1 Assignment

Write a simple text-based "adventure game" that takes a description of the world in JSON and lets a user interactively navigate through the world. The game should print to and take input from the console.

Fork the starting git repository using the URL below. This time we're not providing you any starting code, but you should start from the provided repository so that we can have access for grading purposes

`https://classroom.github.com/a/quptGjb4`

2 JSON representation of the game world

Your project should accept the description of the game world using the following schema:

```
Direction := {
  directionName   : String
  room            : String         //name of the room the direction is point to
}

Room := {
  name            : String
  description     : String
  directions      : Direction []
}

Layout := {
  startingRoom    : String         //the name of the starting room
  endingRoom      : String         //the name of the ending room
  rooms           : Room []
}
```

By default, your project should load JSON from the following URL:

`https://courses.engr.illinois.edu/cs126/adventure/siebel.json`

3 Interface (Output)

Starting with the `startingRoom` your game should display a message describing the player's current environment. This should include three parts, with a newline separating them:

1. The current room's `description` printed verbatim
2. If the room is the `endingRoom` print `You have reached your final destination` and exit
3. A list of the directions that the player can move.

4 Interface (Input)

After you write the message to the console, you should read a line of input from the user.

1. If that line is either the words `quit` or `exit` then you should exit the program.
2. If that line starts with `go some_direction` and `some_direction` is one of the directions they can travel from that room, you should move in that direction and repeat the process. If `some_direction` doesn't match any of the directions possible, you should print `I can't go some_direction`.
3. If the user inputs something that isn't one of the commands above, you should print out `I don't understand` followed by what the user typed in surrounded in single quotes.

Your code should ignore case when handling commands, but when echoing back a user's text should use the original capitalization.

5 Example

```
You are on Matthews, outside the Siebel Center
Your journey begins here
From here, you can go: East
go EAST
You are in the west entry of Siebel Center.  You can see the elevator, the ACM office,
and hallways to the north and east.
You are in the west entry of Siebel Center.  You can see the elevator, the ACM office,
and hallways to the north and east.
GO NoRtH
You are in the north hallway.  You can see Siebel 1105 and Siebel 1112.
From here, you can go: South
go TO HECK!
I can't go TO HECK!
You are in the north hallway.  You can see Siebel 1105 and Siebel 1112.
From here, you can go: South
gophers ARE tasty!
I don't understand 'gophers ARE tasty!'
You are in the north hallway.  You can see Siebel 1105 and Siebel 1112.
From here, you can go: South
go South
You are in the west entry of Siebel Center.  You can see the elevator, the ACM office,
and hallways to the north and east.
From here, you can go: West, Northeast, North, or East
You are in the west entry of Siebel Center.  You can see the elevator, the ACM office,
and hallways to the north and east.
From here, you can go: West, Northeast, North, or East
EXIT

Process finished with exit code 0
```

6 Requirements

The requirements of this assignment are listed below.

1. At run time download and parse JSON from a URL.
2. Correctly display the description / direction options for the current room.
3. Handle the user-directed navigation between rooms.
4. End the game when the user reaches the `endingRoom`.
5. Allow users to specify an alternate URL on the command line to get the room descriptions. Gracefully detect and handle the case where they've provided a bad URL.

7 Optional Features

You can implement these features for extra credit, these are not required and are only suggestions on how to expand the assignment. The more effort you spend, the more extra credit you get. Extra credit will be tracked separately from other grades and will be applied after letter grade cutoffs have been set.

1. Implement a 'map validator' for the layout JSON that checks to see if there is a path between the `startingRoom` and the `endingRoom`. If no path exists between the `startingRoom` and the `endingRoom`, your program should print the following and exit.

   ```
   The layout JSON is not valid. The endingRoom cannot be reached
   from the startingRoom.
   ```

2. Implement an extra 'floor-plan validator' for the layout JSON that checks for every pair of rooms A and B that, if you can get from A to B, then you can get from B to A. An example of a floor plan that should fail this validation is provided at the following URL:

   `https://courses.engr.illinois.edu/cs126/adventure/circular.json`

3. Add items that can be picked up and dropped.
4. Add a more robust vocabulary to the parser for example recognize run, skip, or swim rather then just go. You could extend the json to require a particular type of movement.
5. Additionally, if you are feeling creative and want to write your own JSON description to share with the class (use good judgment of what is appropriate), email `cs126sp19@gmail.com` a copy of the JSON and after brief review for content we will post them at:
   `https://courses.engr.illinois.edu/cs126/adventure`

Design and Style: As always, use the best design and coding style that you are familiar with. In particular, for this assignment we want you to pay attention to your *object decomposition*, that is how you break down the functionality of the assignment into classes and functions (static or non-static) of those classes. Be sure that code related to a particular object is in that class!

Process: You should continue to work on your code so as to implement small bits of functionality, testing them, getting them to work, and committing that piece of work. We expect to see a series of commits in your github with good explanatory commit messages.