Homework 2

CS 1331

Due Friday September 20th, 2013 11:59 PM

1 Introduction

Welcome back everybody! I hope everyone had a fun time working on the anagram homework last week. This week, we'll be writing our first program with classes and arrays. In this assignment, we'll be writing what I like to call Evil Hotel Simulator 2013.

2 Problem Description

You are Mr. (or Ms.!) Evil, and you are the owner and sole proprietor of your very own hotel. Luckily, you've chosen a fantastic location; your hotel is the only place to stay in the small town of BumbleVille, New Mexico. Your hotel is small, and can only hold 10 guests at a time – but you want to make the most money in as little time as you can. You have a complete and total monopoly on the traveler market, and you're using it to your advantage in a number of ways.

1. When checking into your hotel, guests are required to tell you exactly how many days they are staying. They cannot leave in **fewer** than that number of days, but they can extend their stay.
2. You reserve the right to change the nightly rate of your hotel at any time, for any reason.
3. You know a man who works at the only airport in the state (GildedAir), and have a deal with him. If you give him the signal, he can delay the departure of every flight leaving BumbleVille by one day. This in turn forces your guests to stay an extra night (which they must pay you for, of course – you're not running a charity, after all!).

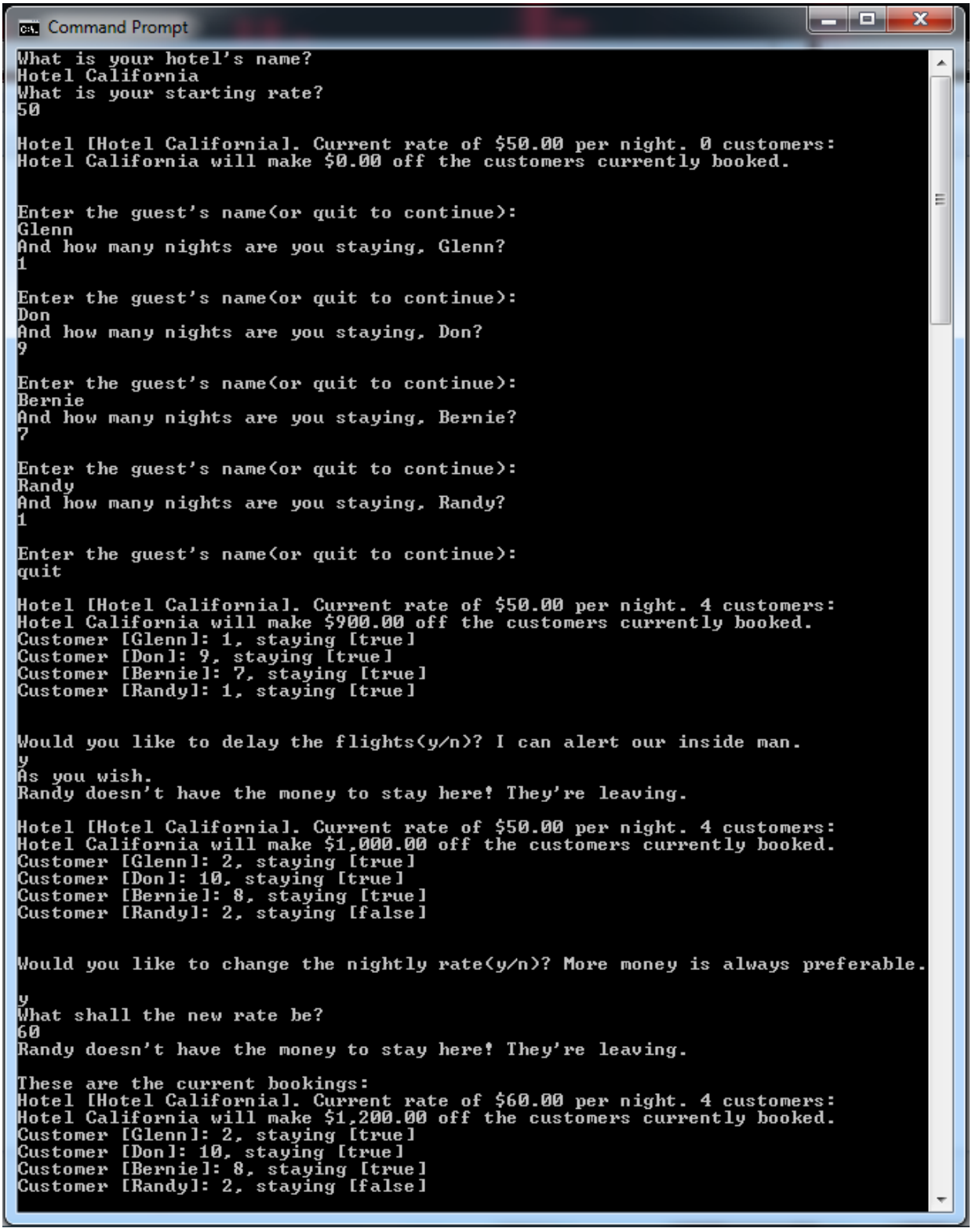
Your task is this: to write a reasonable simulation of this behavior to figure out exactly how much money the hotel is making at any point in time. You must do so in an object-oriented fashion, using a minimum of two (2) classes: a Guest class and a Hotel class. You are required to have classes and objects manage their own internal data, and instance variables should have proper visibility modifiers. Finally, you'll also need a Driver class which sets up your hotel and runs your simulation. Here is a breakdown of what each of your classes should have:

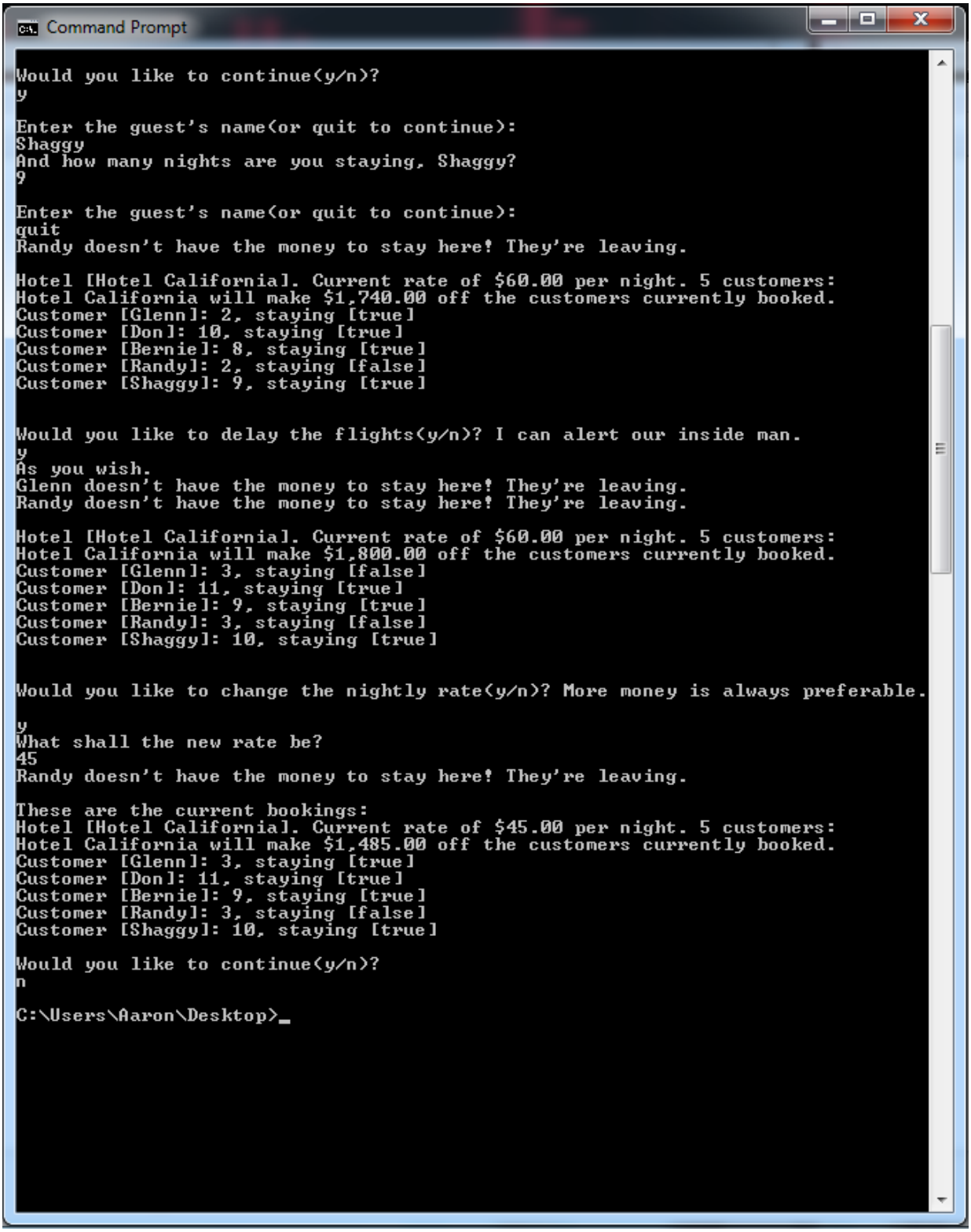
- Guest
    - Variables
        - All guests have a name.
        - All guests have a number of nights they are staying at the hotel.
        - All guests have a random amount of money (between $50-$250 per night they are staying at the hotel).
        - All guests keep track of whether they are staying at your hotel.
    - Methods
        - All guests should have a **getter** for the number of night they are staying, but **not a setter.**
        - All guests should have a **constructor** which sets up their name and number of nights staying.
        - All guests should have a method that increments the number of nights they are staying in the hotel (for example, if their flight got delayed…).
        - All guests should have a method that returns the total cost of their stay as a **double**. Since the customers do not know the price until they check out, the method must take in the nightly rate (also as a double) and use it for calculating the total cost of staying.
        - All guests should have a **getter** for whether they are staying or not.
        - All guests should have a **toString** method that returns a String representation of that guest. It should be formatted like such: `Guest["Aaron"] staying[4] nights.`

- Hotel
    - Variables
        - Your hotel has a name.
        - Your hotel keeps track of the current nightly rate as a **double.**
        - Your hotel keeps track of the current Guests in an **array.**
        - Your hotel should also keep track of the current number of guests.
    - Methods
        - Your hotel class should have a **constructor** that takes in a name for your hotel and the starting nightly rate.
        - Your hotel class should have both a **getter** and a **setter** for the nightly rate.
        - Your hotel class should have a method that adds a Guest to your array if there is any space. If there isn't any space in your hotel, then you should iterate through your Guests and kick out any that cannot pay the current fee, replacing them with the new Guest.
        - Your hotel class should have a method that calculates the total amount of money your guests are going to give you (for all nights they are staying) using the current rate. It should return a **double** which corresponds to the total money amount you'll be making.
        - Your hotel class should have a method to delay the flights and force all your Guests to stay an extra night.
        - Finally, your hotel class should also have a **toString** method which returns a String representation of the Hotel. It should use the toString of each Guest in addition to printing out the Hotel's information, which consists of the hotel name, current rate, and amount of money the hotel is currently making.
- Driver
    - This is where your main method stays. It should start by asking the user questions to set up the hotel, such as what the hotel's name is and starting rate. It should then prompt the user for a Guest name and number of nights that guest is staying and add them to the hotel.
    - After the user is done entering guests, it should print out the current amount of money and ask the user if they want to delay the flights. If the user enters yes, it should delay the flights and print out the new profits. It should then ask the user if they would like to change the nightly rate – if so, it should take that in from the user, set the appropriate

variable in your Hotel object, and print out the new profits. Then it should ask the user if they would like to return to the entering guests task – if so, the loop begins again.

Here is an example of what your program should look like when run.




Command Prompt
Would you like to continue(y/n)?
y
Enter the guest's name(or quit to continue):
Shaggy
And how many nights are you staying, Shaggy?
9
Enter the guest's name(or quit to continue):
quit
Randy doesn't have the money to stay here! They're leaving.
Hotel [Hotel California]. Current rate of $60.00 per night. 5 customers:
Hotel California will make $1,740.00 off the customers currently booked.
Customer [Glenn]: 2, staying [true]
Customer [Don]: 10, staying [true]
Customer [Bernie]: 8, staying [true]
Customer [Randy]: 2, staying [false]
Customer [Shaggy]: 9, staying [true]
Would you like to delay the flights(y/n)? I can alert our inside man.
y
As you wish.
Glenn doesn't have the money to stay here! They're leaving.
Randy doesn't have the money to stay here! They're leaving.
Hotel [Hotel California]. Current rate of $60.00 per night. 5 customers:
Hotel California will make $1,800.00 off the customers currently booked.
Customer [Glenn]: 3, staying [false]
Customer [Don]: 11, staying [true]
Customer [Bernie]: 9, staying [true]
Customer [Randy]: 3, staying [false]
Customer [Shaggy]: 10, staying [true]
Would you like to change the nightly rate(y/n)? More money is always preferable.
y
What shall the new rate be?
45
Randy doesn't have the money to stay here! They're leaving.
These are the current bookings:
Hotel [Hotel California]. Current rate of $45.00 per night. 5 customers:
Hotel California will make $1,485.00 off the customers currently booked.
Customer [Glenn]: 3, staying [true]
Customer [Don]: 11, staying [true]
Customer [Bernie]: 9, staying [true]
Customer [Randy]: 3, staying [false]
Customer [Shaggy]: 10, staying [true]
Would you like to continue(y/n)?
n
C:\Users\Aaron\Desktop>

2.1 Useful Tips

- Use the Numberformat class to correctly turn your double money values into formatted money values. It will save you a lot of headache and is better OO style.
- Use the Random class to get random integers between 50 and 250. Hint: the nextInt(int max) returns numbers in what range?
- One of the first things you should do is write your toString() methods. This will allow you to debug your code much more easily as you'll actually be able to see what values the variables have as you go along.
- Occasionally, the Scanner class might seem to "skip" a method call to .nextLine(), .next(), .nextDouble(), etc. This is caused by the Scanner class leaving a trailing newline character and having it eaten up by the next call. To fix it, just insert an empty call to nextLine() to "eat up" that garbage data so you can continue, or use Double.parseDouble() or Integer.parseInt().
- Writing a single method at a time and testing that method will make programming this assignment much easier in the long run. Each individual piece is relatively simple, but there are a lot of them! If you come across a bug without checking each part individually, it can be very difficult and time-consuming to figure out where the error is.

2.2 Program Requirements

In summary, your program should:

- Prompt the user for information to build the Hotel.
- Prompt the user for guest information.
- Ask the user if they would like to delay the flights.
- Ask the user if they would like to change the rates.
- Print out the hotel state in between each of these operations.
- Have an array of Guests as instance data in Hotel.
- Use some form of NumberFormat object to format the money output of your program.
- Your driver may be in a separate class called HotelDriver.java or as the main method in Hotel.java, whichever you prefer.

2.3 Turn-in Procedure

Turn in the following files to T-Square. When you're ready, double-check that you have *submitted* and not just saved as draft.

- Guest.java
- Hotel.java
- (optionally) HotelDriver.java

Make sure you are submitting your .java files, and not any .class files.

3 Verify the Success of Your HW Turn-In

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
    a. It helps insure that you turn in the correct files.
    b. It helps you realize if you omit a file or files.**

       (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)

    c. Helps find last minute causes of files not compiling and/or running.

**Note: Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight Wednesday. Do not wait until the last minute!