C#.NET ASSESSMENT 4 - Practice

Task: Create a contact manager for a car salesperson which shows the next phone call the salesperson should make and allows the salesperson to log a new phone call.

Build Specifications:

- Clone the classroom repo and create a new Java project within the local cloned repo. (Be sure to find the correct location for your project within your local repo folder.)
- Create a Java class. This first one is called **co.grandcircus.Call** (i.e. package: co.grandcircus, class: Call) and will hold information about a single phone call. It will have the following specification:
    - Include two **private** member variables:
        i. **date**: an integer. Note that while Java has classes for dates and times, to keep this simple *just use an integer*. Dates will be stored as an integer in yyyymmdd format, as in 20240731, which corresponds to 2024 July 31. By saving dates this way, it's easy to sort them and find the minimum or maximum.
        ii. **notes**: a string. This is where the salesperson can log a note about the phone call.
    - Getters and Setters, all **public**:
        i. Create a getter for the date member. Do not create a setter.
        ii. Create a getter for the notes member. Do not create a setter.
        iii. Note: both members will be set in the constructor, and without a setter, can't be changed later. This is intentional.
    - Create a **public** constructor that takes two parameters, date and notes, and saves them in the member variables respectively.

- Create a second Java class. This one is called **co.grandcircus.Customer** and has the following specifications:
    - Include three **private** member variables:
        i. **name**: a string, the name of the customer
        ii. **telephone**: a string, the customer's telephone number
        iii. **calls**: An ArrayList that holds instances of the Call class. This will contain the call log for this particular customer.
    - Create a **public** getter for both the name and the telephone members. Do not include any setters. Do not add a getter or setter for the calls member.

- Add a **public** constructor that takes two parameters: a name and a telephone. (Do not include a parameter for calls.)
    i. Set both member variables respectively.
    ii. Then initialize the **calls** member to a new empty ArrayList that holds Call instances.
- Create two additional **public** methods:
    i. **addCall**. Returns void. Takes two parameters, date and notes.
        1. Use the two parameters, date and notes, to create a new instance of Call.
        2. Add this new instance to the calls ArrayList.
    ii. **getLastCall**. Returns an instance of Call. Has no parameters.
        1. Locate the most recent Call member of the calls list and return it. (Hint: Because you're adding new calls to the end of the list, you can just grab the last in the list. You don't need to check the dates of the calls, as that will be the most recent.)

- Create a third Java class. This one is called **co.grandcircus.SalesPerson** with the following specifications:
    - Include three **private** member variables:
        i. **name**: a string, the name of the salesperson such as "Fred".
        ii. **company**: a string, the name of the company the salesperson works for.
    - Create **public** getters for the name and company members. Do not include a getter or setter for the customers list.
    - Create a constructor that takes two parameters, name and company, and:
        i. Initializes the respective member variables.
        ii. Initialize the **customers** member to an empty ArrayList.
    - Create two additional methods:
        i. **addCustomer**. Returns void. Takes as a parameter an instance of Customer and adds it to the customers ArrayList.
        ii. **getNextCustomerToCall**. Returns an instance of Customer.
            1. This function will go through the list of customers and locate the customer with the oldest most recent phone call.
            2. Hint 1: use the getLastCall() Customer method to determine the most recent phone call of each customer.

3. Hint 2: The dates are stored as simple integers, so think about how you can find the customer with the oldest recent phone call.
4. Hint 3: Add some test code to your App's main method and write some code to test out that this method works correctly. This is how professional developers work. Test the method before using it.

- In the App.java's **main** method:
    - Create a new instance of SalesPerson. His name is Fred and he works for Grand Circus Auto Sales.
    - Create an initial set of call data for the salesperson. Take this slowly; start with the first customer and follow these steps *carefully*:
        i. Create a new customer instance with name Jane Vasquez, and telephone number 555-5678. (Hint: You can call the variable Jane if you like.)
        ii. Call that customer's addCall() method, with date 20240601and notes "Scheduled visit for tomorrow".
        iii. Add that customer to Fred's SalesPerson instance using the addCustomer method.
    - Repeat the previous step for the following customers (You can create a new variable each time, or reuse the variable from the first customer, whichever you like):

        Customer: Name "Jamal Smith", Phone "555-1234"
        Call: Date 20240701, Notes "Left phone message"

        Customer: Name "Bob Johnson", Phone "555-9012"
        Call: Date 20240709, Notes "Might be interested soon"

        Customer: Name "Alice Williams", Phone "555-3456"
        Call: Date 20240501, Notes "Interested in purchasing"

        Customer: Name "Tom Davis", Phone "555-7890"
        Call: Date 20240106, Notes "Purchased a vehicle"

        Customer: Name "Emily Wilson", Phone "555-2345"
        Call: Date 20240312, Notes "Scheduled product demo"

- Now for the fun stuff. Create a variable that holds a Customer (just name it **next**) and then call Fred's getNextCustomerToCall() method and save the result in the variable. Then:
    i. Print out information about the customer stored in the **next** variable:
        1. Print out "Hello Fred of Grand Circus Auto Sales. Here is your next call." (But don't hardcode either name or company. Use the SalesPerson's name and phone number.
        2. Print out the customer's name.
        3. Print out the customer's phone number.
    ii. Use the same customer and add a new call with date 20240801 and notes "Interested in an extended warranty". (Note: By adding a new call, this customer will no longer be the one with the oldest last call!)
- Do it again!
    i. Call getNextCustomerToCall() method, save it in the same **next** variable, and print the same information as the previous step.
    ii. Add a new call to that customer with date 20240802, and notes "Needs repair work"
- And finally, do only part of it again:
    i. Call getNextCustomerToCall() one more time and print the same information as before.
    ii. Don't add a new call. Fred is done for the week!

Hints

- Follow the above steps in order! Finish one class before moving on to the next.
- After finishing each class, you might write some test code in your App's main method to make sure the methods in the class work. Then you can comment out this test code when you're done.

Example output

```
Hello Fred of Grand Circus Auto Sales. Here is your next call.
Tom Davis
555-7890
Hello Fred of Grand Circus Auto Sales. Here is your next call.
Emily Wilson
555-2345
Hello Fred of Grand Circus Auto Sales. Here is your next call.
```

Alice Williams
555-3456