Design Document

Version 1.1 – 2023.11.04
***Created* 2023.11.04**

FC Adventure Story

Akinbowale Adetunji
Joshua Spencer
Jordan Jon
Cedric Ngounou

GitLab Repository:

https://mcsscm.utm.utoronto.ca/csc207_20239/group_98

SECTION 1: PROJECT IDENTIFICATION

Our objective for this project is to produce a fun and interactive game model by adding a variety of accessibility features to expand our potential playerbase. We aim to accomplish the latter by implementing sound features such as speech-to-text, text-to-speech, etc, and visual features like colour-blind mode, zoom, etc.

SECTION 2: USER STORIES

Name	ID	Owner	Description	Acceptance Criteria	Implementation Details	Priority	Effort
Input	3.1	Jordan	As a user of the game who has trouble reading the game text, I need a way for the user to understand how the game functions.	Given that I am a user who has trouble reading the game text, when I try to play the game, I can without difficulty.	Add a Text-to-Speech button that can convert a text input using a third party library into an audio file to be played.	2	3
Colour Blind	1.2	Akinbowale	As a user of the game I need to be able to toggle between dark and light mode for the game to make my playing experience more enjoyable so and so that the game adheres to my visual preferences.	Given that I am a user of the game when the game loads I should be able to switch between dark and light mode based on my preference.	Add a dark and light mode button in the settings so that once pressed it changes the game's appearance from light mode to dark mode and vice versa.	1	3
Augment view	2.3	Cedric	As a low-vision user, I need to be able to augment the display panel to full size, so that I am able to acknowledge and read text.	Given that I am a low-vision user when the game loads I should be able to press the full screen button, to display the full screen.	Add the option to go full screen to the panel, adjust the ratio of all the components of the gridpanel to the full size one.	2	2
Settings	1.4	Cedric	As a developer, I want create settings button, so that a collapsed sidebar containing buttons to change to dark, light, high-contrast and colour blind mode, save a game, load a game, display the instructions via the help button and quit the game is displayed when clicked	Given that I am a developer when the settings button is clicked a collapsed sidebar should appear, contain all cited buttons who work, and disappear when "x" is clicked	Add a button with the settings icon, relate that button to an event, create a sidebar containing the dark mode, light mode, high-contrast mode, colour-blind mode, help, save, load, and quit buttons.	1	2
Death Troll	2.7	Akinbowale	As a developer I want to edit the Troll class originally given in A2, to include a new type of troll given as the Death Troll.	Given that I am a developer of the game I want to be able to extend the pre-existing abstract troll class already established within the game to add more functionality to the Death troll such as the attributes of the types of weapons the troll has and the damage taken and sustained by the Death	Create a child class that extends the Abstract troll class, to include the attributes such as the type of weapon the troll has, the troll's current health status and the methods given as attackPlayer.	2	3

				Troll.			
Display	1.8	Jordan	As a visually impaired user, I need to be able to play the game without struggling to interpret words and numbers.	Given that I am a visually impaired user, words and numbers in small font are very difficult to see and the game should have a feature that should allow me to understand what the game is trying to display.	Implement a health bar which contains easy to understand, large symbols of which are easily visible and do not contain text.	1	2
Alt Text	1.9	Jordan	As a non-sighted screen reader user, I need informative images and buttons to include descriptive alt text, so that my screen reader can communicate the information I need to play the game.	Given that I am a non-sighted screen reader user, I need images and buttons to include descriptive alt text so that my screen reader can communicate the information I need to play the game.	Add descriptive alt text to all buttons and images that will be displayed so that any non-sighted screen reader users will have a smooth gameplay experience.	1	2
Player fight trolls	1.10	Cedric	As a developer, I want to extend the player class, so that player can have a username and be able to fight and heal	Player HP changes when healed Player attacks Troll with equipped weapon Player contains the username entered by user Player can pick up weapon in the room	Add HP, Weapon and Username attributes to Player and add attack, pickup_weapon, heal and protect to Player class.	1	1
iSpy Troll	4.11	Jordan	As a developer, I want to implement the Troll interface as an iSpy troll so that users can enjoy the game in a unique way.	Given that I am a developer, I want to be able to implement an iSpy troll as a fun minigame in order to entertain and provide a new experience for all types of users that might have accessibility needs.	Add different presets of iSpy images with varying types of colours in order to suit the needs of different types of colour blindness.	1	1
Math Troll	3.12	Akinbowale	As a developer I want to edit the Troll class originally given in A2, to implement a new type of	Given that I am a developer of the game I want to be able to extend the pre-existing	Create a child class that extends the Abstract troll class, to include the attributes	1	2

			troll given as the MathTroll to add a new element to the game.	abstract troll class already established within the game to add more functionality to the Math troll such as the attributes such as the type and number of questions as well as the answer key to the questions and the amount of HP damage associated with each wrong answer.	such as an array of questions and answers, as well as the amount of HP damage associated with each wrong/right answer. This would also contain the method checkAnswer and askQuestion.		
Sport Troll	3.14	Cedric	As a user facing a sports troll, I want to be able to read and hear the questions so that I can answer the question and attack the troll.	Wrong answer the troll attacks the user Right answer the player/user attacks the troll When the troll's HP is 0, the player advances If the player's HP is 0, they die	Create a SportTroll abstract class extending the Troll class to include attributes such as an array of question/answers, a weapon and HP. The class will contain an attack method.	2	2
Leaderboard	4.15	Cedric	As a user who wants to see the top times to complete the game, I want to be able to press the leaderboard button so that a leaderboard panel containing the best scores appears.	When clicked a panel with LEADERBOARD written on top must appear Only displays the top 10 times Displays the times from lowest to highest Displays as position, username and time	Create a leaderboard file with position, username and time on each line, and put each line in an array of lists containing a name and time when the game is started. Create a leaderboard class, which contains the arraylist of leaders and a method which checks if a player should be added to the list.	2	2
Enlarge Text	3.16	Joshua	As a user who is slightly visually impaired I want to be able enlarge the text the on the game screen so that I can read any text that might be too small for me to see.	Given that I am a slightly visually impaired user playing the game when I come across text that is too small, I can opt to enlarge the text on the screen using settings.	Implement an enlarge function that will enlarge all text on screen when a button is pressed.	2	3
Username Panel	1.17	Joshua	As a general user who plays the game I want to be able to successfully create my own username	Given that I am general user of the game when the game first loads up I will be able to input and pick my own username and that will	On the startup screen implement an input field that will ask for the player's username which will then be used to instantiate a new player object for that	3	3

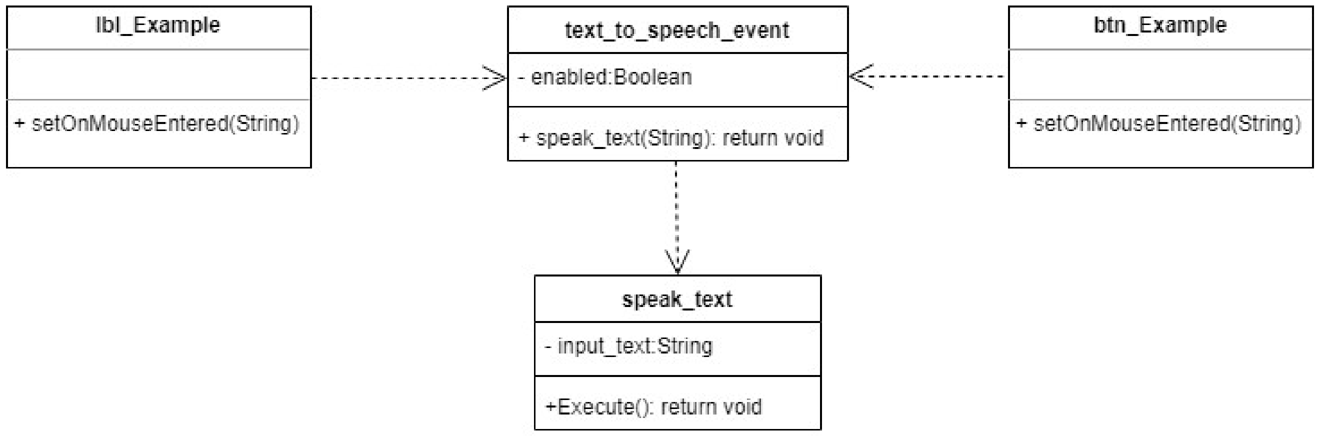
			to keep track of my progress and status in the game.	be associated to my progress in the game,	game session.		
Easy, hard load	3.18	Joshua	As a general user who plays the game I want to be able to adjust the difficulty of the game based on my own preference	Given that I am a general user of the game, when the game loads up on the startup screen I can change or edit the difficulty of the game before the game starts up and begins.	Implement two buttons on the start up screen one for easy and for hard mode. Depending on which button is selected by the user then it would indicate which game file will be loaded (easymode game file or hardmode game file).	1	2
Time Count	4.20	Joshua	As a competitive user I want to be able to see how much time it took to complete the game so that I am able to compare my times to other times on the leaderboard.	Given that I am a competitive user I should be able to click locate the amount of time currently elapsed for the game session until I complete the game or die.	Add a panel to the top left or right corner of the game screen that keeps track of the time elapsed which makes use of the Timer class in java.	1	2
Weapons Class	4.21	Joshua	As a general user who plays the game I want to be able to identify the different types of weapons available to use and know what type of damage it does.	Given that I am a general user of the game when playing the game, I should be able to inspect a weapon and get information about it.	Implement a pop up window (Pane) that allows the user to select whether they want to view weapon information and if selected yes it displays the information which would be attributes of the weapons class.	2	1

SECTION 3: SOFTWARE DESIGN

UML Diagram:

Design Pattern #1: Adapter

Overview: This pattern will be used to implement Text-to-Speech by converting a string of AccessibleText into an audible output of words.



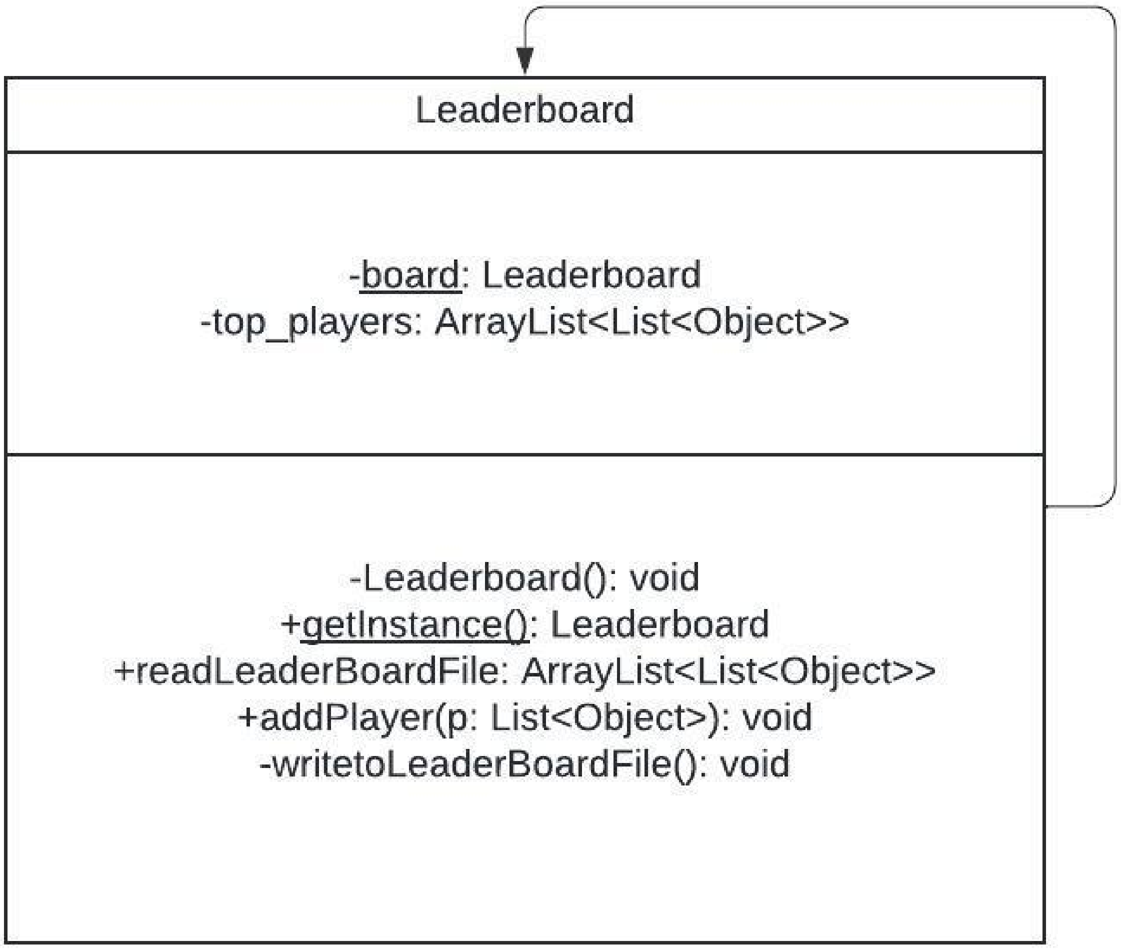
Implementation Details: The UML diagram outlines these main components:

- The *text_to_speech_event* class, which includes the method, speak_text.
- The *Speak_text* class, which includes the method, Execute.

The btn_Example/lbl_Example is a button/label when mouse hovered, it calls the setOnMouseEntered method which will create a text_to_speech_event object. When the text_to_speech_event object is created, it creates a speak_text object if text to speech is enabled by checking the enabled boolean. The speak_text method calls Execute and begins the audio output of input_text using the freetts library. The adapter design pattern is used here as the *speak_text* class converts the text into verbal audio that will be used to play the game as required. This in particular relates the user story Input which adheres to players who might have trouble reading labels and buttons to play the game.

Design Pattern #2: Singleton

Overview: This pattern will be used to implement the Leaderboard class in our adventure. It will allow our code to only create a singular instance of this class.


Leaderboard
-board: Leaderboard
-top_players: ArrayList<List<Object>>
-Leaderboard(): void
+getInstance(): Leaderboard
+readLeaderBoardFile: ArrayList<List<Object>>
+addPlayer(p: List<Object>): void
-writetoLeaderBoardFile(): void

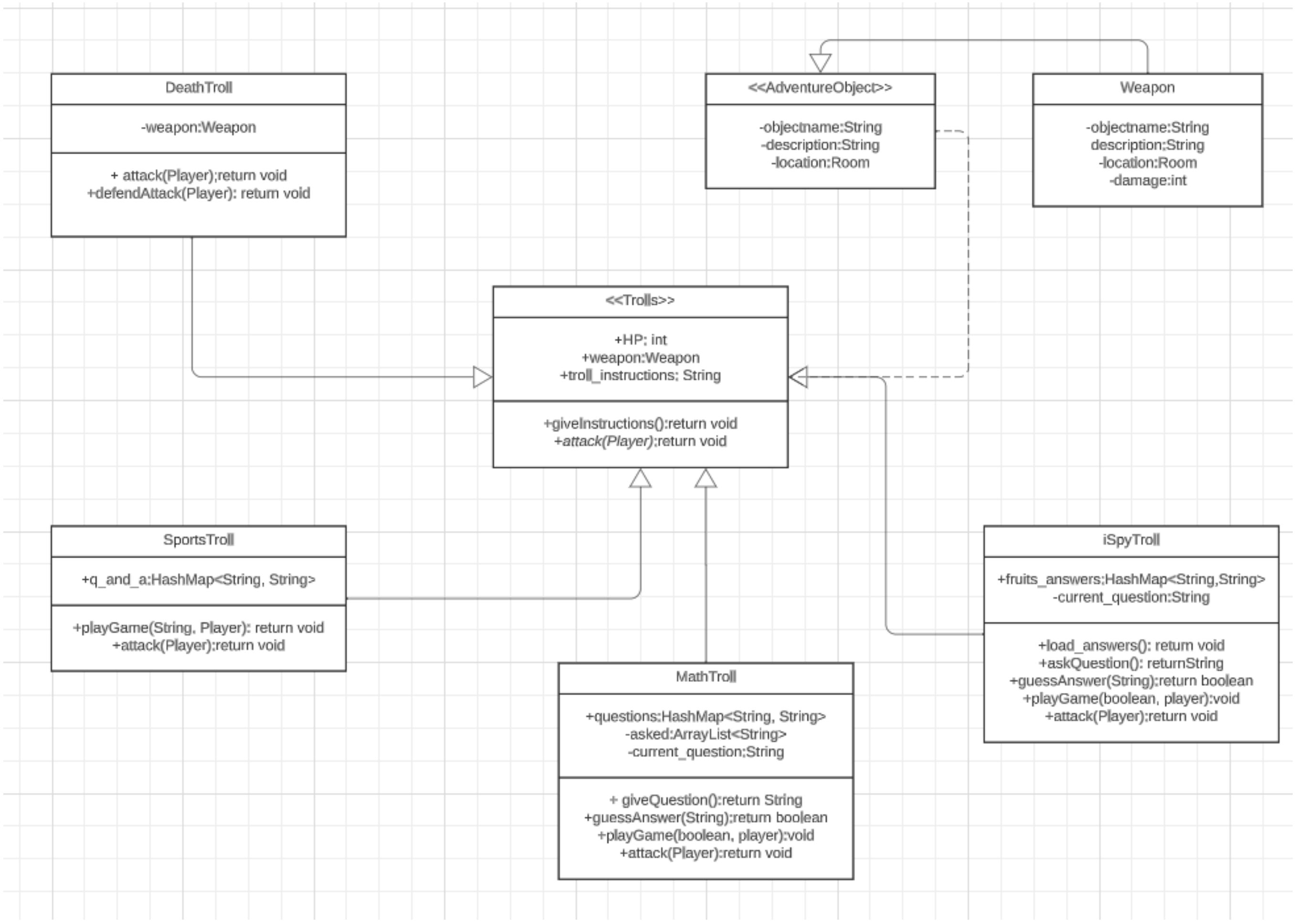
Implementation Details: The UML diagram outlines these main components:

- The Leaderboard class, which creates a single instance of a Leaderboard object, it creates its instance by using the public getInstance class, it has a private construct where, we initiate the top_players attribute, and contains a public addPlayer method, readLeaderBoardFile and writetoLeaderBoardFile.
- To avoid many dependencies and avoid multiple creation of instances

When the AdventureGameView is instantiated, we pass the singular Leaderboard object to a leaderboard attribute. We create it by using Leaderboard.getInstance(). When doing so, a Leaderboard.txt file containing the name of the players and their times will be read and used to populate the Leaderboard's top_players attributes which is an arraylist containing a list of username-time pairs. When the game ends, we try to addPlayer(p) to the leaderboard if his time can figure in the top ten, we update the top_players attributes and write the array list to the Leaderboard.txt file by using writetoLeaderBoardFile.

Design Pattern #3: Strategy

Overview: This pattern will be used to implement the various trolls within the game.


DeathTroll
-weapon:Weapon
+ attack(Player);return void
+defendAttack(Player): return void
<<AdventureObject>>
-objectname:String
-description:String
-location:Room
Weapon
-objectname:String
description:String
-location:Room
-damage:int
<<Trolls>>
+HP; int
+weapon:Weapon
+troll_instructions; String
+giveInstructions():return void
+attack(Player):return void
SportsTroll
+q_and_a:HashMap<String, String>
+playGame(String, Player): return void
+attack(Player):return void
iSpyTroll
+fruits_answers;HashMap<String,String>
-current_question:String
+load_answers(): return void
+askQuestion(): returnString
+guessAnswer(String):return boolean
+playGame(boolean, player):void
+attack(Player):return void
MathTroll
+questions:HashMap<String, String>
-asked:ArrayList<String>
-current_question;String
+ giveQuestion():return String
+guessAnswer(String):return boolean
+playGame(boolean, player):void
+attack(Player):return void

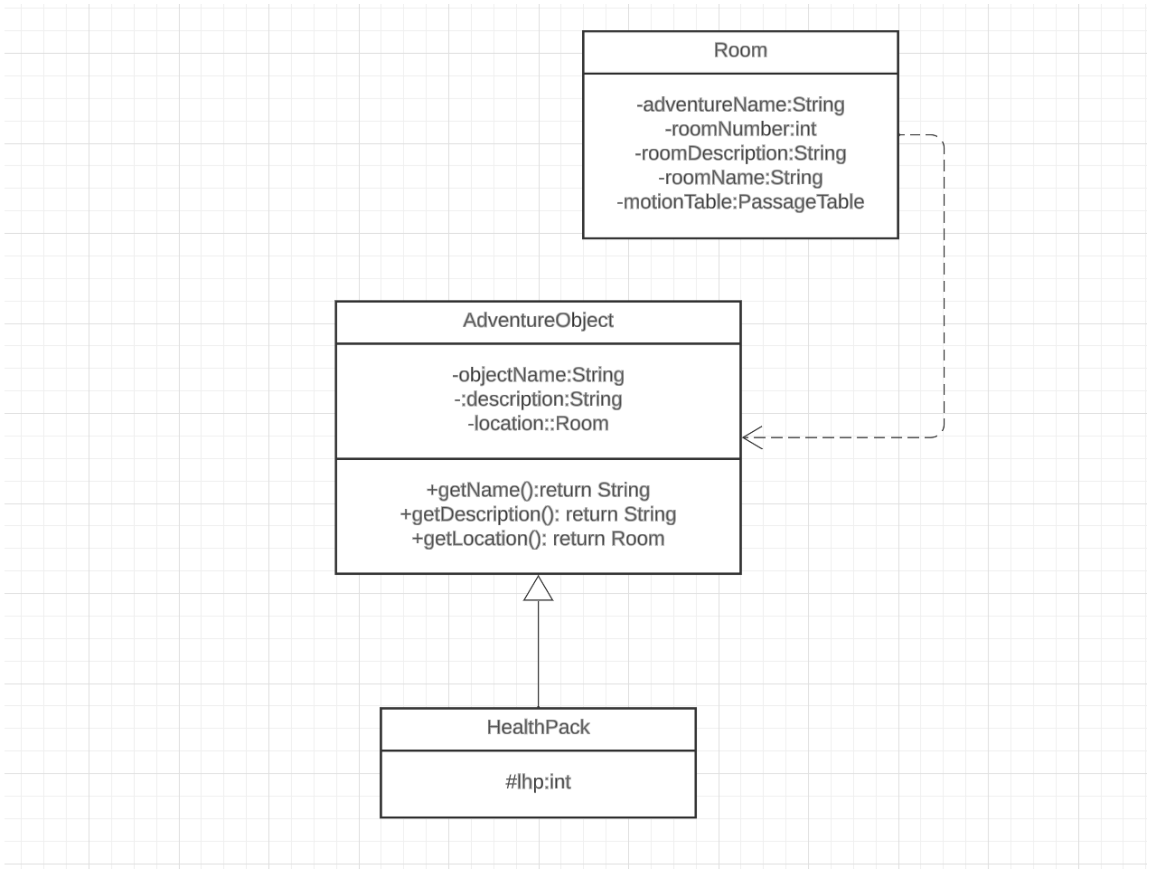
Implementation Details: The UML diagram outlines these main components:

- The *Abstract Troll class*, which provides the framework for the different types of trolls.
- The *Weapons Class*, which includes the attributes and methods associated with the different types of weapons the players can have.

Each of the separate troll classes contains different methods which represent the different ways in which the player can attack or defeat the troll. The strategy design pattern is demonstrated within the trolls class as for each troll encountered the player has a variety of options to pick on how to attack the troll. This ranges from directly attacking the troll or having to answer a series of questions. Each child troll class inherits the *giveInstructions* method which gives the players instructions on how to defeat the troll. Additionally, each troll class also inherits the abstract method *attack*, which is the method the troll uses to attack the player. Based on the type of troll there are certain attributes and methods associated with the different ways the player can attack and defeat the troll. Additionally. each child class of the parent troll class inherits the attributes *HP*, *Weapon and troll_instructions.* The *attack* method in each troll child class hits the player and returns void. The player is notified when they are hit with an update to their health bar. The *HP* attribute gives the health of each troll whereas the *Weapon* attribute keeps track of the weapon the troll has and the amount of damage the troll can do. Additionally, the *troll_instrcutions* attribute stores the specific instruction on how to play the game for that given troll. This relates to all the various troll user stories as it addresses the functionality and the acceptance criteria required for each troll user story.

Design Pattern #4: Open-Closed (SOLID Principles)

Overview: This pattern will be used to implement a healthpack into the game



Implementation Details: The UML diagram outlines the following main components:

- The *AdventureObject* class, which is the class used to instantiate an AdeventureObject in the game. The *AdventureObject* class is dependant on the *Room* class which denotes which room the adventure object is located.
- The *HealthPack* class which is an extension of the *AdventureObject* class.

The HealthPack class is an extension of the *AdventureObject* class in the main program. That is, it takes the *AdventureObjec*t class and adds additional functionality. This functionality lies mainly in the hp variable which allows the heal_player function from Player class (which is dependent on HealthPack) to be implemented. This heal_player function is used to restore the player's hp. It does this by adding the value stored in the Healthpack's hp to the player's hp. This is shown visually on the screen, through the hearts at the top of the screen. The implementation of HealthPack adheres to the open-closed principle which states that classes should be open for extension but not to modification. In this instance, the AdventureObject class is not modified but instead extended to allow for the inclusion of HealthPacks in the game. The above UML describes this relationship and offers a simple way to grasp the intricacies of the extension and dependencies of the HealthPack.