Homework 4:

Dining Hall

For this assignment, you will be completing methods so that students can successfully enter, order, and pay for food at a dining hall. You will also be correcting test cases.

Review the starter code thoroughly before beginning this assignment. Understanding how the classes interact with each other is important. Take notes or draw a diagram if necessary.

Overview

Student Class

The *Student* class represents a student who can order food at a dining hall. Each student object has 3 instance variables:

name - a string representing a student's name,

blue_bucks - a float showing how many blue bucks are on the student's mcard, and a

location - the dining hall object if the student is in a dining hall and otherwise **None**

The *Student* class also includes several methods:

__init__ - which initializes the student's attributes,

__str__ **-** which returns the student's name,

buy_blue_bucks(self, amount) - which adds the passed amount to **blue_bucks** and doesn't return anything

enter_dining_hall(self, dinning_hall) - which takes a dining hall object and returns a boolean value,

leave_dining_hall(self) - which returns a boolean value, and

order_food(self, food_list) - which takes a list of food objects in the order and returns a boolean value

Food Class

The *Food* class represents a food option at a dining hall. A food object has 2 instance variables:

name - a string representing the food's name

price - a float representing the food's price

The *Food* class includes 2 methods:

__init__ - which initializes the food's name and price

__str__ **-** which returns the name and price

Dining_Hall Class

The *Dining_Hall* class (which you will finish implementing) represents the dining hall's food options. Each dining hall object has 4 instance variables:

name - a string which is the name of the dining hall,

capacity - an int representing the maximum number of students allowed in the dining hall,

student_list - a list of student objects representing the students who are in the dining hall, and

inventory - a dictionary with a food object as a key and the current quantity of that food as the value

The *Dining_Hall* class also includes several methods:

__init__ - which initializes the attributes

__str__ - which returns a string with the dining hall name and the number of students in the student_list,

inside(self, student) - which checks if the student is in this dining hall location and returns a boolean value,

can_enter(self, student) **-** which takes a student and returns a boolean value,

add_inventory(self, food, quantity) - which adds the quantity of food to the inventory, and

fill_order(self, food_list) - which takes the list of food to order and returns a boolean value

Tasks to Complete

- **Complete three methods in the *Student* Class**

  Complete the ***enter_dining_hall*** method.

  1) First check if the student's location is not **None** and if it is not then print "Already in a dining hall" and return **False**.
  2) Otherwise call ***can_enter*** on the **dining_hall**. If the result is **False** print "Too full - try later" and return **False**. Otherwise change the student's **location** to the dining hall and add them to the **student_list** for the dining hall, and then return True.

  Complete the ***leave_dining_hall*** method.

  1) If the student's location is **None** print "Not in a dining hall" and return **False**
  2) Otherwise remove the student from the dining hall's **student_list** and set the student's **location** to **None** and return **True**

  Complete the ***order_food*** method.

  1) Calculate the total cost of the order by totaling the **price** for each food object in the **food_list**.
  2) Check if the student has the total cost or more in their **blue_bucks**. If they do not have enough money then print "Insufficient funds" and return **False**.
  3) If they have enough money, check if their **location** is **None**. If it is **None** print "You must be in a dining hall to order food" and return **False**.
  4) Call ***fill_order(food_list)*** on the dining hall object.If ***fill_order*** returns **True** remove the total cost from **blue_bucks** and return **True**. Otherwise, return **False**.

- **Complete two methods in the *Dining_Hall* class**

  Complete the ***add_inventory*** method that takes a **Food** object and a **quantity.** If the food is already in the inventory then add the passed **quantity** to the existing value. Otherwise set the value for the food in the **inventory** dictionary to the **quantity**.

  Complete the ***fill_order*** method that takes a list of food in an order and checks that there is enough food in the **inventory** for the order and if not returns **False**. Otherwise it subtracts the food item from the **quantity** in the **inventory** and returns **True**.

- **Fix Test Cases**
  - Note: Many test cases have already been written for you. **Please do not edit test cases outside of the one below.** As you are working on one test case, feel free to comment out the test cases that you are not working on, but be sure to uncomment all test cases before you turn in your homework.
  - ***test_dining_hall_entrace_and_exit_corner_cases*** has three test cases. The first one has the correct expected values, but the other two do not. Fix the tests to use the correct values.

Grading Rubric (60 points)

Note that if you use hardcoding (specify expected values directly) in any of the methods by way of editing to get them to pass the test cases, or you edit any test cases other than the ones you have been directed to, you will NOT receive credit for those related portions.

Note - use the provided methods you will not earn credit if you implement the functionality instead.

- 30 points for correctly implementing the *Student* class (10 per method).
- 20 points for correctly completing the *Dining_Hall* class (10 per method).
- 10 points for fixing ***test_dining_hall_entrace_and_exit_corner_cases*** (5 per case).

Extra Credit (6 points)

To gain extra credit on this assignment, please complete the following task:

Create a **lottery** method for in the *Dining_Hall* class, which randomly selects one student currently in the dining hall, and gives the student a $10 blue bucks reward on their mcard. If there's no student in the dining hall, then nothing happens. Also, add a test function (***test_lottery***) to test your new method.

For ***test_lottery,*** you are required to test two cases:

1) If there's no student in the dining hall, nothing happens
2) If there are several students, make sure that there's one student who gets the reward.