Learners Global School

NexTurn Corporate Training

Programming using C# - Assessment 1

Entork - Energy Utility App

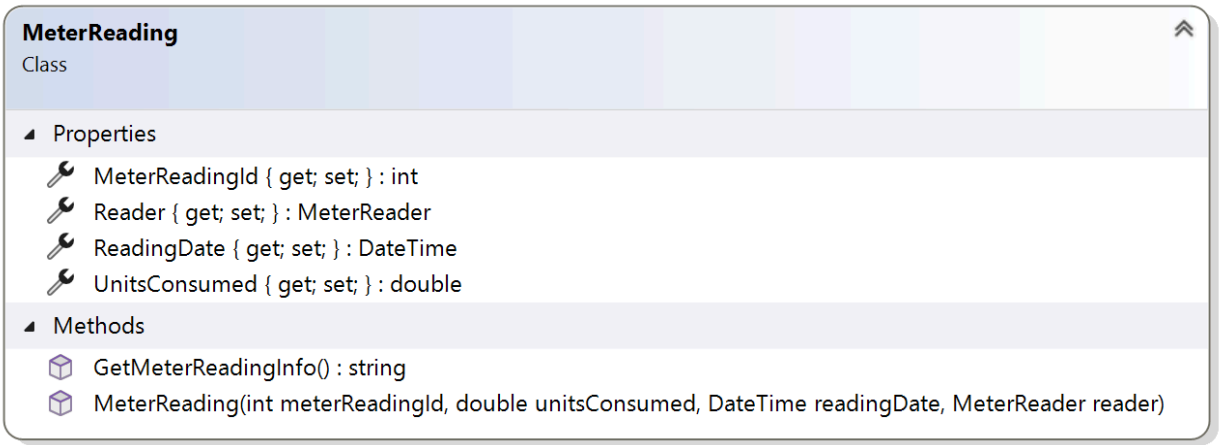
Objective:

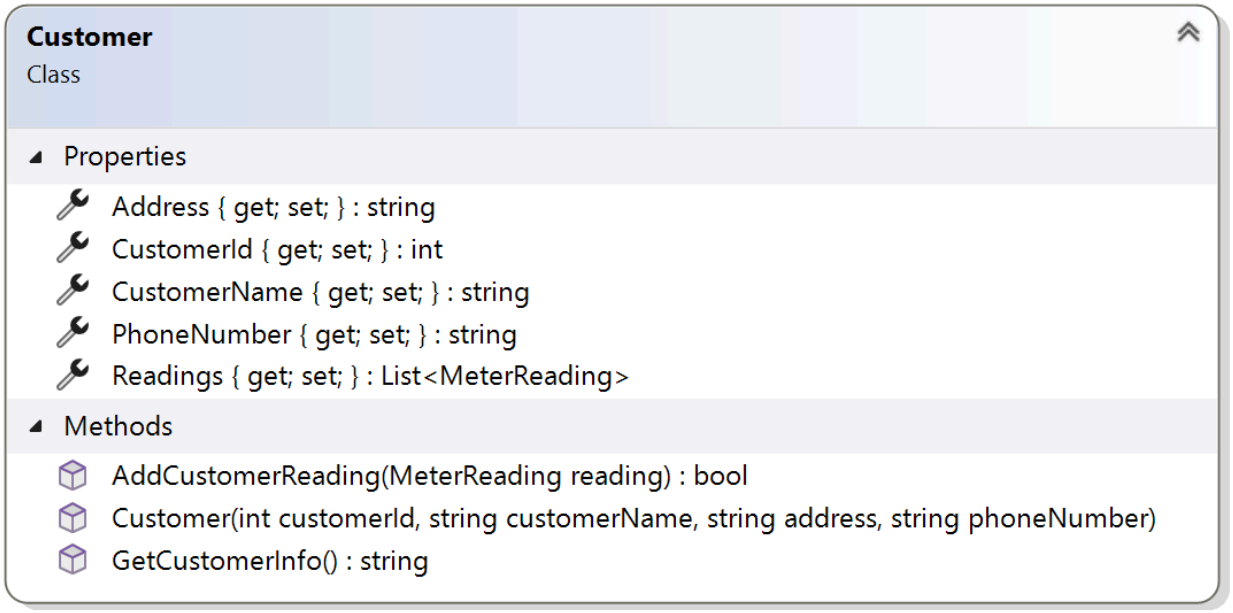
EntorkApp is a C# console application that simulates an Energy and Utility management system. It models core functionalities such as customer management, meter readings, bill generation, employee management, and bill payments.

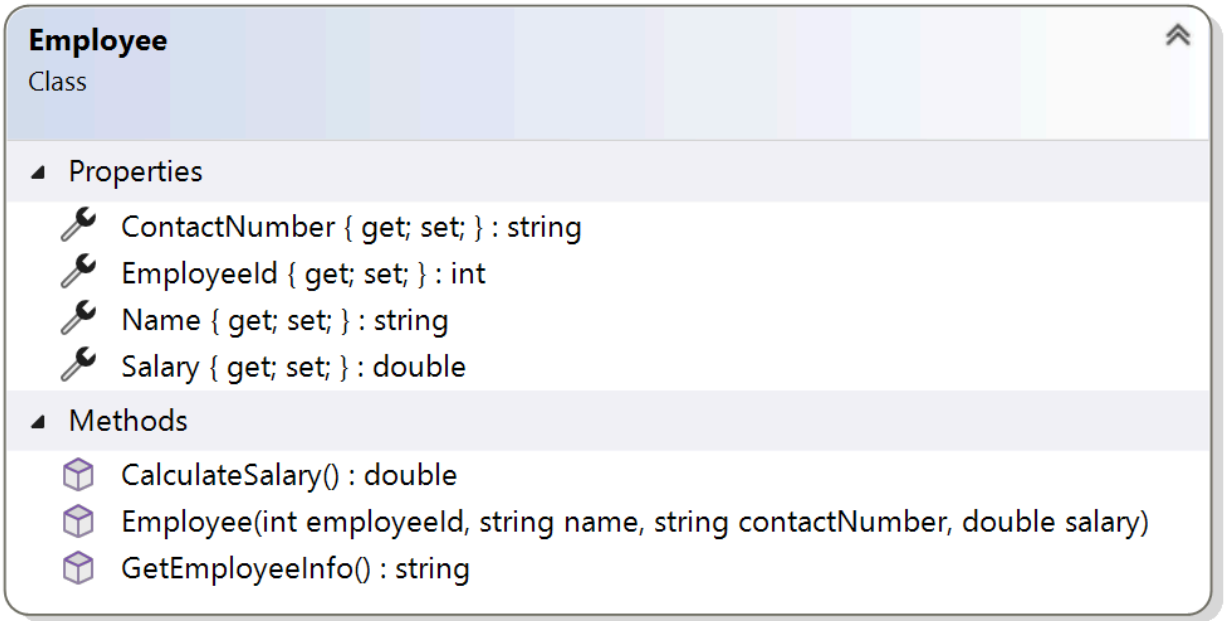
Solution Structure:

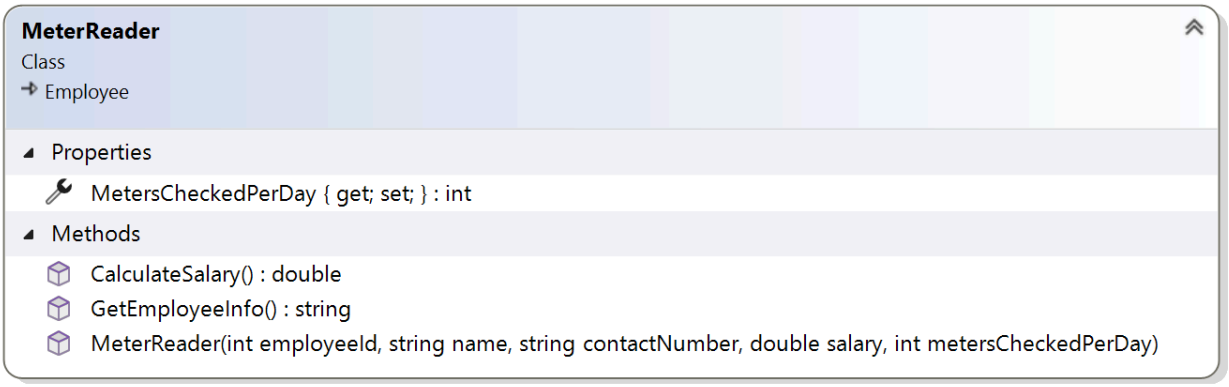
- **Solution Name:** EntorkApp
- **Class Library:** EntorkBusinessLayer
- **Console Application:** EntorkConsoleApp

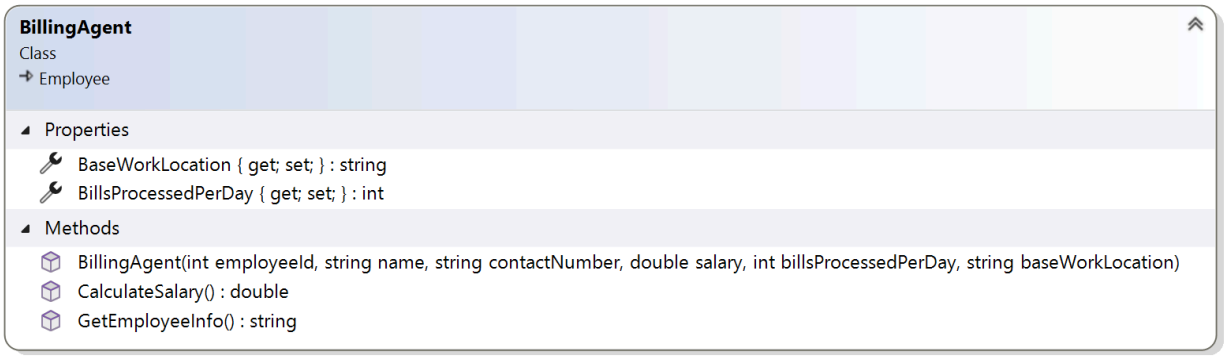
Class Diagram:

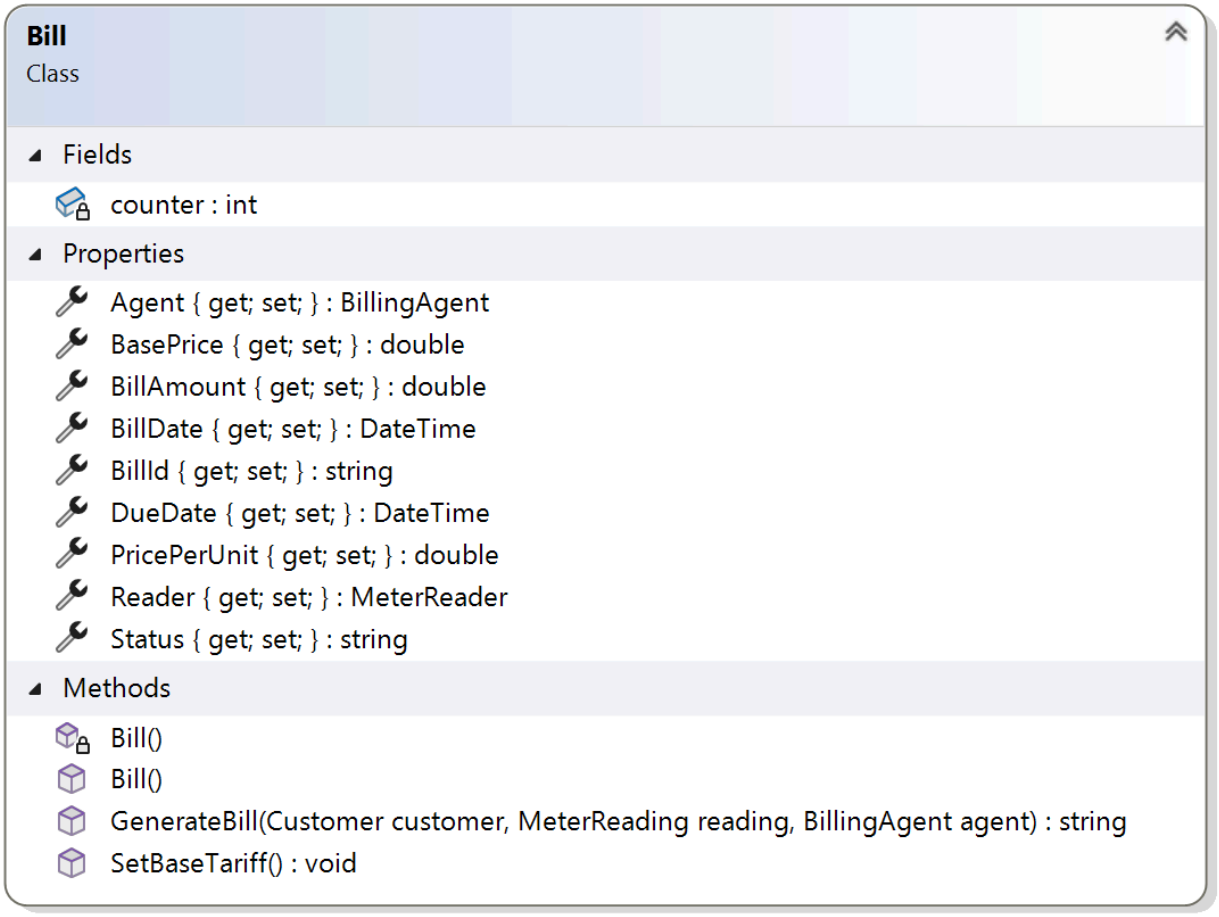



Customer
Class
Properties
Address { get; set; } : string
CustomerId { get; set; } : int
CustomerName { get; set; } : string
PhoneNumber { get; set; } : string
Readings { get; set; } : List<MeterReading>
Methods
AddCustomerReading(MeterReading reading) : bool
Customer(int customerId, string customerName, string address, string phoneNumber)
GetCustomerInfo() : string

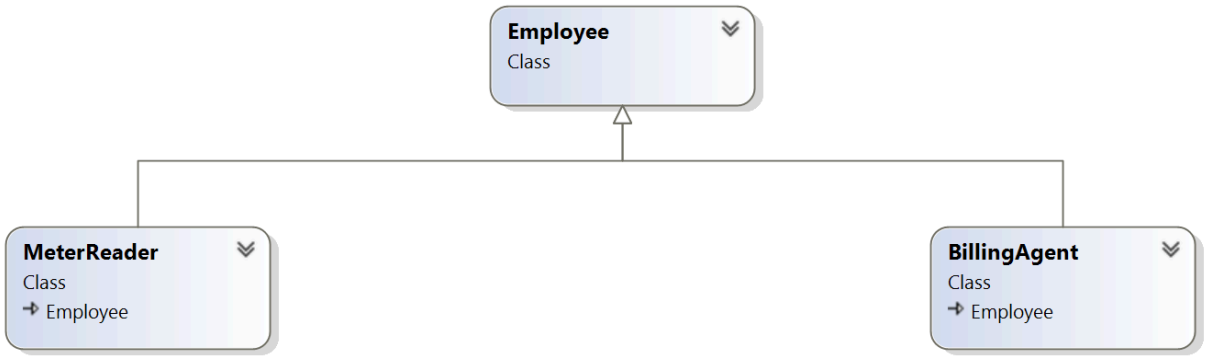

Employee
Class
Properties
ContactNumber { get; set; } : string
EmployeeId { get; set; } : int
Name { get; set; } : string
Salary { get; set; } : double
Methods
CalculateSalary() : double
Employee(int employeeId, string name, string contactNumber, double salary)
GetEmployeeInfo() : string


MeterReader
Class
Employee
Properties
MetersCheckedPerDay { get; set; } : int
Methods
CalculateSalary() : double
GetEmployeeInfo() : string
MeterReader(int employeeId, string name, string contactNumber, double salary, int metersCheckedPerDay)


BillingAgent
Class
Employee
Properties
BaseWorkLocation { get; set; } : string
BillsProcessedPerDay { get; set; } : int
Methods
BillingAgent(int employeeId, string name, string contactNumber, double salary, int billsProcessedPerDay, string baseWorkLocation)
CalculateSalary() : double
GetEmployeeInfo() : string



Class Diagram Relationships:



Assignment Tasks:

Task 1: Create the Visual Studio Solution

1. Open Visual Studio and create a new solution named **EntorkApp**.
2. Add a **Class Library** project named **EntorkBusinessLayer**.
3. Add a **Console Application** project named **EntorkConsoleApp**.
4. Add a reference to **EntorkBusinessLayer** in **EntorkConsoleApp**.

Task 2: Create Classes in EntorkBusinessLayer

1. Class Customer

- **Properties:**
  - CustomerId (int)
  - CustomerName (string)
  - Address (string)
  - PhoneNumber (string)
  - Readings (List<MeterReading>)
- **Constructor:** Initialize CustomerId, CustomerName, Address and PhoneNumber. Initialize Readings to a new List<MeterReading>.
- **Methods:**
  - string GetCustomerInfo() → Method which returns customer details as a concatenated string (Refer output screenshot).
  - bool AddCustomerReading(MeterReading reading) → Adds the reading to the Readings list of this customer. Returns true.

2. Class Employee

- **Properties:**
  - EmployeeId (int)

    - Name (string)
    - ContactNumber (string)
    - Salary (double)
- **Constructor:** Initialize EmployeeId, Name, ContactNumber and Salary.
- **Methods:**
    - string GetEmployeeInfo() → Virtual method which returns employee details as a concatenated string (Refer output screenshot).
    - double CalculateSalary() → Virtual method which returns Salary of the employee.

3. Class MeterReader

- **Inherits from the base class Employee**
- **Properties:**
    - MetersCheckedPerDay (int)
- **Constructor:** Initialize EmployeeId, Name, ContactNumber and Salary of the base class. Initialize MetersCheckedPerDay.
- **Methods:**
    - string GetEmployeeInfo() → Overridden method which returns employee details as a concatenated string (Refer output screenshot).
    - double CalculateSalary()
        - Overridden method to calculate Salary of a MeterReader Employee.
        - Invoke the base class method CalculateSalary() to get totalSalary.
        - Check if MetersCheckedPerDay is greater than 10
            - If, calculate the additional number of meters (greater than 10).
            - Calculate totalSalary using the following formula:
                - totalSalary + (extraMeters * 250)
        - Assign totalSalary value to Salary property.

        - Return totalSalary.

4. Class BillingAgent

- **Inherits from the base class Employee**
- **Properties:**
    - BillsProcessedPerDay (int)
    - BaseWorkLocation (string)
- **Constructor:** Initialize EmployeeId, Name, ContactNumber and Salary of the base class. Initialize BillsProcessedPerDay and BaseWorkLocation .
- **Methods:**
    - string GetEmployeeInfo() → Overridden method which returns employee details as a concatenated string (Refer output screenshot).
    - double CalculateSalary()
        - Overridden method to calculate Salary of a MeterReader Employee.
        - Invoke the base class method CalculateSalary() to get totalSalary.
        - Check if BaseWorkLocation is "New York" or "London".
            - If yes,
                - If BillsProcessedPerDay is greater than 20
                    - Calculate totalSalary as "totalSalary + (BillsProcessedPerDay * 100)"
                - Else if BillsProcessedPerDay is greater than 10
                    - Calculate totalSalary as "totalSalary + (BillsProcessedPerDay * 75)"
                - Else,
                    - Calculate totalSalary as "totalSalary + (BillsProcessedPerDay * 50)"
        - Else,
            - If BillsProcessedPerDay is greater than 20

        - Calculate totalSalary as "totalSalary + (BillsProcessedPerDay * 80)"
    - Else if BillsProcessedPerDay is greater than 10
        - Calculate totalSalary as "totalSalary + (BillsProcessedPerDay * 60)"
    - Else,
        - Calculate totalSalary as "totalSalary + (BillsProcessedPerDay * 40)"
- Assign totalSalary value to Salary property.
- Return totalSalary.

5. Class MeterReading

- **Properties:**
    - MeterReadingId (int)
    - UnitsConsumed (double)
    - ReadingDate (DateTime)
    - Reader (MeterReader)
- **Constructor:** Initialize MeterReadingId, UnitsConsumed, ReadingDate and Reader.
- **Methods:**
    - string GetMeterReadingInfo() → Method which returns meter reading details as a concatenated string (Refer output screenshot).

6. Class Bill

- **Properties:**
    - BillId (string)
    - BasePrice (double)
    - PricePerUnit (double)
    - BillDate (DateTime)
    - DueDate (DateTime)

    - Status (string)
    - BillAmount (double)
    - Reader (MeterReader)
    - Agent (BillingAgent)
- **Field:** counter (int) - private static variable to auto-generate BillId as B1001, B1002, B1003…
- **Static Constructor -** Initialize the static variable counter accordingly.
- **Parameterless Constructor:** Invoke the method SetBaseTariff() of the Bill class.
- **Methods:**
    - void SetBaseTariff()
        - Assign BasePrice as 100.
        - Assign PricePerUnit as 4.
    - string GenerateBill(Customer customer, MeterReading reading, BillingAgent agent)
        - Method which generates the bill for the given customer with the given reading by the given agent.
        - Implement exception handling in this method.
        - Check if any Readings are available for the customer.
            - If yes, check if the given reading parameter is available as a reading in the list Readings of the customer.
            - If found, generate a bill as follows:
                - Assign the Reader value from the given reading parameter to the Reader property of the Bill.
                - Assign the given agent to the Agent property of the Bill.
                - Generate the BillId using the static variable counter as B1001, B1002, B1003 etc.
                - Set BillDate as the current date time.
                - Set DueDate as 30 days from the current date time.
                - Set the bill Status as “Pending”.

        - If UnitsConsumed for the given reading is 0, then set BillAmount as BasePrice.
        - Else if, UnitsConsumed for the given reading is greater than 0, then set the BillAmount using the following formula:
            - BasePrice + (reading.UnitsConsumed + PricePerUnit)
        - Return a string message as
            - Bill successfully generated for Customer : <customer name> with Bill ID : <bill Id>.
    - If the reading is not available or not found, return the message as "Bill could not be generated".
    - In case of any exception, return the message "Something went wrong, please try again later!"

Task 3: EntorkConsoleApp Implementation

Program.cs Code:

```
using EntorkBusinessLayer;

namespace EntorkConsoleApp
{
    internal class Program
    {
        static void Main(string[] args)
        {
            #region Customer One

            Console.WriteLine("----------------------------Customer One Bill Generation----------------------------");

            Customer customerOne = new Customer(101, "Lily Elizabeth", "London", "9999999999");
```

```
        MeterReader readerOne = new MeterReader(1, "Philip
Dunphy", "8888888888", 18000, 15);

        MeterReading readingOne = new MeterReading(1, 145,
DateTime.Now, readerOne);

        //Assign reading to customer
        customerOne.AddCustomerReading(readingOne);

        //Display details
        Console.WriteLine("Customer Details: ");
        Console.WriteLine(customerOne.GetCustomerInfo());


Console.WriteLine("-----------------------------------------------------
----------------------------");

        Console.WriteLine("Customer Readings: ");
        foreach (var item in customerOne.Readings)
        {
            Console.WriteLine(item.MeterReadingId + " " +
item.UnitsConsumed + " " + item.ReadingDate);
            Console.WriteLine(".............................................");
        }


Console.WriteLine("-----------------------------------------------------
----------------------------");

        //Calculate Reader Salary
        readerOne.CalculateSalary();

        Console.WriteLine("Meter Reader Details: ");
        Console.WriteLine(readerOne.GetEmployeeInfo());
```

```
Console.WriteLine("------------------------------------------------------------
---------------------------");

        BillingAgent agentOne = new BillingAgent(2, "Margaret
Townsend", "7777777777", 22000, 20, "London");

        //Calculate Agent Salary
        agentOne.CalculateSalary();

        Console.WriteLine("Billing Agent Details: ");
        Console.WriteLine(agentOne.GetEmployeeInfo());


Console.WriteLine("------------------------------------------------------------
---------------------------");

        Bill billOne = new Bill();

        Console.WriteLine("Generating Bill...");
        string message = billOne.GenerateBill(customerOne,
readingOne, agentOne);

        Console.WriteLine(message);


Console.WriteLine("------------------------------------------------------------
---------------------------");

        Console.WriteLine("Bill Details");

        Console.WriteLine("Bill Id : " + billOne.BillId);
        Console.WriteLine("Bill Amount : " + billOne.BillAmount);
        Console.WriteLine("Bill Date : " + billOne.BillDate);
```

```
            Console.WriteLine("Due Date : " + billOne.DueDate);
            Console.WriteLine("Bill Status : " + billOne.Status);
            Console.WriteLine("Base Price  : " + billOne.BasePrice);
            Console.WriteLine("Price Per Unit : " + billOne.PricePerUnit);
            Console.WriteLine("Meter Reader : " + billOne.Reader.Name);
            Console.WriteLine("Billing Agent : " + billOne.Agent.Name);


Console.WriteLine("------------------------------------------------------
-----------------------------");

            #endregion

            #region Customer Two

            Console.WriteLine("----------------------Customer Two Bill
Generation----------------------");

            Customer customerTwo = new Customer(102, "Alex Pritchett",
"New York", "9876789687");

            MeterReading readingTwo = new MeterReading(2, 89,
DateTime.Now, readerOne);

            //Assign reading to customer
            customerTwo.AddCustomerReading(readingTwo);

            //Display details
            Console.WriteLine("Customer Details: ");
            Console.WriteLine(customerTwo.GetCustomerInfo());


Console.WriteLine("------------------------------------------------------
-----------------------------");
```

```
            Console.WriteLine("Customer Readings: ");
            foreach (var item in customerTwo.Readings)
            {
                Console.WriteLine(item.MeterReadingId + " " +
item.UnitsConsumed + " " + item.ReadingDate);
                Console.WriteLine(".....");
            }


Console.WriteLine("-----------------------------------------------------
---------------------------");

            Bill billTwo = new Bill();

            Console.WriteLine("Generating Bill...");
            message = billTwo.GenerateBill(customerTwo, readingTwo,
agentOne);

            Console.WriteLine(message);


Console.WriteLine("-----------------------------------------------------
---------------------------");

            Console.WriteLine("Bill Details");

            Console.WriteLine("Bill Id : " + billTwo.BillId);
            Console.WriteLine("Bill Amount : " + billTwo.BillAmount);
            Console.WriteLine("Bill Date : " + billTwo.BillDate);
            Console.WriteLine("Due Date : " + billTwo.DueDate);
            Console.WriteLine("Bill Status : " + billTwo.Status);
            Console.WriteLine("Base Price  : " + billTwo.BasePrice);
            Console.WriteLine("Price Per Unit : " + billTwo.PricePerUnit);
            Console.WriteLine("Meter Reader : " + billTwo.Reader.Name);
            Console.WriteLine("Billing Agent : " + billTwo.Agent.Name);
```

```
Console.WriteLine("-------------------------------------------------------------
----------------------------");

        #endregion
    }
  }
}
```

Expected Output:

```
------------------------------Customer One Bill Generation-----------------------------
Customer Details:
101 Lily Elizabeth London 9999999999
--------------------------------------------------------------------------------------
Customer Readings:
1 145 02-02-2025 12:00:59
.............................................
--------------------------------------------------------------------------------------
Meter Reader Details:
1 Philip Dunphy 8888888888 19250 15
--------------------------------------------------------------------------------------
Billing Agent Details:
2 Margaret Townsend 7777777777 23500 London 20
--------------------------------------------------------------------------------------
Generating Bill...
Bill successfully generated for Customer : Lily Elizabeth with Bill ID : B1001
--------------------------------------------------------------------------------------
Bill Details
Bill Id : B1001
Bill Amount : 249
Bill Date : 02-02-2025 12:00:59
Due Date : 04-03-2025 12:00:59
Bill Status : Pending
Base Price  : 100
Price Per Unit : 4
Meter Reader : Philip Dunphy
Billing Agent : Margaret Townsend
--------------------------------------------------------------------------------------
```

```
----------------------Customer Two Bill Generation----------------------
Customer Details:
102 Alex Pritchett New York 9876789687
-------------------------------------------------------------------------------
Customer Readings:
2 89 02-02-2025 12:00:59
.........................................
-------------------------------------------------------------------------------
Generating Bill...
Bill successfully generated for Customer : Alex Pritchett with Bill ID : B1002
-------------------------------------------------------------------------------
Bill Details
Bill Id : B1002
Bill Amount : 193
Bill Date : 02-02-2025 12:00:59
Due Date : 04-03-2025 12:00:59
Bill Status : Pending
Base Price  : 100
Price Per Unit : 4
Meter Reader : Philip Dunphy
Billing Agent : Margaret Townsend
-------------------------------------------------------------------------------
```

**********All the best**********