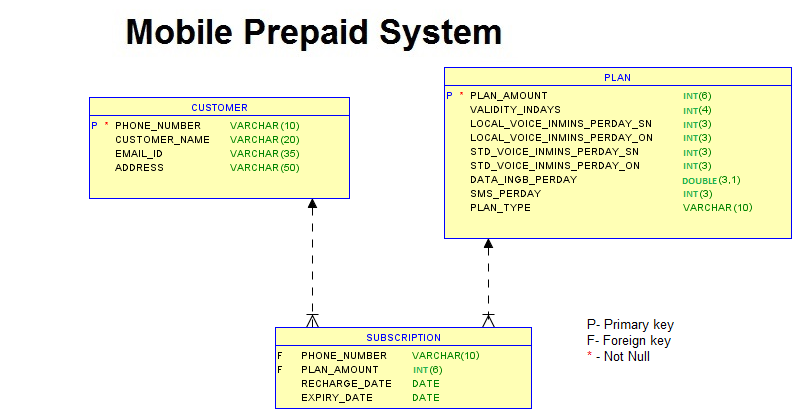
1 . Write a query to display the customer name and plan amount of all the subscribers, if and only if the year of the recharge is 2019 or 2022.

Sort the records based on the customer name in ascending order.



Answer

SELECT CUSTOMER\_NAME,PLAN\_AMOUNT

FROM CUSTOMER C JOIN SUBSCRIPTION S

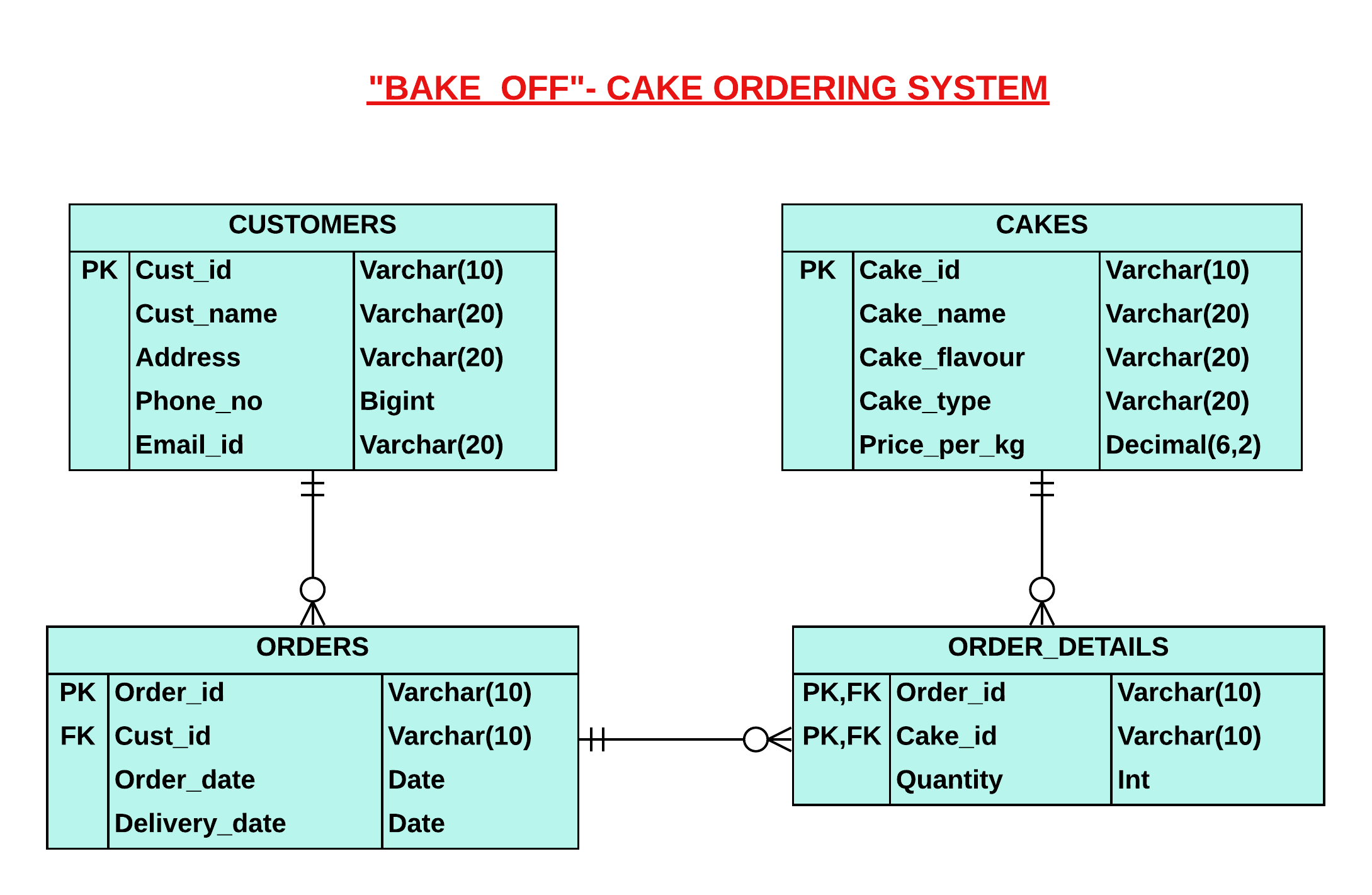
ON C.PHONE\_NUMBER=S.PHONE\_NUMBER

WHERE YEAR(RECHARGE\_DATE)=2019 OR YEAR(RECHARGE\_DATE)=2022

ORDER BY C.CUSTOMER\_NAME;

2 .The shop manager likes to take a list of customers who ordered cakes from 'February' to 'August'. Write a query to display the unique customer name, email id and phone number of all customers.

Sort the results based on the customer name in ascending order.



Answer

SELECT DISTINCT Cust\_name,Email\_id,Phone\_no

FROM CUSTOMERS C JOIN ORDERS O

ON C.CUST\_ID=O.CUST\_ID

WHERE MONTH(ORDER\_DATE) BETWEEN 2 AND 8

ORDER BY CUST\_NAME;

**3 .Scenario:**

Win Cinemas is an inclusive theater that caters to everyone in our community. They take pride in enriching all their community through exceptional and daring theatrical performances, as well as engaging educational experiences. With their commitment to excellence, they guarantee a world-class theatrical journey enhanced by the mesmerizing 4K Dolby Atmos technology.  They need a software to maintain movie screening details.  
  
As their software consultant, you can help them by developing a C# application.

**Functionalities:**

In class **Movie**, implement the below-given properties.

|  |  |
| --- | --- |
| **Data Type** | **Property Name** |
| string | MovieName |
| string | ScreenedDate |
| string | RemovedDate |
| double | Price |

In class **Program,**

**public static Dictionary<int,Movie> screeningDetails**-In the code template, it is already provided. 

Implement the below-given method.

|  |  |  |
| --- | --- | --- |
| **Class** | **Method** | **Description** |
| Program | public Dictionary<string, double> MovieScreenedMoreNumberOfDays() | This method is used to find the movie which is screened more number of days using the screened and removed date from **screeningDetails**dictionary.  Then store the **movie name**and **price**in **Dictionary**and return it. |
| Program | public Dictionary<string, double> MovieWithScreenedDays() | This method is used to find all the movie as well as total number of screened days.  Then store the **movie name** and that movie's **total number of screened days**  as  **Dictionary**and return it. |

**Note**: The date format is **(MM/dd/yyyy).**

In **Program** class, **Main** method,

**1.**Get the values from the **user**.

**2.**Call the methods accordingly and display the result.

**3.**     In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

**Note:**

* Keep the properties, methods and classes as **public.**
* Please read the method rules **clearly**.
* Do not use **Environment.Exit()** to terminate the program.
* Do not change the given code template.

Assume these below-given data as example,

If you need you can try this in your code also.

**Sample Input / Output:**

1. Movie screening more number of days

2. Movie with their screening days

3. Exit

Enter your choice

**1**

Avatar 150

1. Movie screening more number of days

2. Movie with their screening days

3. Exit

Enter your choice

**2**

Eternals  35

Iron Man  31

Avatar     103

LightYear  20

Black Panther  24

1. Movie screening more number of days

2. Movie with their screening days

3. Exit

Enter your choice

**3**

Thank You

Answer

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinCinemas //DO NOT change the namespace name

{

public class Program //DO NOT change the class name

{

public static Dictionary<int,Movie> screeningDetails = new Dictionary<int,Movie>();

//Implement the methods as per the description

public static Dictionary<string,double> MovieScreenedMoreNumberOfDays()

{

int totalDays = 0;

foreach(KeyValuePair<int,Movie> x in screeningDetails)

{

DateTime d1 = DateTime.Parse(x.Value.RemovedDate);

DateTime d2 = DateTime.Parse(x.Value.ScreenedDate);

TimeSpan dt = d2-d1;

if(totalDays<Math.Abs(dt.Days))

{

totalDays = Math.Abs(dt.Days);

}

}

Dictionary<string,double> newDict = new Dictionary<string,double>();

foreach(KeyValuePair<int,Movie> x in screeningDetails)

{

DateTime d1 = DateTime.Parse(x.Value.RemovedDate);

DateTime d2 = DateTime.Parse(x.Value.ScreenedDate);

TimeSpan dt = d1-d2;

if(dt.Days==totalDays)

{

newDict.Add(x.Value.MovieName,x.Value.Price);

}

}

return newDict;

}

public static Dictionary<string,double> MovieWithScreenedDays()

{

Dictionary<string,double> newDict = new Dictionary<string,double>();

foreach(KeyValuePair<int,Movie> x in screeningDetails)

{

DateTime d1 = DateTime.Parse(x.Value.RemovedDate);

DateTime d2 = DateTime.Parse(x.Value.ScreenedDate);

TimeSpan dt = d1-d2;

newDict.Add(x.Value.MovieName,dt.Days);

}

return newDict;

}

public static void Main(string[] args) //DO NOT change the method signature

{

screeningDetails.Add(1,new Movie{MovieName="Eternals",ScreenedDate="04/25/2020",RemovedDate="05/30/2020",Price=350});

screeningDetails.Add(2,new Movie{MovieName="Iron Man",ScreenedDate="07/15/2008",RemovedDate="08/15/2008",Price=100});

screeningDetails.Add(3,new Movie{MovieName="Avatar",ScreenedDate="10/25/2003",RemovedDate="02/05/2004",Price=150});

screeningDetails.Add(4,new Movie{MovieName="LightYear",ScreenedDate="07/10/2020",RemovedDate="07/30/2020",Price=175});

screeningDetails.Add(5,new Movie{MovieName="Black Panther",ScreenedDate="05/17/2020",RemovedDate="06/10/2020",Price=200});

while(true)

{

Console.WriteLine("1. Movie screening more number of days\n2. Movie with their screening days\n3. Exit");

int opt = Convert.ToInt32(Console.ReadLine());

switch(opt)

{

case 1:

Dictionary<string,double> newDict1 = MovieScreenedMoreNumberOfDays();

foreach(KeyValuePair<string,double> x in newDict1)

{

Console.WriteLine(x.Key+" "+x.Value);

}

break;

case 2:

Dictionary<string,double> newDict2 = MovieWithScreenedDays();

foreach(KeyValuePair<string,double> x in newDict2)

{

Console.WriteLine(x.Key+" "+x.Value);

}

break;

case 3: Console.WriteLine("Thank You");

return ;

}

}

}

}

}

Movie.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinCinemas //DO NOT change the namespace name

{

public class Movie //DO NOT change the class name

{

//Implement code here

public string MovieName {get;set;}

public string ScreenedDate {get;set;}

public string RemovedDate {get;set;}

public double Price {get;set;}

public Movie(){}

public Movie(string MovieName,string ScreenedDate,string RemovedDate,double Price)

{

this.MovieName=MovieName;

this.ScreenedDate=ScreenedDate;

this.RemovedDate=RemovedDate;

this.Price=Price;

}

}

}

**Scenario:**

There are various types of mobile apps and web apps that greatly assist customers of a well-known software company in the United States, which provides products to its customers. Recently, the company received an order from a client for a tax amount calculator. The employee provides their ID and salary, and the company wishes to display their tax amount based on the salary.

 As their software consultant, you need to help them by developing a C# application.

**Functionalities:**

In class **Employee,**implement the below-given properties.

|  |  |
| --- | --- |
| **Datatype** | **Property Name** |
| string | EmployeeId |
| double | Salary |

In class **EmployeeUtility,**implement the below-given methods and also **Inherit** the class **Employee**.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public bool ValidateEmployeeId() | This method is used to validate the employee id.  Condition :  1. The employee id length should be 4.  2. It should be in the format of a capital letter followed by 3 digits  **For Example :**E123  If the above-given conditions are passed, then return **true**. Otherwise return **false.** |
| public double CalculateTaxAmount() | This method is used to calculate the tax amount of the employee based on their salary and return it.  If the employee's salary is up to 20,000, then the tax amount is **0**.  If the employee's salary is between 20,001 and 50,000, the first 20,000 is tax-free, after that, the tax rate is **10%**.  If an employee's salary is between 50,001 and 100000, the first 20,000 is tax-free, after that, the tax rate is **10%**, and after 50,000, the tax rate is **20%**.  If an employee's salary is above 100000, the first 20,000 is tax-free, after that, the tax rate is **10%,**and after 50,000, the tax rate is **20%,** and after 100000, the tax rate is **30%**. |

**For Example :**

1. **Employee Salary =  45000.**

     The first 20000 is tax free, and the remaining amount of 25000 is taxed at 10%.

     So, **Tax Amount**= (Salary - 20000) \* 0.10

2. **Employee Salary =  65000.**

     The first 20,000 is tax-free, the next 30,000 is taxed at 10%, and the remaining 15,000 is taxed at 20%.

     So, **Tax Amount** = (20000 \* 0) + (30000 \* 0.10) + ((Salary - 50000) \* 0.20)

3. **Employee Salary =  155000.**

     The first 20,000 is tax-free, the next 30,000 is taxed at 10%, the next 50,000 is taxed at 20% and the remaining 55,000 is taxed at 30%.

     So, **Tax Amount** = (20000 \* 0) + (30000 \* 0.10) + (50000 \* 0.20) + ((Salary - 100000) \* 0.30)

In **Program** class - **Main** method,

**1.**Get the values from the **user**as per the Sample Input.

**2.**Call the **ValidateEmployeeId**method, If it returns true then get the **Salary**value from the **user**and move on to step 3, If it returns false then display **Invalid employee id.**

**3.**Use the values in method**CalculateTaxAmount**and display the result as per the Sample Output.

**Note:**

* Keep the properties, methods and classes as **public.**
* Please read the method rules **clearly**.
* Do not use **Environment.Exit()** to terminate the program.
* Do not change the given code template.

**Sample Input 1:**

Enter the employee id

**H456**

Enter the salary

**15000**

**Sample Output 1:**

Tax amount to pay is 0

**Sample Input 2:**

Enter the employee id

**A234**

Enter the salary

**200000**

**Sample Output 2:**

Tax amount to pay is 43000

**Sample Input 3:**

Enter the employee id

**A123456**

**Sample Output 3:**

Invalid employee id

**Sample Input 4:**

Enter the employee id

**e123**

**Sample Output 4:**

Invalid employee id

Answer

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace IncomeTax //DO NOT change the namespace name

{

public class Program //DO NOT change the class name

{

public static void Main(string[] args) //DO NOT change the 'Main' method signature

{

//Implement your code here

Console.WriteLine("Enter the employee id");

string id = Console.ReadLine();

Console.WriteLine("Enter the salary");

double salary = Convert.ToDouble(Console.ReadLine());

EmployeeUtility EU = new EmployeeUtility(id,salary);

if(EU.ValidateEmployeeId())

Console.WriteLine("Tax amount to pay is "+EU.CalculateTaxAmount());

else

Console.WriteLine("Invalid employee id");

}

}

}

[**Employee.cs**](https://cognizant.tekstac.com/mod/vpl/forms/edit.php?id=105273&userid=132076)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace IncomeTax //DO NOT change the namespace name

{

public class Employee //DO NOT change the class name

{

//Implement your code here

public string EmployeeId {get;set;}

public double Salary {get;set;}

public Employee(){}

public Employee(string EmployeeId,double Salary)

{

this.EmployeeId=EmployeeId;

this.Salary=Salary;

}

}

}

[**EmployeeUtility.cs**](https://cognizant.tekstac.com/mod/vpl/forms/edit.php?id=105273&userid=132076)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace IncomeTax //DO NOT change the namespace name

{

public class EmployeeUtility:Employee //DO NOT change the class name

{

//Implement your code here

public EmployeeUtility(string EmployeeId,double Salary):base(EmployeeId,Salary)

{}

public EmployeeUtility():base(){}

public bool ValidateEmployeeId()

{

Regex reg = new Regex("^[A-Z]{1}[0-9]{3}$");

if(EmployeeId.Length==4 && reg.IsMatch(EmployeeId))

return true;

else

return false;

}

public double CalculateTaxAmount()

{

if(Salary<=20000)

return 0;

else if(Salary>20000 && Salary<=50000)

return (Salary-20000)\*0.1;

else if(Salary>50000 && Salary<=100000)

{

return (30000\*0.1)+((Salary-50000)\*0.2);

}

else if(Salary>100000)

{

return (30000\*0.1)+(50000\*0.2)+((Salary-100000)\*0.3);

}

return 0;

}

}

}

}

**Scenario:**

Get Holiday is a well-known tourist agency and they usually book hotel rooms for their customers based on the hotel rating. Now, manually locating the best hotel is extremely difficult. So, they need an application that can use web scraping techniques to gather information on various hotels and their ratings, and then allow the user to filter and sort through the results to find the best options for their customers.

As their software consultant, you can help them by developing a C# application.

**Functionalities:**

In class **Program,** implement the below-given method.

**public static Dictionary<String, float> hotelDetails**-In the code template, it is already provided.

implement the features listed below.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public Dictionary<String, float> SearchHotel(String hotelName) | This method is used to find the hotel details by hotel name.  If the hotel name is available in the **hotelDetails,**this method should return the hotel name and rating as a **Dictionary**.  If the hotel name is not available in the **hotelDetails ,** then it should return an empty **Dictionary**.  If this method returns an empty **Dictionary,**then print **"Hotel Not Found"** in Main method. |
| public Dictionary<String, float> UpdateHotelRating(string hotelName, float rating) | This method is used to update the rating of the hotel by hotel name.  If the hotel name is available in the **hotelDetails ,**it should  return the hotel name and updated rating as a  **Dictionary**.  If the hotel name is not available in the**hotelDetails ,** then it should return an empty **Dictionary**.  If this method returns an empty **Dictionary,**then print **"Hotel Not Found"** in Main method. |
| public Dictionary<String, float> SortByHotelName() | This method is used to display all the hotels available in the **hotelDetails**in ascending order by hotel name.  The result should be **Dictionary** |

In **Program** class, **Main** method,

**1.**Get the values from the **user**.

**2.**Call the methods accordingly and display the result.

**3.**     In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

**Note:**

* Keep the method and class as **public.**
* Please read the method rules **clearly**.
* Do not use **Environment.Exit()** to terminate the program.
* Do not change the given code template.

**Sample Input / Output:**

1. Search by hotel name

2. Update hotel rating

3. Sort hotels by name

4. Exit

Enter your choice

**1**

Enter the hotel name

**The Hay Adams**

The Hay Adams 3

1. Search by hotel name

2. Update hotel rating

3. Sort hotels by name

4. Exit

Enter your choice

**2**

Enter the hotel name

**Montage Kapalua Bay**

Enter the rating

**4**

Montage Kapalua Bay  4

1. Search by hotel name

2. Update hotel rating

3. Sort hotels by name

4. Exit

Enter your choice

**2**

Enter the hotel name

**Line Bay**

Enter the rating

**5**

Hotel Not Found

1. Search by hotel name

2. Update hotel rating

3. Sort hotels by name

4. Exit

Enter your choice

**3**

Jungle Resort   4.5

Mandarin Oriental   5

Montage Kapalua Bay   4

The Greenwich Hotel    5

The Hay Adams    3

1. Search by hotel name

2. Update hotel rating

3. Sort hotels by name

4. Exit

Enter your choice

**4**Thank You

Answer

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GetHoliday //DO NOT change the namespace name

{

public class Program //DO NOT change the class name

{

public static Dictionary<String, float> hotelDetails = new Dictionary<string, float>();

//Implement the methods as per the description

public static Dictionary<string,float> SearchHotel(string hotelName)

{

Dictionary<string,float> newDict = new Dictionary<string,float>();

foreach(KeyValuePair<string,float> x in hotelDetails)

{

if(hotelName.Equals(x.Key))

{

newDict.Add(x.Key,x.Value);

return newDict;

}

}

return newDict;

}

public static Dictionary<string,float> UpdateHotelRating(string hotelName,float rating)

{

Dictionary<string,float> newDict = new Dictionary<string,float>();

foreach(KeyValuePair<string,float> x in hotelDetails)

{

if(hotelName.Equals(x.Key))

{

newDict.Add(x.Key,rating);

}

}

return newDict;

}

public static Dictionary<string,float> SortByHotelName()

{

List<string> name = new List<string>();

foreach(KeyValuePair<string,float> x in hotelDetails)

{

name.Add(x.Key);

}

name.Sort();

Dictionary<string,float> newDict = new Dictionary<string,float>();

foreach(string x in name)

{

foreach(KeyValuePair<string,float> y in hotelDetails)

{

if(x.Equals(y.Key))

{

newDict.Add(y.Key,y.Value);

}

}

}

return newDict;

}

public static void Main(string[] args) //DO NOT change the method signature

{

hotelDetails.Add("Jungle Resort", 4.5f);

hotelDetails.Add("Mandarin Oriental", 5);

hotelDetails.Add("Montage Kapalua Bay", 4);

hotelDetails.Add("The Greenwich Hotel", 5);

hotelDetails.Add("The Hay Adams", 3);

//Implement code here

while(true)

{

Console.WriteLine("1. Search by hotel name\n2. Update hotel rating\n3. Sort hotels by name\n4. Exit");

int opt = Convert.ToInt32(Console.ReadLine());

switch(opt)

{

case 1:

Console.WriteLine("Enter the hotel name");

string s1 = Console.ReadLine();

Dictionary<string,float> n1 = SearchHotel(s1);

foreach(KeyValuePair<string,float> x in n1)

{

Console.WriteLine(x.Key+" "+x.Value);

}

break;

case 2:

Console.WriteLine("Enter the hotel name");

string s2 = Console.ReadLine();

Console.WriteLine("Enter the rating");

float r = Convert.ToSingle(Console.ReadLine());

Dictionary<string,float> n2 = UpdateHotelRating(s2,r);

if(n2.Count==0)

{

Console.WriteLine("Hotel Not Found");

}

else

{

foreach(KeyValuePair<string,float> x in n2)

{

Console.WriteLine(x.Key+" "+x.Value);

}

}

break;

case 3:

Dictionary<string,float> n3 = SortByHotelName();

foreach(KeyValuePair<string,float> x in n3)

{

Console.WriteLine(x.Key+" "+x.Value);

}

break;

case 4:

Console.WriteLine("Thank You");

return ;

}

}

}

}

}