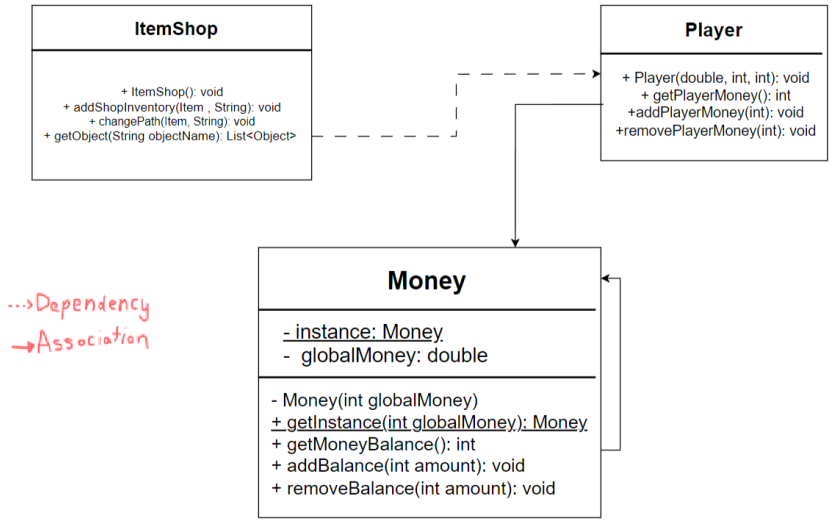
- Grass and PoisonIvy classes that inherit from the FoliageGeneration abstract class.
- The TouchGrassModel to switch between different FoliageGeneration strategies at runtime.
- The FoliageGeneration abstract class with a foliageGeneration() method and cutFoliage() method to be overridden by different generation strategies, depending on the type of foliage.

The Grass foliageGeneration() method takes in a Tile as a parameter, which ensures that the method accurately updates the correct tile with the new type of foliage. Similarly, the PoisonIvy foliageGeneration() method takes in a Tile as a parameter so that if poison ivy needs to be generated on a tile, it can accurately do so. Both methods and classes inherit from the FoliageGeneration abstract class and override the foliageGeneration() method when needed. The cutfFoliage() method is also an abstract method that gets overridden by both foliage types. For both Grass and Poison Ivy, it takes in a Tile, and Player object, and updates the status of the tile to be "cut". The FoliageGeneration abstract class receives its context from the TouchGrassModel, which holds all relevant data, including the gridmodel, which holds all the tiles, with each tile being a Hashmap with the keys being Integers, and the values being Tile objects.

Design Pattern #3: Singleton

UML

Youssef - Singleton



Implementation Details:

Here are the implementation details for this diagram:

This UML diagram utilizes the Singleton design pattern for managing a global currency within the game. The Singleton pattern ensures that only a single instance of the Money class is created which provides a consistent point of reference for the parts of the program – the Player and Shop - that interact with the game's currency. The Player and Shop classes depend on the Money class for all transactions, maintaining data integrity and consistency. The single instance of Money prevents conflicts that could arise from having multiple instances of a global currency in different parts of the game.

Money Class:

- instance: A private static attribute that holds the single instance of the Money class.
- globalMoney: A private instance attribute that stores the total amount of money (of type double) available globally within the game.
- Money(): A private constructor that initializes the Money instance with the provided amount of money. It is private to prevent direct instantiation from outside the class.
- getInstance(): A public static method that returns the single instance of the Money class. Within the code implementation, If the instance does not exist, it is then created with the provided initial global money amount. Otherwise, the initial money amount is ignored, and the existing instance is returned.

- getMoneyBalance(): A public method that allows getting the global money balance.
- addBalance(): A public method that allows adding to the global money balance.
- removeBalance(): A public method that allows subtracting from the global money balance.

Interactions with Other Classes:

Player Class:

- A class containing methods such as the constructor, Player(), getPlayerMoney(), addPlayerMoney(), and removePlayerMoney() that interact with the Money class, allowing players to query their balance and add to their funds.

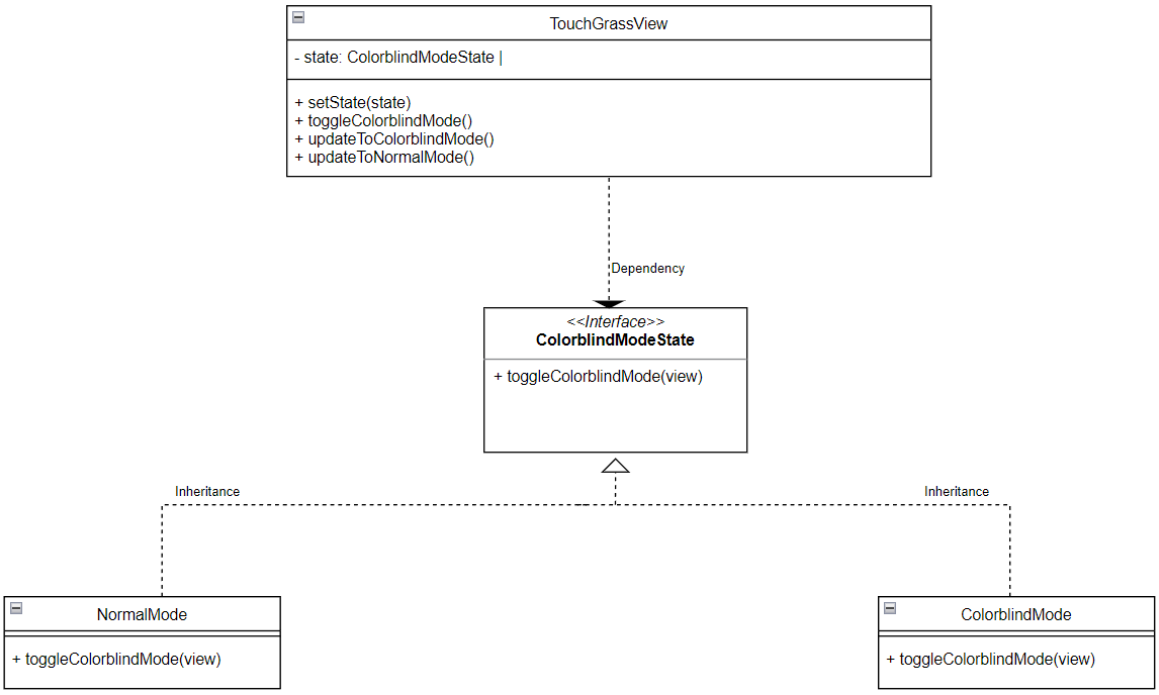
Shop Class:

- A class Contains methods like getMoney() and removeBalance() that directly interact with the player class and indirectly interact with the Money class, facilitating transactions where players can spend their money.

Design Pattern #4: State

UML

Adam -



Implementation Details:

The ColorblindModeState interface is pivotal in abstracting the state behavior in the TouchGrassView context. It declares a method to toggle the colorblind mode, toggleColorblindMode(TouchGrassView view).

Concrete states, NormalMode and ColorblindMode, implement this interface. They encapsulate the specific adjustments needed for their respective modes, such as modifying UI elements for color visibility.

TouchGrassView maintains a reference to the current ColorblindModeState. It delegates the action of switching modes to the current state's toggleColorblindMode method. It also provides updateToColorblindMode() and updateToNormalMode() methods for actual UI changes.

The UML diagram reflects a dependency from TouchGrassView to ColorblindModeState,

signifying that TouchGrassView relies on this interface for managing state-related behavior. The inheritance from ColorblindModeState to NormalMode and ColorblindMode shows these concrete states are specific implementations of the interface.

This setup makes the application's state management flexible and extendable, allowing for future enhancements with minimal impact on existing code.