CODE REVIEW -- JAKUB SLIWINSKI

--

BY OLIVER LAZARUS-KEENE 29218390.

General Comments:

* Program runs to completion without any unhandled exceptions.
* Erroneous inputs are handled and do not cause unexpected behaviour.
* The player can navigate through rooms, and pick up items on the floor to add them to their inventory
* The player can get the description of the room they’re in, and see their health.

Examples Of OOP Use:

* Classes and objects are used throughout
  + Game, Player and Room classes all interact with one and other to make the game run. The Game class contains two instances of Player and Room.
* Constructors:
  + Constructors are correctly used to assign object values when they are initialized.
* Encapsulation is widely used:
  + Within the Room Class, Room.Description is private (Assigned in the constructor) and there is a public Room.GetDescription function, ensuring the value is read only after construction of the object.
  + For attributes in the Player Class, such as Name which again is assigned only in the constructor and then left read only.
  + Within the Player Class, the inventory is left private and a simple get function outputs its contents.
* Instantiation:
  + Within the Game Class, an instance of player and room is made and used in the game loop.
* Exception Handling:
  + Within the Main function, a try-catch block is used which displays a simple error message to the user in the case there is errors within the Game.Start() Function

Improvements:

* Within the Game class, the game loop is present in the Start() function. This function takes input from the user and then does actions depending on what the user inputs. This is done using 7 if statements, consider using a switch() statement for easier reading.

Final Thoughts:

* The objectives of the brief have been well met, and the code shows good use of OOP concepts.
* The code is well commented and easy and understandable to read. It follows the style guidelines.
* All errors are correctly handled.