MOCK2:

VALIDATE CUSTOMER DETAILS:

using System;

using System.Collections.Generic;

using System.Text.RegularExpressions;

class Program

{

    static void Main(string[] args)

    {

        int choice = 1;

        do

        {

            Console.WriteLine("Menu");

            Console.WriteLine("1. Parse Name");

            Console.WriteLine("2. Valid Email");

            Console.WriteLine("3. Play Contact Number");

            Console.WriteLine("4. User Lifetime");

            Console.WriteLine("5. Exit");

            choice = Convert.ToInt32(Console.ReadLine());

            switch (choice)

            {

                case 1 :

                    Console.WriteLine("Enter name:");

                    string name = Console.ReadLine();

                    Program.ParseName(name);

                    break;

                case 2:

                    Console.WriteLine("Enter E-mail id:");

                    string mail = Console.ReadLine();

                    Program.IsValidEmail(mail);

                    break;

                case 3:

                    Console.WriteLine("Enter contact number:");

                    string contact = Console.ReadLine();

                    Program.playContactNumber(contact);

                    break;

                case 4:

                    Console.WriteLine("Enter Created on date(dd-MM-yyyy HH:mm):");

                    string sss = Console.ReadLine();

                    Program.userLifeTime(sss);

                    break;

            }

        } while (choice != 5);

    }

    public static void ParseName(string name)

    {

        string pattern = "[^0-9A-Za-z ]";

        string n = Regex.Replace(name, pattern, " ");

        Console.WriteLine(n);

    }

    public static void userLifeTime(string time)

    {

        DateTime dateOld = DateTime.ParseExact("28-07-2017 09:00", "dd-MM-yyyy hh:mm", null);

        //fill your code

         DateTime dd = DateTime.ParseExact(time, "dd-MM-yyyy HH:mm", null, System.Globalization.DateTimeStyles.None);

         TimeSpan x = dd - dateOld;

         var f = x.TotalMinutes;

         Console.WriteLine("Life time is: {0} minutes",f);

    }

    public static void IsValidEmail(string email)

    {

        //fill your code

        if ((!email.Contains("@.")) && email.Contains("@") && (email.EndsWith("com") || email.EndsWith("org") || email.EndsWith("net")))

            Console.WriteLine("Email id is valid");

        else

            Console.WriteLine("Email is invalid");

    }

    public static void playContactNumber(string phone)

    {

        int i,j,sum=0;

        string[] s = phone.Split('-');

        if (s.Length == 3 && s[0].Length == 3 && s[1].Length == 4 && s[2].Length == 10)

        {

            for (i = 0; i < 3; i++)

            {

                char[] c = s[i].ToCharArray();

                for (j = 0; j < c.Length; j++)

                {

                    if(!char.IsDigit(c[j]))

                        break;

                }

                if (j < c.Length)

                    break;

            }

            if (i < 3)

            {

                Console.WriteLine("Contact number invalid");

            }

            else

            {

                int[] ss = Array.ConvertAll(Array.ConvertAll(s[2].ToCharArray(),Convert.ToString),Convert.ToInt32);

                for (i = 0; i < ss.Length; i++)

                {

                    sum = sum + ss[i];

                }

                while (sum >= 10)

                {

                    string s5 = Convert.ToString(sum);

                    sum=0;

                    int[] temp = Array.ConvertAll(Array.ConvertAll(s5.ToCharArray(), Convert.ToString), Convert.ToInt32);

                    for (i = 0; i < temp.Length; i++)

                    {

                        sum = sum + temp[i];

                    }

                }

                Console.WriteLine("Sum of contact number: "+sum);

            }

        }

        else

            Console.WriteLine("Contact number invalid");

    }

}

OVERLOADING PAYMENT:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

class PaymentUtil

{

    public double MakePayment(Dictionary<string, float> bankTax, string bankName, double amount)

    {

        double tamount=0;

        foreach (var k in bankTax.Keys)

        {

            if (k.Equals(bankName,StringComparison.InvariantCultureIgnoreCase))

            {

                tamount = (amount \* (bankTax[k] / 100))+amount;

            }

        }

        return tamount;

    }

   public double MakePayment(double amount)

    {

        float serviceTax = 5.2f;

        float vat = 2.3f;

        double tamount = 0;

        tamount = amount + (amount \* (serviceTax)/ 100f);

        tamount = tamount + (tamount \* (vat / 100f));

        return tamount;

    }

    public double MakePayment(double amount, float discountPercent)

    {

        double tamount = 0;

        tamount = amount - (amount \* (discountPercent / 100));

        return tamount;

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

class Program

{

    static void Main(string[] args)

    {

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

        onlineBankingDict.Add("ICICI", 4.2f);

        onlineBankingDict.Add("IBRD", 3f);

        onlineBankingDict.Add("IFC", 4.9f);

        onlineBankingDict.Add("HSBC", 3.9f);

        Console.WriteLine("1. Online banking\n2. Credit card\n3. Wallet");

        Console.WriteLine("Enter the choice:");

        Int32 ch = Convert.ToInt32(Console.ReadLine());

        PaymentUtil pu = new PaymentUtil();

        double tamount = 0;

        if (ch == 1)

        {

            Console.WriteLine("Enter the user name:");

            string uname = Console.ReadLine();

            Console.WriteLine("Enter the password:");

            string pass = Console.ReadLine();

            Console.WriteLine("Enter the amount:");

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

            Console.WriteLine("Enter the bank name:");

            string bname = Console.ReadLine();

            Console.WriteLine("Username : {0}",uname);

            Console.WriteLine("Password : {0}",pass);

            Console.WriteLine("Amount : {0}",amt);

            Console.WriteLine("Bank : {0}",bname);

            tamount = pu.MakePayment(onlineBankingDict, bname, amt);

            Console.WriteLine("Total amount(Inclusive of Service Tax): {0:0.00}",tamount);

        }

        else if (ch == 2)

        {

            Console.WriteLine("Enter the account number:");

            string accnum = Console.ReadLine();

            Console.WriteLine("Enter the pin:");

            Int32 pin = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Enter the amount:");

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

            tamount = pu.MakePayment(amt);

            Console.WriteLine("AccountNumber : {0}",accnum);

            Console.WriteLine("Pin : {0}",pin);

            Console.WriteLine("Amount : {0}",amt);

            Console.WriteLine("Total amount(Inclusive of Service Tax and VAT): {0:0.00}", tamount);

        }

        else if (ch == 3)

        {

            Console.WriteLine("Enter the user name:");

            string accnum = Console.ReadLine();

            Console.WriteLine("Enter the password:");

            string pass = Console.ReadLine();

            Console.WriteLine("Enter the amount:");

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

            tamount = pu.MakePayment(amt, 20.2f);

            Console.WriteLine("AccountNumber : {0}",accnum);

            Console.WriteLine("Password : {0}",pass);

            Console.WriteLine("Amount : {0}",amt);

            Console.WriteLine("Discounted amount: {0:0.00}", tamount);

        }

    }

}

FIND CUSTOMER BY ID AND STATE:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

    class Customer

    {

        private long \_id;

        public long Id

        {

            get { return \_id; }

            set { \_id = value; }

        }

        private string \_name;

        public string Name

        {

            get { return \_name; }

            set { \_name = value; }

        }

        private char \_gender;

        public char Gender

        {

            get { return \_gender; }

            set { \_gender = value; }

        }

        private string \_email;

        public string Email

        {

            get { return \_email; }

            set { \_email = value; }

        }

        private string \_contactNumber;

        public string ContactNumber

        {

            get { return \_contactNumber; }

            set { \_contactNumber = value; }

        }

        private DateTime \_createdOn;

        public DateTime CreatedOn

        {

            get { return \_createdOn; }

            set { \_createdOn = value; }

        }

        private Address \_address;

        public Address Address

        {

            get { return \_address; }

            set { \_address = value; }

        }

        public Customer() { }

        public Customer(long id, string name, char gender, string email, string contactNumber, DateTime createdOn,Address address)

        {

            this.\_id = id;

            this.\_name = name;

            this.\_gender = gender;

            this.\_email = email;

            this.\_contactNumber = contactNumber;

            this.\_createdOn = createdOn;

            this.\_address = address;

        }

        public Customer FindCustomerById(List<Customer> customerList, int id)

        {

            Customer c = new Customer();

            int count = 0;

            foreach(Customer cust in customerList)

            {

                if (id == cust.\_id)

                {

                    count = 1;

                    c = cust;

                }

            }

            if (count == 0)

                return null;

            else

                return c;

        }

        public List<Customer> FindCustomerListByState(List<Customer> customerList, string state)

        {

            List<Customer> cust1 = new List<Customer>();

            foreach (Customer c in customerList)

            {

                if (state.Equals(c.Address.State))

                {

                    cust1.Add(c);

                }

            }

            return cust1;

        }

    }

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

    class Address

    {

        private string \_street;

        public string Street

        {

            get { return \_street; }

            set { \_street = value; }

        }

        private string \_city;

        public string City

        {

            get { return \_city; }

            set { \_city = value; }

        }

        private string \_state;

        public string State

        {

            get { return \_state; }

            set { \_state = value; }

        }

        private string \_country;

        public string Country

        {

            get { return \_country; }

            set { \_country = value; }

        }

        private int \_zipCode;

        public int ZipCode

        {

            get { return \_zipCode; }

            set { \_zipCode = value; }

        }

        public Address() { }

        public Address(string street, string city, string state, string country, int zipCode)

        {

            this.\_street = street;

            this.\_city = city;

            this.\_state = state;

            this.\_country = country;

            this.\_zipCode = zipCode;

        }

    }

using System;

using System.Collections.Generic;

class Program

{

     public static void Main(string[] args)

     {

          List<Customer> customerList = new List<Customer>

          {

              new Customer(1, "John Smith", 'M', "[johnsmith@a.com](mailto:johnsmith@a.com)", "+98-7488-8554744596",

                   DateTime.ParseExact("15/02/2017 16:30:00","dd/MM/yyyy HH:mm:ss",null,System.Globalization.DateTimeStyles.None),

                   new Address("15th St", "Buffalo", "New York", "USA", 14220)),

               new Customer(2, "Aekerman", 'M', "[aekerman@a.com](mailto:aekerman@a.com)", "+78-7485-9555874846",

                   DateTime.ParseExact("18/03/2017 15:45:00","dd/MM/yyyy HH:mm:ss",null,System.Globalization.DateTimeStyles.None),

                   new Address("Avenue", "Plano", "Texas", "USA", 75025)),

              new Customer(3, "Madeleine", 'F', "[madeleine@a.com](mailto:madeleine@a.com)", "+78-9855-7488742136",

                   DateTime.ParseExact("22/02/2017 16:45:00","dd/MM/yyyy HH:mm:ss",null,System.Globalization.DateTimeStyles.None),

                   new Address("Parc St", "Lubbock", "Texas", "USA", 79404)),

               new Customer(4, "Edrick", 'M', "[edrick@a.com](mailto:edrick@a.com)", "+99-8787-7844859978",

                   DateTime.ParseExact("15/03/2017 15:45:00","dd/MM/yyyy HH:mm:ss",null,System.Globalization.DateTimeStyles.None),

                   new Address("145th St", "Wasilla", "Alaska", "USA", 99629)),

              new Customer(5, "Tedmond", 'M', "[tedmond@a.com](mailto:tedmond@a.com)", "+88-7844-8854799658",

                   DateTime.ParseExact("15/03/2017 15:45:00","dd/MM/yyyy HH:mm:ss",null,System.Globalization.DateTimeStyles.None),

                   new Address("Port Townsend", "Tacoma", "Washington", "USA", 98412)),

              new Customer(6, "Nelson", 'M', "[nelson@a.com](mailto:nelson@a.com)", "+88-7848-8857488956",

                   DateTime.ParseExact("17/05/2017 10:35:00","dd/MM/yyyy HH:mm:ss",null,System.Globalization.DateTimeStyles.None),

                   new Address("1st St", "Akron", "Ohio", "USA", 44304)),

              new Customer(7, "Dalton", 'M', "[dalton@a.com](mailto:dalton@a.com)", "+88-8879-8854741124",

                   DateTime.ParseExact("01/05/2017 17:25:00","dd/MM/yyyy HH:mm:ss",null,System.Globalization.DateTimeStyles.None),

                   new Address("Lake city", "Newburgh", "New York", "USA", 12550)),

               new Customer(8, "Raymond", 'M', "[raymond@a.com](mailto:raymond@a.com)", "+89-7748-8859112478",

                   DateTime.ParseExact("17/06/2017 08:45:00","dd/MM/yyyy HH:mm:ss",null,System.Globalization.DateTimeStyles.None),

                   new Address("Wall Street", "Texas City", "Texas", "USA", 77591)),

              new Customer(9, "Rosemary", 'F', "[rosemary@a.com](mailto:rosemary@a.com)", "+89-7844-8857489958",

                   DateTime.ParseExact("22/04/2017 16:15:00","dd/MM/yyyy HH:mm:ss",null,System.Globalization.DateTimeStyles.None),

                   new Address("Georgetown", "Olympia", "Washington", "USA", 98506)),

              new Customer(10, "Ruford", 'M', "[ruford@a.com](mailto:ruford@a.com)", "+88-7485-8597448596",

                   DateTime.ParseExact("12/02/2017 09:05:00","dd/MM/yyyy HH:mm:ss",null,System.Globalization.DateTimeStyles.None),

                   new Address("Baker street", "Miles City", "Montana", "USA", 59301))

          };

          Customer customer = new Customer();

          Console.WriteLine("Menu\n1. Find customer by id\n2. Find customer by states\nEnter the choice:");

          switch(Convert.ToInt32(Console.ReadLine()))

          {

              case 1:

                   Console.WriteLine("Enter the Id to find customer:");

                   int id=int.Parse(Console.ReadLine());

              customer=customer.FindCustomerById(customerList,id);

                   if (customer!=null)

                   {

                        Console.WriteLine("Customer Name: {0}\nEmail: {1}\nCity: {2}\nState: {3}\nCountry: {4}\nZip code: {5}",

                        customer.Name,customer.Email,customer.Address.City,customer.Address.State,

                        customer.Address.Country,customer.Address.ZipCode);

                   }

                   else

                        Console.WriteLine("No Customer with that id");

                   break;

              case 2:

                   Console.WriteLine("Enter the state:");

                   string state=Console.ReadLine();

                List<Customer> customerByState=new List<Customer>();

                customerByState=customer.FindCustomerListByState(customerList,state);

                   if(customerByState.Count!=0)

                   {

                        Console.WriteLine("{0,-15}{1,-20}{2,-15}{3,-15}{4}", "Name", "Email", "City", "Country", "Zipcode");

                    foreach (Customer cc in customerByState)

                    {

                        Console.WriteLine("{0,-15}{1,-20}{2,-15}{3,-15}{4}",cc.Name,cc.Email,cc.Address.City,cc.Address.Country,cc.Address.ZipCode);

                    }

                    }

                   else

                        Console.WriteLine("No customer belongs that state");

                   break;

          }

     }

}

BATCH PROCESSING:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

class Customer

{

    private long \_id;

    private string \_name;

    private char \_gender;

    private string \_email;

    private string \_contactNumber;

    private DateTime \_createdOn;

    public Customer(long id, string name, char gender, string email, string contactNumber, DateTime createdOn)

    {

        \_id = id;

        \_name = name;

        \_gender = gender;

        \_email = email;

        \_contactNumber = contactNumber;

        \_createdOn = createdOn;

    }

    public long Id

    {

        get { return \_id; }

        set { \_id = value; }

    }

    public string Name

    {

        get { return \_name; }

        set { \_name = value; }

    }

    public char Gender

    {

        get { return \_gender; }

        set { \_gender = value; }

    }

    public string Email

    {

        get { return \_email; }

        set { \_email = value; }

    }

    public string ContactNumber

    {

        get { return \_contactNumber; }

        set { \_contactNumber = value; }

    }

    public DateTime CreatedOn

    {

        get { return \_createdOn; }

        set { \_createdOn = value; }

    }

    public static List<Customer> PopulateCustomers(List<string> csvList)

    {

        List<Customer> cList = new List<Customer>();

        foreach (string s in csvList)

        {

            string[] ss = s.Split(',');

            string[] format = new string[] {  "dd/MM/yyyy HH:mm:ss" };

            cList.Add(new Customer(long.Parse(ss[0]),ss[1],char.Parse(ss[2]),ss[3],ss[4],DateTime.ParseExact(ss[5],format,null,System.Globalization.DateTimeStyles.None)));

        }

        return cList;

    }

    public static List<Customer> FindCustomerNameFromList(List<Customer> customers, string subString)

    {

        List<Customer> clist = new List<Customer>();

        foreach (Customer c in customers)

        {

            if (c.Name.Contains(subString))

            {

                clist.Add(c);

            }

        }

        return clist;

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

class Program

{

    static void Main(string[] args)

    {

        Console.WriteLine("Enter the number of customer:");

        Int32 n = Convert.ToInt32(Console.ReadLine());

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

        for (int i = 0; i < n; i++)

        {

            Console.WriteLine("Enter the customer "+(i+1)+" detail:");

            customerList.Add( Console.ReadLine());

        }

        List<Customer> custList = new List<Customer>();

        custList = Customer.PopulateCustomers(customerList);

        Console.WriteLine("{0,-10}{1,-10}{2,-10}{3,-10}{4,-10}{5}", "Id", "Name", "Gender", "Email", "Contact no", "Created on");

        foreach (Customer c in custList)

        {

            Console.WriteLine("{0,-10}{1,-10}{2,-10}{3,-10}{4,-10}{5}"

                , c.Id, c.Name, c.Gender, c.Email, c.ContactNumber, c.CreatedOn);

        }

        Console.WriteLine("Enter the substring to search from customer list:");

        string name = Console.ReadLine();

        List<Customer> byName = Customer.FindCustomerNameFromList(custList, name);

        Console.WriteLine("{0,-15}{1,-20}{2,-15}{3,-15}{4,-15}{5}", "Id", "Name", "Gender", "Email", "Contact no", "Created on");

        foreach (Customer c in byName)

        {

            Console.WriteLine("{0,-15}{1,-20}{2,-15}{3,-15}{4,-15}{5}"

                ,c.Id,c.Name,c.Gender,c.Email,c.ContactNumber,c.CreatedOn);

        }

    }

}