|  |  |
| --- | --- |
|  |  |
|  | Name: Jean-Erick Voigt  Date: 2 September 2017 |
|  | Current Module: Python OOP |
|  | Project Name: Dungeons\_and\_dudes.py |
|  |  |
|  | Project Goals: |
|  | Project was designed to evaluate our ability to use OOP. We were tasked to utilize class objects in order to make it easier to make instances of a variable with all the requisite attributes. The project is a text-based adventure game. We had the option of putting a variable amount of rooms in the project. When in each room we had a menu of options that we could perform. In order to advance to the end we had to either fight the monsters in the rooms or find a way around them. The loot that we received was not necessarily beneficial to the quest but could be. |
|  | Considerations: |
|  | <A bulleted list of considerations and conditions which affected the design of your project> |
|  | * Classes   + Inheritance   + super\_init * File input   + Read * Using modules   Initial Design:    This program was split into multiple modules. The main module classes in order to make the objects. The objects were broken down as rooms, characters, monster and player. The player class and monster class inherited name, hp, maxHP. I used two while loops. One for the game state and one for the room. The program was planned to not continue to the next room until it beat all the monsters in the current room it was in. The max amount of monsters that you could have in a room was 4.  Data Flow:    Initially, i use the welcome function to get the name of the player and assign it to a character object to play with. I use file i/o and randomize the selection of a monster in the text file. The amount of monsters needed for that room are then appended into the room. Play in each room is automatically rolled for to see who gets the initiative of the first turn. Afterwards, the user has an option to use an item, look at an item, runaway, attack and check his stats from option menu. The game continues in this fashion until you either run out of HP, the game is won or you quit.  Communication Protocol:    <Discuss any custom communication protocol you have used including the various request and responses that may be sent and received.>  Potential Pitfalls:  The big potential pitfalls have to do with modularity. Also, when you do things in classes, some methods can not be used for them because those methods apply to int or str objects and not class objects.  Test Plan:  User Test: The main test method that I utilized was the “look before you leap” method by using a lot of print statements and if checks.   Test Cases:  Conclusion:  The project became an obstacle because I struggled a bit with the modularity. After I got that there were issues with how is was saving lists and to what object to lists were being saved to. As a result I was not able to go beyond the attack option on the full game. However, while checking the classes i was able to check the individual classes to ensure that they would work as desired. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |