L7-L9 Homework: Hotel

Problem Description

Hello! Please make sure to read all parts of this document carefully.

You will be writing a Hotel class to help with managing guest check in, calculating payment for each day, and managing guest check out after the last payment. To do this, you will create and turn in a Hotel.java file with two required methods. You are allowed to create additional helper methods (and we recommend you do), but the only required methods are the ones described.

Solution Description

You will write a **single file, Hotel.java**. You **must use each form of iteration shown in L7-L8 at least once**.

Hotel.java

main

The program receives the number of floors and rooms per floor the hotel will have by console arguments, in that order. If the values (which are guaranteed to be integers) are invalid (a valid hotel has at least one floor with at least one room), the program will print "Invalid number of floors or rooms." (with an ending newline) and terminate. Otherwise, it will continue with the instructions below to request the cost per night of each room and then start processing commands.

The hotel has floors in ascending order starting at 1, and rooms on each floor that also start at 1. For example, in a hotel that has two floors and three rooms per floor, we would have “floor 1, room 1”, “floor 1, room 2”, “floor 1, room 3”, “floor 2, room 1”, “floor 2, room 2”, and “floor 2, room 3”.

For each floor, in ascending order, the program will request the cost for the floor, expecting in the input line a sequence of integers. The input will always be integers and will always be the correct number of integers for one floor. If all of those are valid (a cost value must be positive), those will become the cost of each room, otherwise, the program will print "Room costs must be positive." and prompt the user again for the costs of that floor (all of that floor but not of any other). See the example output for details.

After determining the cost of each room, the program will be receiving commands. It will print "> " and wait for a command. Commands are determined by the first word in the line and always come in the correct format (for example, if a command expects one String and three

integers, the upcoming values after the first word are guaranteed to be text without spaces and then three integers, but the integers might be values that are invalid in that context). There are five, commands: in, nd (short for next day), price, print, quit.

- The in command receives information and attempts to check in a guest. It receives the guest's name (a simple word, no last name, and no whitespace), the duration of the stay, the floor that they will be at, and the room on that floor they will be at. For example, "in Ignacio 3 2 4" will attempt to check in Ignacio for a duration of 3 days, staying on floor 2, room 4.
    - If the guest is already checked in (there is a guest in any room with the same name), the program will print "[guest name] already checked in."
    - Otherwise, if the floor or room doesn't match an existing room, it will print "Invalid floor or room."
    - Otherwise, if the guest tries to check in for less than a single day, it will print "Guests must stay at least one day."
    - Otherwise, if another guest is in the desired room, it will print "Room is already occupied."
    - Otherwise, it will check in the guest, and print "Checking in [guest name] to floor [floor], room [room], for [days] [day/days].", for example, "Checking in Ignacio to floor 2, room 4, for 3 days."
- The nd command doesn't receive any values. It indicates that one night was completed. It will calculate the number of guests that night and the total payment received from them (each guest pays the value of their room, as determined at the beginning of the program), followed by checking out all guests whose stay ends.
    - It will print "Total payment from [guest count] [guest/guests]: [price in U.S. format].". For example, "Total payment from 1 guest: $200.00.", "Total payment from 2 guests: $350.00.", "Total payment from 0 guests: $0.00." While the total payment will always be an integer, it should always be formatted with a dollar sign and two decimals.
    - After that, it will check out all guests who were on the last day of their stay (for each guest, you should remember how many days left they have in their stay). It will go in ascending order of floor, then in ascending order of room number. If the guest in that room is checking out, it will print "Checking out [guest name] from floor [floor], room [room].", for example, "Checking out Ignacio from floor 2, room 4."
    - Note: the second required method of the homework will be of use for this command. We recommend reviewing that method and implementing it before this code.
- The price command will print out the cost of a room. It receives the floor number and the room number on that floor, and prints the cost, with the line "The price for floor [floor], room [room] is [price in U.S. format] per day.", for example, "The price for floor 2, room 3 is $175.00 per day.". If any of the values are invalid (the passed values don't match an existing room) it will instead print "Invalid floor or room."
- The print command will print out which guests are in which room. It will print the floors in descending order (so that when you look at the output, the bottom floor is floor 1),

and in each line, it will print the rooms in order, with the name of the guest that it's staying if there is one, and a single space if there is none. A bar, |, will be added at the beginning and end of each line, as well as between rooms. For example, if in a hotel with two floors and 3 rooms per floor, if we have Siam on floor 1, room 2, and Ameerah on floor 2, room 3, it will print:

```
| | |Ameerah|
| |Siam| |
```

- The quit command ends the program. All other commands should not terminate the program and instead should re-prompt the user with "> " and wait for a new command.

Based on the requirements of the program, you will need to remember, for each room on each floor, the following things:

- Which guest, if any, is in that room
- How much the room costs per day
- If there is a guest, how many days of their stay they have left

You should keep three rectangular 2D arrays to keep track of the values.

calculatePayment

calculatePayment is a method that achieves some of the objectives of the nd command in main. It will receive two 2D arrays, one storing the guests in the hotel (2D String array) and another determining the cost of that room (2D int array). Both will be of the same shape and will be rectangular and non-empty. While there are many ways of representing the hotel rooms in a 2D array, and you can choose any for your implementation of main, in the arrays that you will receive in this method all elements will be rooms (there will be no sentinel elements, if you are familiar with that term). Elements in the first array will be non-null if there is a guest, and if so, you should add the cost of the corresponding entry in the second array to the total payment. The method should return the total payment that would be received from the guests, as an int.

Example Input/Output

Example Output 1 – User Input is Bolded. Your program should look exactly like this.
Command: java Hotel 2 3

```
Costs for floor 1: 2 4 8
Costs for floor 2: 3 5 7

> price 2 3
The price for floor 2, room 3 is $7.00 per day.
> in Ignacio 4 2 3
Checking in Ignacio to floor 2, room 3, for 4 days.
> print
| | |Ignacio|
| | | |
> in Max 2 1 1
Checking in Max to floor 1, room 1, for 2 days.
> nd
Total payment from 2 guests: $9.00.
> nd
Total payment from 2 guests: $9.00.
Checking out Max from floor 1, room 1.
> nd
Total payment from 1 guest: $7.00.
> nd
Total payment from 1 guest: $7.00.
Checking out Ignacio from floor 2, room 3.
> nd
Total payment from 0 guests: $0.00.
> print
| | | |
| | | |
> quit
```

Example Output 2 – User Input is Bolded. Your program should look exactly like this.
Command: java Hotel 3 2

```
Costs for floor 1: 1 1
Costs for floor 2: -2 4
Room costs must be positive.
Costs for floor 2: 3 0
Room costs must be positive.
Costs for floor 2: 2 2
Costs for floor 3: 3 3

> in Siam 3 3 3
Invalid floor or room.
> in Siam 1 1 1
Checking in Siam to floor 1, room 1, for 1 day.
> in Siam 2 2 2
Siam already checked in.
> in Ameerah 1 3 2
Checking in Ameerah to floor 3, room 2, for 1 day.
> print
| |Ameerah|
| | |
|Siam| |
> quit
```

Example Output 3 – User Input is Bolded. Your program should look exactly like this.
Command: java Hotel 0 0

```
Invalid number of floors or rooms.
```

Rubric

[100] Hotel.java

- [80] Multiple scenarios for main. Each scenario is scored all-or-nothing, depending on whether the input/output matches the solution
- [10] calculatePayment test cases
- [10] Other checks
    - [2.5 x4] Each form of iteration in L7-L8 appears at least once

We reserve the right to adjust the rubric, but this is typically only done for correcting mistakes.

Allowed Imports

To prevent trivialization of the assignment, you are only allowed to import **java.util.Scanner**. You are not allowed to import any other classes or packages.

Feature Restriction

There are a few features and methods in Java that overly simplify the concepts we are trying to teach. For that reason, do not use any of the following in your final submission:

- **var** (the reserved keyword)

Checkstyle

The Checkstyle deduction limit for this assignment is 5 points, not counting Javadoc errors. Refer to the course guide for the use of Checkstyle and review the autograder Checkstyle report on your submissions.

Collaboration

Collaboration Statement

To ensure that you acknowledge a collaboration and give credit where credit is due, we require that you place a collaboration statement as a comment at the top of at least one .java file that you submit. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

If the autograder encounters issues processing your collaboration statement, begin it with "Collaboration Statement: ".

Allowed Collaboration

When completing homeworks for CS 1331 you may talk with other students about:

- What general strategies or algorithms you used to solve problems in the homeworks
- Parts of the homework you are unsure of and need more explanation
- Online resources that helped you find a solution
- Key course concepts and Java language features used in your solution

You may **not** discuss, show, or share by other means the specifics of your code, including screenshots, file sharing, or showing someone else the code on your computer, or use code shared by others.

Examples of approved/disapproved collaboration:

- approved:
    - "Hey, I'm really confused on how we are supposed to implement this part of the homework.
    - What strategies/resources did you use to solve it?"
- disapproved:
    - "Hey, it's 10:40 on Thursday... Can I see your code? I won't copy it directly I promise"

In addition to the above rules, note that it is not allowed to upload your code to any sort of public repository. This could be considered an Honor Code violation, even if it is after the homework is due.

Turn-In Procedure

Upload the files listed below to the corresponding assignment on Gradecope:

- Hotel.java

For each submission, you will be able to see the results of a few basic test cases on your code. Each test typically corresponds to a rubric item, and the score returned represents the

performance of your code on those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

Important Notes

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit .class files or .jar files
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Piazza for a note containing all official clarifications