Homework 10 - Object Oriented Programming

CS 1301 - Intro to Computing - Fall 2020

Important

- Due Date: **Tuesday, November 17th, 11:59 PM**.
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use class Piazza
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

Python is one of many coding languages which uses object oriented programming (OOP). In OOP, classes can be created which contain certain attributes and methods which are shared by all objects of that class. This helps you create concise code which you can re-use. The goal of this homework is to understand OOP and its real world applications.

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not true
```

Written by Caitlin Yang (caitlinyang@gatech.edu) & Damian Henry (dhenry35@gatech.edu)

Introduction

For this assignment, you will be building the Among Us game. there will be five classes working together: a Crewmate class, an Impostor class, a Task Class, a Room class, and an AmongUs class. Each of these classes will have attributes and methods, as described below. You have been provided with a file that has the beginning of these classes. You are responsible for filling in the rest, and the methods you need will be clearly listed out in the grading rubric.

Note: The given `HW10.py` has __repr__ methods, as well as some other useful methods that we have already defined for you. Do **not** delete or change these, as these are needed for testing.

Among Us

Among Us is a game where there are two types of players: a crewmate and an impostor. Crewmates and impostors look the exact same, so crewmates cannot distinguish who is an impostor. During the game, the crewmates must travel through a map to different rooms to complete tasks. However, the impostors are able to eliminate the crewmates as they go about the map.

If a crewmate stumbles upon an eliminated crewmate's body, they can call a meeting. At this meeting, all the players vote on the player that they think is an impostor. The player that has the most votes gets removed from the game, and it is revealed whether that player is a crewmate or an impostor.

For the purposes of this assignment, the game ends when all of the crewmates have been eliminiated or when all of the impostors have been eliminated.

Please read through the entire assignment before writing your code in order to understand how the different classes interact with each other!

Room (provided)

Attributes:

- name (str): the name of the room

Methods:

- __init__
  - initializes the following attributes
    - name (str)
- __eq__
  - Two `Room` objects are equal if they have the same `name`.

Task

Attributes:

- name (`str`): the name of the task
- isCompleted (`bool`): the status of whether the task has been completed (e.g. True if it is completed)

Methods:

- __init__
  - initializes the following attributes
    - name (`str`)
    - isCompleted (`bool`) - **every** task is not completed when the game starts
- __eq__
  - Two `Task` objects are equal if they have the same `name` and `isCompleted` status.

Crewmate

Attributes:

- name (`str`): the name of the crewmate
- color (`str`): the color of the crewmate

- accessories (`tup`): the accessories a crewmate has on
- isAlive (`bool`): the status of whether a crewmate is alive or not
- tasksDone (`int`): the number of tasks a crewmate has done

Methods:

- __init__
    - initializes the following attributes
        - name (`str`)
        - color (`str`)
        - accessories (`tup`) - if a crewmate does not have accessories, this attribute should be an empty tuple by default
        - isAlive (`bool`) - **every** crewmate starts off alive
        - tasksDone (`int`) - **every** crewmate starts off having done 0 tasks
- doTask
    - This method should take in a `Task` object.
    - If the task is not completed, the `Crewmate` should complete the task (look at the `Task` class to see what this means), and the `Crewmate`'s `tasksDone` should be incremented by 1.
    - Otherwise, return `"Nothing to do here."`
- vote
    - This method should take in an `AmongUs` object.
    - You must iterate over all crewmates and impostors. Return the first player whose name shares the first letter with the voting crewmate's name. All player names will have at least one character.
    - However, the names cannot be equal and the player must be alive.
    - You can assume there will always be a valid player to vote out. You should check the crewmates first to see if there is a player to vote for, and if there isn't, you should check the impostors.
- callMeeting
    - This method should take in an `AmongUs` object.
    - All of the crewmates and impostors in the game must `vote` a player out. The method should count how many votes a player recieves.
    - Then, the method should find the player that has received the most votes and set their `isAlive` attribute to False.

    - If the type of the player was an impostor, return `"{player name} was An Impostor."`. Otherwise, return `"{player name} was not An Impostor."`.

- __eq__ (provided)
    - Two `Crewmate` objects are equal if they share the same `name`, `color`, and `accessories`.

Impostor

Attributes:

- name (`str`): the name of the impostor
- color (`str`): the color of the impostor
- accessories (`tup`): the accessories a impostor has on
- isAlive (`bool`): the status of whether a impostor is alive or not
- eliminateCount (`int`): the number of crewmates an impostor has eliminated

Methods:

- __init__
    - initializes the following attributes
        - name (`str`)
        - color (`str`)
        - accessories (`tup`) - if a impostor does not have accessories, this attribute should be an empty tuple by default
        - isAlive (`bool`) - **every** impostor starts off alive
        - eliminateCount (`int`) - **every** impostor starts off having eliminated 0 crewmates

- eliminate
    - This method should take in a player, being a `Crewmate` object or an `Impostor` object.
    - If the player is of `Impostor` type, return `"They're on your team -_-"`
    - If the player is of `Crewmate` type, eliminate the crewmate by changing their `isAlive` status, and then, increase the impostor's eliminate count by 1.

- vote
    - This method should take in an `AmongUs` object.

    - You must iterate over all crewmates and then all impostors. Return the first player whose name shares the first letter with the voting crewmate's name. All player names will have at least one character.
    - However, the names cannot be equal and the player must be alive.
    - You can assume there will always be a valid player to vote out.
    - **Hint**: Same implementation as in the `Crewmate` class

- __str__
    - This method should create a string representation of the Impostor object in the format of: `"My name is {impostor name} and I'm an impostor."`

- __eq__ (provided)
    - Two `Impostor` objects are equal if they share the same `name`, `color`, and `accessories`.

AmongUs

Attributes:

- maxPlayers (`int`): the max number of players in the game
- rooms (`dict`): the key is the name of a room and the value is a list of `Task` objects that are in that room
- crewmates (`list`): a list of the crewmates in the game
- impostors (`list`): a list of the impostors in the game

Methods:

- __init__
    - initializes the following attributes
        - maxPlayers (`int`) - The max number of players allowed in the game
        - rooms (`dict`) - A dictionary in which the keys are the **names** of the rooms, and the value is a list of `Task` objects that are in that room. The rooms dict should be empty upon initialization
        - crewmates (`list`) - A list containing all `Crewmate` objects in the game. This should be an empty list upon initialization
        - impostors (`list`) - A list containing all `Impostor` objects in the game. This should be an empty list upon initalization

- registerPlayer

    - This method should take in either a `Crewmate` object or an `Impostor` object. This means only one "player" will be added at a time.
    - This method should first check to see if the number of crewmates and the number of impostors in the game is equal to the number of players allowed. If it is, the method should return `"Lobby is full."`
    - Next, the method should first check if there exists a crewmate with the same name as the player. If so, then return `"Player with name: {player name} exists."`
    - Then, the method should check if there exists an Impostor with the same name as the player. If so, then return `"Player with name: {player name} exists."`
    - If not, check the `type` of the player passed in. If the type of the player is a Crewmate, add the player to the crewmates list. If the type of the player is an Impostor, add the player to the impostors list.

- registerTask
    - This method should take in a `Task` object and a `Room` object.
    - It should add the task to the list associated with the room object in the `rooms` attribute.
    - However, you must first check that an equivalent `Task` object does not exist in **any** of the other lists. If this task exists in another list, return `"This task has already been registered."`
      Note: the keys of the `rooms` attribute are the **names** of the `Room` objects, not the objects themselves.

- gameOver
    - This method takes in no arguments.
    - The method checks how many crewmates are alive and how many impostors are alive.
    - If no crewmates are alive, return `"Defeat! All crewmates have been eliminated."`
    - Else, if no impostors are alive, return `"Victory! All impostors have been eliminated."`
    - Otherwise, return `"Game is not over yet!"`

Grading Rubric

Function	Points	Function	Points
Task: __init__	5	Impostor: eliminate	5
Task: __eq__	5	Impostor: vote	5
Crewmate: __init__	5	Impostor: __str__	5
Crewmate: doTask	5	AmongUs: __init__	5
Crewmate: vote	10	AmongUs: registerPlayer	10
Crewmate: callMeeting	20	AmongUs: registerTask	10
Impostor: __init__	5	AmongUs: gameOver	5
		Total:	**100**

Provided

The `HW10.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW10.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can re-submit this assignment an unlimited number of times until the deadline; just click the "Re-submit" button at the lower right-hand corner of Gradescope. You do not need to submit your `HW10.py` on Canvas.