CSC207 LEC0103

Design Document

Version 1.0; Created October 20th 2023
Version 2.0 (current); Created December 5th 2023

Hia Aggrawal
Kanupreet Arora
Mahak Mishra
Dhvani Patel

Project Name: Code Chronicles: Wizard's Quest
GitLab Repository: https://mcsscm.utm.utoronto.ca/csc207_20239/group_99

Section 1: Project Identification

Game Synopsis:

You are a coding apprentice who has recently been admitted to Hackwards School of Codecraft. However, Incognito Phantom, the evil code crafter has sent his (shapeshifting) Polymorphic Prowlers to ensure that all talented codecrafters at Hackwards School of Codecraft perish! You must defeat the Polymorphic Prowlers by putting together clues, fighting virus monsters and collecting code bytes, all while navigating your new school.

Code Chronicles: Wizard's Quest aims to make the learning process more engaging for programmers by combining elements of education, gaming and creativity. The game reinforces programming concepts such as polymorphism, object-oriented-programming and inheritance while simultaneously providing a compelling narrative for players.

While existing educational resources may offer coding tutorials, they tend to lack the creative and immersive aspects of gaming. However, this game will do just that. It will enhance the functionality of such resources by reinforcing programming concepts in a fun, hands-on way. In addition, it will provide additional accessibility features such as audio and visual aid (i.e. color contrast, font size, visibility), which enhances the functionality for all users. Finally, this game appeals to a wider range of programmers, which enhances the functionality of pre-existing resources that may only be aimed towards a specific target audience.

Overall, by combining education with gaming, it provides a holistic experience for programmers and gamers alike.

Section 2: User Stories

For the following user stories, the effort for each user story ranges from 1 to 10; 1 being the lowest and 10 being the highest. Similarly, the priority is ranked on a scale of most to least; 1 being the most and 8 being the least.

User Stories

Name	ID	Owner	Description	Implementation Details	Priority	Effort
Character Customization	1.1	Mahak	As a general user, I would like to have the ability to customize my mage (i.e. appearance, disabilities) so that I feel represented and included.	At the start of the game, allow the user to choose their appearance from a pre-programmed list of playable characters. The super class is "Character" and each type of Character object is a subclass with distinct features. (ie. AlchemistCharacter, WarriorCharacter).	5	4
Menu	1.2	Kanupreet	As a user who needs more information on how to play, I need the option to see the instructions, restart, and exit the game.	Add a pop-up with instructions on how to play, restart, exit the game. This will be implemented as a button that is always accessible to the user from every stage in the game.	4	3
Navigation	1.3	Dhvani	As a user, I want a guided map, so that I can have easy access to rooms as they unlock and meet new people, see the remaining unlocked rooms, number of good and bad people identified and identify the current room.	Show a visualization of the route taken by the player through the form of a map. Include labels and mark the visited levels differently than the ones that are not visited.	1	8
Code bytes	1.4	Hia	As a general user, I want to game currency (called code bytes) in order to access commands in the game and ultimately win the game.	The code bytes will be implemented as a private attribute in Player class, and each time there needs to be a use of code byte, setter, update, and getter methods will be called for this attribute.	6	5

Visuals	1.5	Dhvani	As a user with a visual learning style, I would like to have helpful visual aids such as symbols, high contrast options and font customization to help me visualize the gameplay.	Add visuals relevant to the game's theme and setting. Include dark and light mode options. Allow for font resizing and styles changes.	2	8
Audio	1.6	Mahak	As a non-sighted user, I need descriptive audio that communicates the information I need to play the game.	Include audio descriptions of rooms, powerups, objects, possible moves, characters, etc.	3	7
Interacting With Other Characters	1.7	Kanupreet	As a general user, I would like to interact with the NPC characters and defeat the Polymorphic prowler at the end.	Add a pop-up which asks the user whether they would like to ignore, trust, or hack into the other character's mind when encountering another character/opponent.	7	6
Pets	1.8	Hia	As a general user, I want a pet so that I have a companion in the game and take advantage of all the benefits they offer.	At the start of the game (after "character customization"), allow the user to select their choice of pet from a pre-decided list of pets. We keep track of these pets using a superclass called "Pet" and subclasses of different pets, with their special abilities and general information as attributes.	8	8
Quests + Last Battle	1.9	Hia	As a general user, I want to play quests and the final battle so that I can win the game.	Every time the user hacks a prowler and they have sufficient code bytes, the questView will load and allow the user to play the quest. For the last battle, once all prowlers have been defeated, the last battle view would be called.		

Acceptance Criteria

Name	ID	Description
Character Customization	1.1	Users should be able to customize or choose a playable character in some way. Users should be able to choose the name of their playable character. The characters have lives and code bytes that can be displayed on the screen.
Menu	1.2	Descriptive instructions are provided to the user. Users can restart the game. Users can click on instructions to load instructions about how to play the game. Users can exit the game anytime.
Navigation	1.3	Navigation map can be accessed at any time during the game. Users are aware they can access a map using the menu or keyboard shortcut. Users are able to see which places they have visited and haven't visited. User is able to obtain a summary of the room when clicked on the map. Users are able to zoom-in or focus on a specific area of the map.
Code bytes	1.4	The code bytes must be in the inventory in order for the user to use them. The code bytes are single use. Users can acquire these codebytes from quests and trusting school members.
Visuals	1.5	Given that I am a user with low vision I would like the option to customize visuals. There is a pop-up to customize visuals at the beginning of the game. User is able to select the preferred font size and color mode. Users are able to change their visuals settings at any time during the game through the menu.
Audio	1.6	Descriptive audio commentary is provided for the events of the game. There is auditory description for the setting (i.e. "You have entered [Room]"). There is auditory description for the characters appearance (i.e. "An alchemist is a character that (...)") There is an auditory description for each button (i.e. instructions, help and map) There is auditory description for the player's setting upon the user's command (i.e. "You are standing across from a Polymorphic prowler! It is now in the form of a wolf and looks at you hungrily, eyes gleaming.") There is auditory description that describes the player's possible actions when encountering an opponent (i.e. Hack, Ignore, Trust) There are sound effects for when the button is pressed. This is to ensure that the player knows what is interactive.

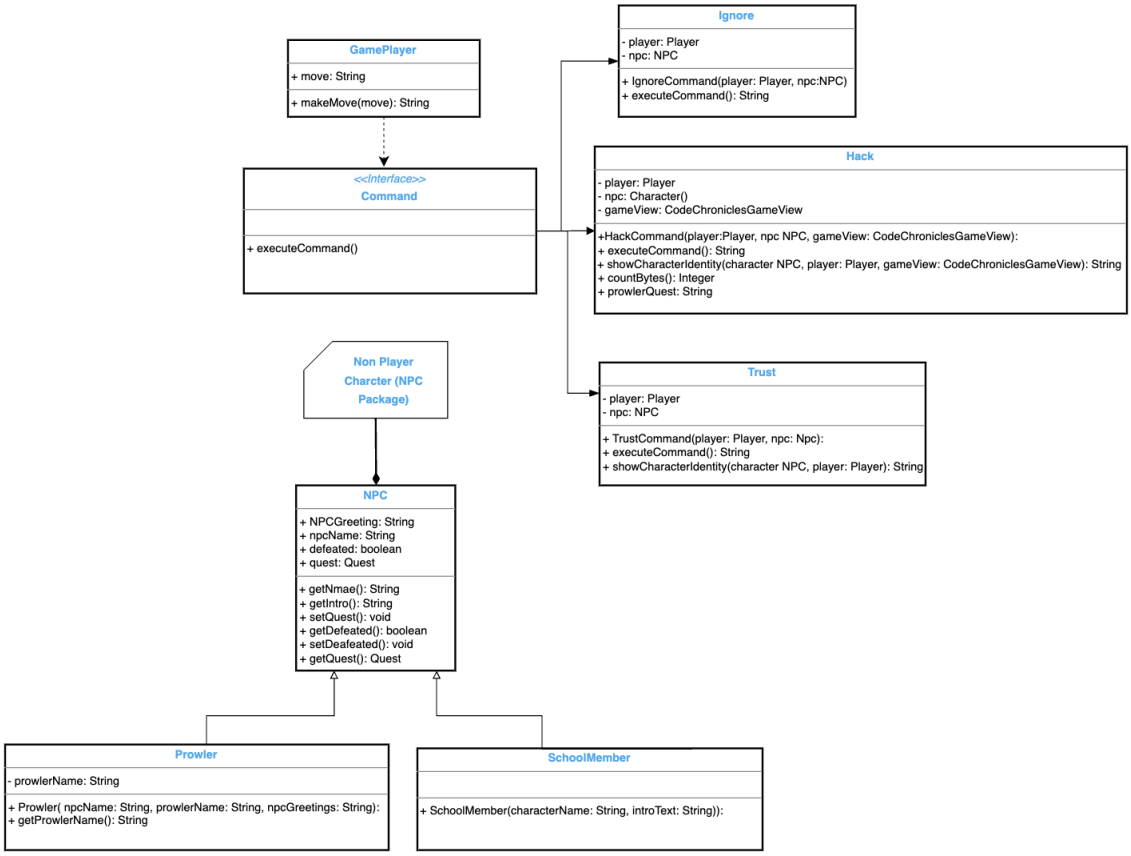
Interacting With Other Characters	1.7	The player (User) interacts with other characters in the game. Users can approach another character in the room to get their introduction. Users decide to interact with the character not knowing if it is a disguised prowler. Users have the choice to ignore, hack or trust the character. Users trusting a prowler will come with the cost of losing some of their character's power. Users find out if it is a polymorphic prowler or school member by hacking into their mind at the cost of code bytes which they collect along the adventure.
Pets	1.8	The pet must search for threats in the room. The pet must use their special abilities to help the character in quests. The pet acts as a companion to the user while playing the game.
Quests + Last Battle	1.9	The player is able to play the quest successfully. The audio and visuals run smoothly. The last battle is functional and allows the player to defeat the Incognito Phantom.

Section 3: Software Design

Design Pattern #1: Command

Overview: This pattern will be used to implement the game's player interacting with other characters.

UML Diagram: (on the next page)



Implementation Details:

- The <**Command**> interface has the **execute()** method that follows the Command design pattern and the three commands, use the execute method.

- The player selects from the three <**Command**> that are the <**Ignore**>, <**Hack**> and <**Trust**> classes.
  - The <**Ignore**> command, the player wants to ignore the NPC character and move forward in the game.

  - The <**Hack**> command works when the player wants to hack into the NPC character's mind.
    - If the character is an instance of prowler the player can fight against it in the form of solving a quest, which if the player wins gets more bytes in order to move forward.
    - On the other hand, if the character is an instance of a school member then the player loses a life.

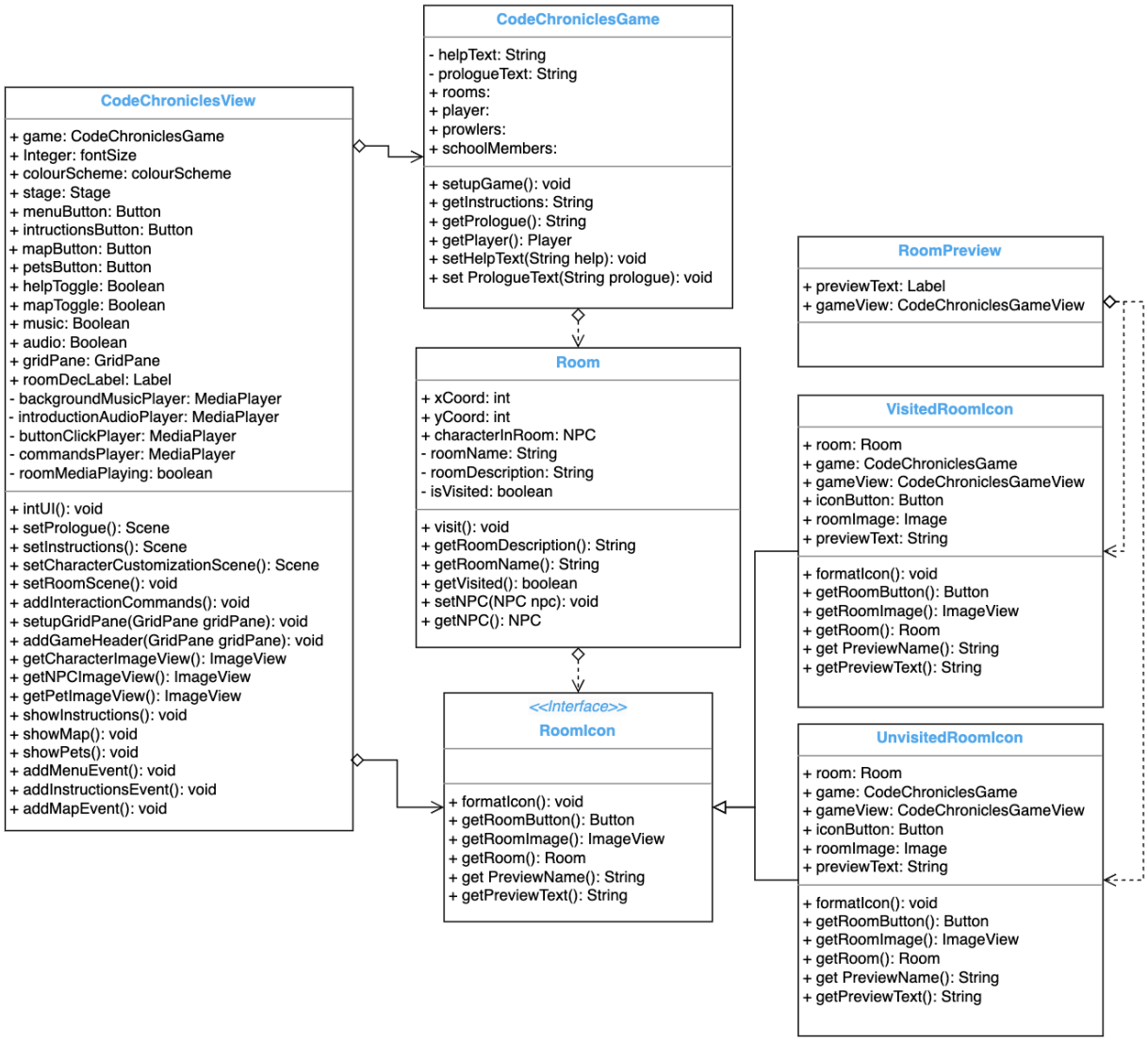
  - The <**Trust**> command comes into action when the player decides to trust the NPC character.
    - If the character is an instance of prowler, then the player loses a life.
    - On the other hand, if the character is an instance of a school member then the player gets +10 code bytes in reward to the player.

- Both the characters <**SchoolMember**> and <**PolymorphicProwler**> extend the NPC character class which is inside a package Non Player Character.

Design Pattern #2: State

Overview: This pattern was used to implement the player's navigation throughout rooms using the "Map". How the rooms are displayed on the map will vary depending on their ***state***.

UML Diagram:



Implementation Details:

- The <**CodeChroniclesView**> class displays the stage for the game.

- The <**showMap**> method is called when the user clicks the "Map" button and iterates through the <rooms>, creating <**RoomIcon**> instances to display on the map. The iteration checks the room's visited boolean attribute. If the room has already been visited by the player, it creates a <VisitedRoomIcon>, and if the room hasn't been visited, it creates an <UnvisitedRoomIcon>.

- Each <RoomIcon> initially displays the <roomName>. If the player clicks on the button of <RoomIcon> on the map it initializes a <**RoomPreview**> which is a new pop-up displaying the room name, room image, room description, and a "Go Here" button.
    - <**VisitedRoomIcon**> extends the <RoomIcon> interface. When the <VisitedRoomIcon.iconButton> is clicked it displays a clear image of the room along with its description.
    - <**UnvisitedRoomIcon**> extends the <RoomIcon> interface. When the <UnvisitedRoomIcon.iconButton> is clicked it displays a blurred image of the room, along with a message stating the player must enter the room to see its details, instead of displaying the room description.

- When the player clicks the "Go Here" button on a <RoomPreview>, for a <**VisitedRoomIcon**> it calls a method which sets the player's room to the respective room and sets the room's visited attribute to true.

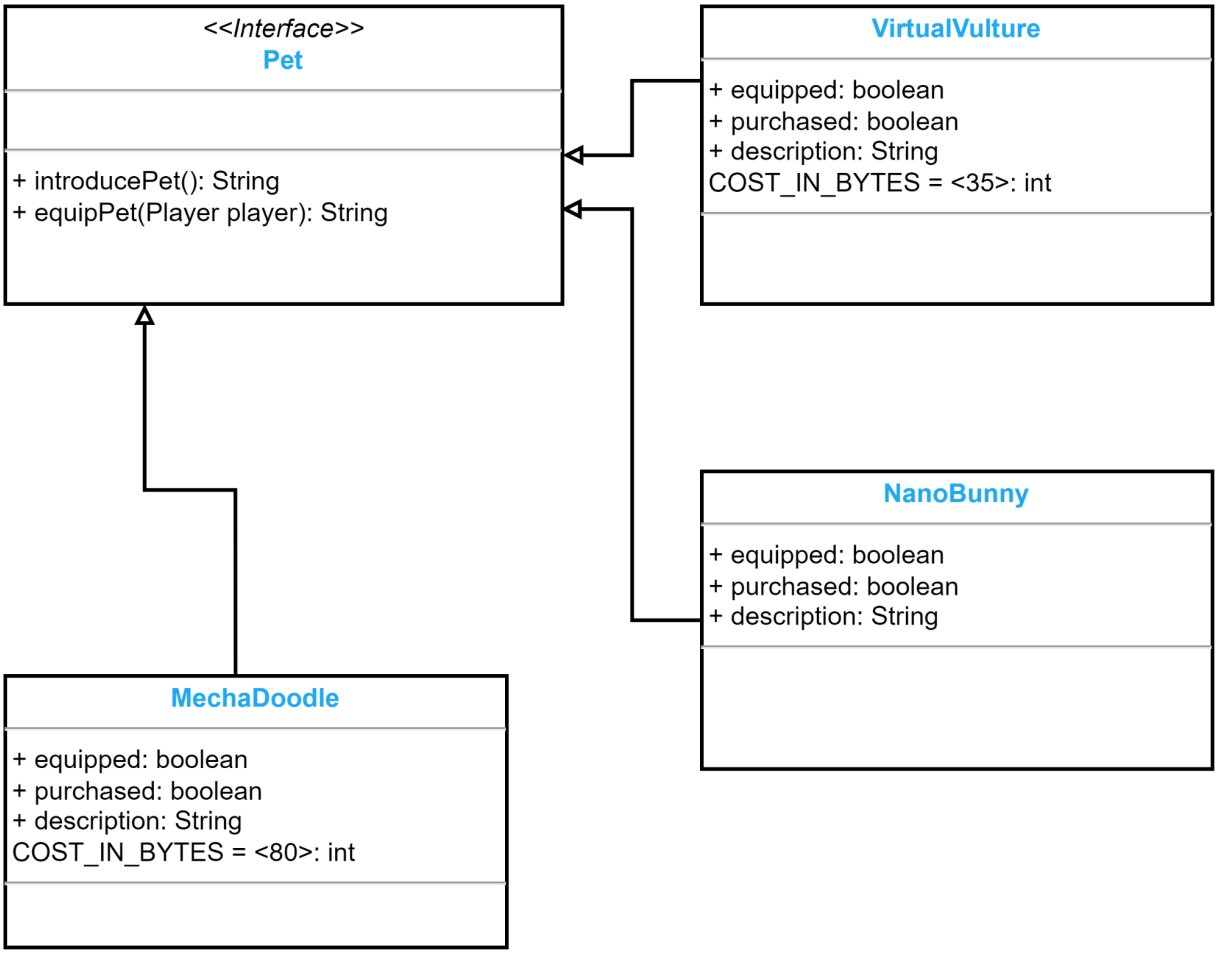
- When the player clicks the "Go Here" button on a <RoomPreview>, for a <**UnvisitedRoomIcon**> it calls a method which first checks if the player has enough <**CodeBytes**> to unlock a new room.
    - If the player has enough <CodeBytes>, it sets the player's room to the respective room and sets the room's visited attribute to true.
    - If the player doesn't have enough <CodeBytes>, it sets the <RoomPreview>'s previewText attribute label to a string message for the player

- Each time the b method is called, it is updated based on the newly visited rooms as it newly created icons each time.

Design Pattern #3: Strategy

Overview: This pattern will be used to implement the interface for "Pet" to allow the player to pick their choice of pet and make use of their abilities.

UML Diagram:



Implementation Details:

The <**Pet**> Interface provides the basic functionalities and characteristics of a pet in the game.

The classes <**VirtualVulture**>, <**NanoBunny**>, and <**MechaDoodle**> implement this interface.

I. <**VirtualVulture**> represents a pet that can only be unlocked once the character acquires a number specific code bytes. It has the ability to give hints in quests and reveal answers.

II. <**NanoBunny**> represents a starter pet that does not have any requirements from the character, and can give hints during quests.

III. <**MechaDoodle**> represents a pet that, like the <**VirtualVulture**>, also needs a specific number of code bytes. They are special because they provide hints in both quests and the last battle, while the other pets don't give hints in the last battle.

The <**Pet**> interface itself offers methods that are to be implemented by its subclasses. They are listed below

For example,

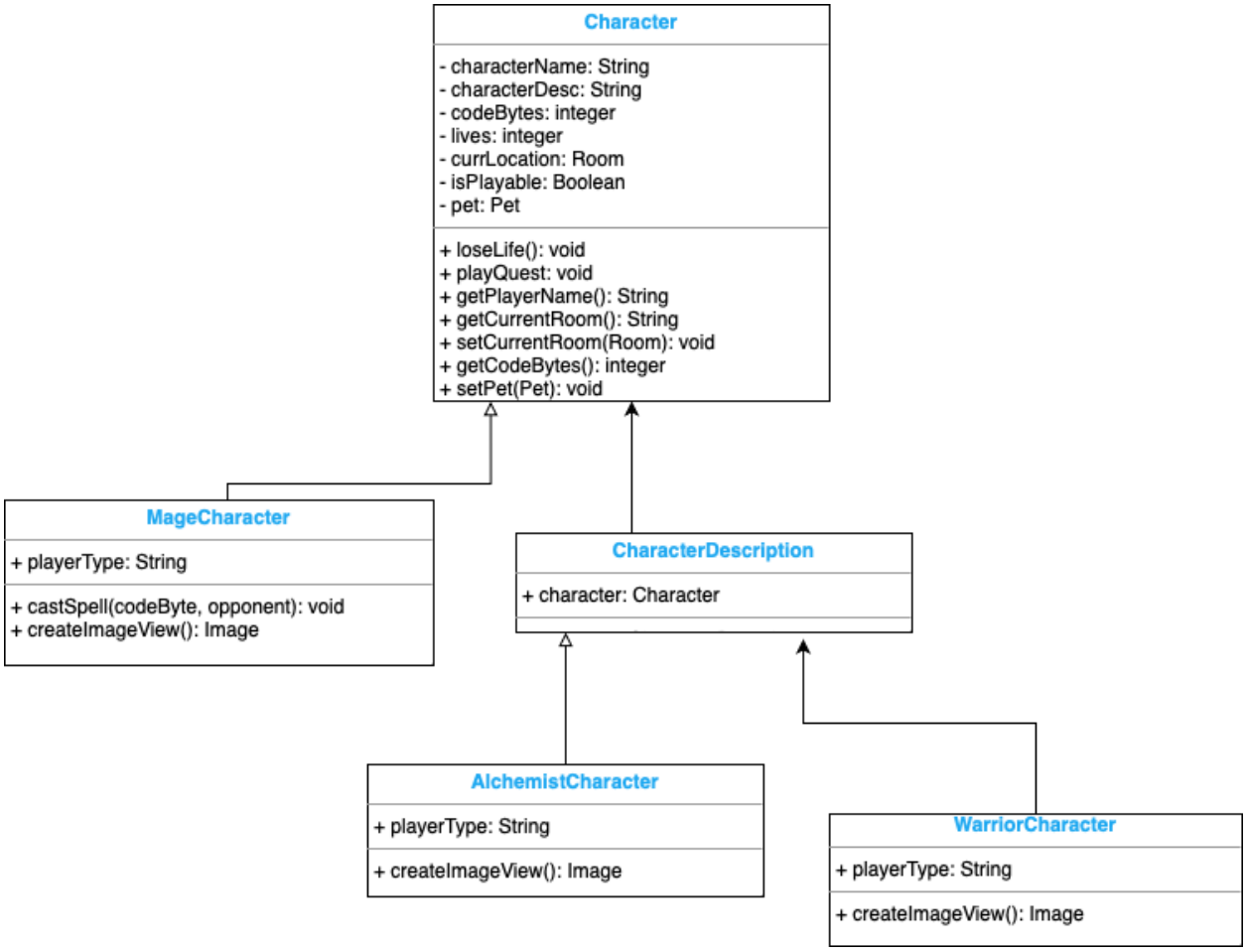
I. introducePet() returns an introduction of the pet.

II. equipPet() takes in a player and evaluates and returns a string that indicates whether the player can choose the pet or not.

Design Pattern #4: Decorator

Overview: This pattern will be used to implement the "Character" interface to allow for character customization (user story 1.1).

UML Diagram:



Implementation Details:

The <**Character**> interface reflects the general character structure.

- <**MageCharacter**> extends the superclass and has no special powers/features.

- <**CharacterDecorator**> acts as a wrapper and implements the Character interface.

    A. <**AlchemistCharacter**> extends the decorator functionality and modifies the Character behavior as such:
        a. AlchemistCharacter objects have an elementalAffinity attribute (i.e. fire, water, wind, earth) and hold the Philosophers Stone. Both can be used in battle once during the game using castAffinity() and usePhilosphopersStone().
        b. Their usage is tracked through the affinityUsed and stoneUsed attributes, respectively.
        c. The castSpell() method is overridden to reflect the change in AlchemistCharacter's lines when in battle.

    B. <**WarriorCharacter**> extends the decorator functionality and modifies the Character behavior as such:
        a. WarriorCharacter objects hold a diamond shield (guarantees victory in a battle when used). The shield can be used in battle once during the game using useShield().
        b. Its usage is tracked through the shieldUsed attribute.
        c. The castSpell() method is overridden to reflect the change in WarriorCharacter's lines when in battle.