**Expense - Requirement 1**

The Accounts team of your organization finds it difficult to keep note of the monthly expenses of the company. They would love to have a software which would help them to store the bills and expenses. Help your team by building an application. There are two major domains Expense and Bill.  
  
**Requirement 1:**

Let’s start off by creating two **Bill** objects and check whether they are equal.

1. Create a **Bill** Class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_soldBy | string |
| \_soldTo | string |
| \_billNo | int |
| \_amount | double |
| \_billDate | DateTime |

1. Mark all the attributes as private
2. Create / Generate appropriate Properties
3. Add a default constructor and a parameterized constructor to take in all attributes in the given order:   
   **Bill ( string \_soldBy, string \_soldTo, int \_billNo, double \_amount,DateTime \_billDate )**
4. When the “Bill” object is printed, it should display the following details: **[Override the ToString method]**  
   Print format:  
   Sold By:"\_soldBy"  
   Sold To:"\_soldTo"  
   Bill No:"\_billNo"  
   Amount:"\_amount"  
   Bill Date:"\_billDate"
5. Two Bills are considered same if they have the same \_**billNo,\_amount, and \_billDate**. Implement the logic in the appropriate function.   
   **[Override the Equals method]**  
     
   The input format consists of Bill details separated by comma in the below order,  
   \_soldBy, \_soldTo, \_billNo, \_amount,\_billDate

The Input to your program would be details of two Bills, you need to display their details as given in "5th point(refer above)" and compare the two Bills and display if the Bills are same or different.  
  
**Problem Overview:**  
The first two line of input consist of a string, that corresponds to the bill details(which is comma seperated). Refer above input format.  
Display the two bill details in Main method using **ToString** method(Refer above format).  
And also check if the two bill are same or different(Use Equals method to compare the two objects).  
Equals method return bool value to Main method(true or false).  
If the Equals method returns true, then print "**Bill 1 is same as Bill 2**".  
If the method returns false, then print "**Bill 1 and Bill 2 are different**".  
  
**Note:**There is an empty line between display statements. Print the empty lines in Mainfunction.  
              **Display one digit after the decimal point for double datatype.**  
  
**Sample Input & Output 1**  
Enter bill 1 details:  
**Harry,Joe,254,2500,08-02-2017**  
Enter bill 2 details:  
**Harry,Joe,254,2500,08-02-2017**  
  
Bill 1  
Sold By:Harry  
Sold To:Joe  
Bill No:254  
Amount:2500.0  
Bill Date:08-02-2017  
  
Bill 2  
Sold By:Harry  
Sold To:Joe  
Bill No:254  
Amount:2500.0  
Bill Date:08-02-2017  
  
Bill 1 is same as Bill 2  
  
**Sample Input & Output 2**  
Enter bill 1 details:  
**Harry,Joe,254,2500,08-02-2017**  
Enter bill 2 details:  
**Oliver,Tina,77,8540,15-12-2017**  
  
Bill 1  
Sold By:Harry  
Sold To:Joe  
Bill No:254  
Amount:2500.0  
Bill Date:08-02-2017  
  
Bill 2  
Sold By:Oliver  
Sold To:Tina  
Bill No:77  
Amount:8540.0  
Bill Date:15-12-2017  
  
Bill 1 and Bill 2 are different

**Expense - Requirement 2**

**Requirement 2:**  
Now we are gonna start creating an Expense and add Bills to it. Start with creating a Expense and use menu-driven approach to add, remove, display details of the Bills in the Expense.  
  
a)Create a Class **Bill** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_soldBy | string |
| \_soldTo | string |
| \_billNo | int |
| \_amount | double |
| \_billDate | DateTime |

Mark all the attributes as private.  
Create / Generate appropriate properties  
Add a default constructor and a parameterized constructor to take in all attributes in the given order:  
**Bill(string \_soldBy, string \_soldTo, int \_billNo, double \_amount, DateTime \_billDate )**  
  
b)Create a Class **Expense** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_name | string |
| \_billList | List<Bill> |

Mark all the attributes as private.  
Create / Generate appropriate properties.  
 Add a default constructor and a parameterized constructor to take in all attributes in the given order:  
**Expense( string \_name, List<Bill> \_billList ).**  
In constructor pass the \_billList value as an empty list. Only one Expense is calculated for a time.  
  
c) Create the following static method in Bill class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public static Bill CreateBill(string detail) | This method accepts a string which contains bill details separated by commas. Split the details and create a bill object from the details and return it. |

The bill details should be given as a comma-separated value in the below order,  
**\_soldBy, \_soldTo, \_billNo, \_amount, \_billDate**  
  
d) Create the following methods in Expense class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public void AddBillToExpense(Bill bill) | This method accepts a bill object and adds the bill to the bill list of the current Expense. |
| public bool RemoveBillFromExpense(int billNo) | This method will get the \_billNo of the bill and delete the bill with the specified billNo from the current Expense. If a bill with the given \_billNo found, delete the bill and return **true**. If a bill with the \_billNo is not found return **false**. |
| public void DisplayBills() | This method will display the bill list in the current Expense. If the bill list is empty display "**No bills to show"**, else display "Bills in [expense name]" and display all the bill details in the specified format. Where [expense name] specifies the name of the expense. |

After deletion, if true is returned print "**Bill successfully deleted**", else print "**Bill not found in the expense**".  
  
**Problem Overview:**  
In this requirement contains menu driven,  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
  
If the user select option-1, then get the bill details from the user(which is seperated by comma). Split that bill details and create a bill object(Use CreateBill method) and add that bill to the bill list(which is in Expense) Use AddBillToExpense method to add the bill to the expense.  
  
If the user select option-2, then get the bill number from the user, If that bill will found in the billList(Expense), then remove the bill from the expense and print the message "**Bill successfully deleted**". If the bill not found in the list, then print the message "**Bill not found in expense**". Use RemoveBillFromExpense method, that method returns boolean(either true or false) value. Display the message in Main menu based upon the appropriate return type.  
  
If the user select option-3, then display the bill details for the expense(Use following format to print the details). Use DisplayBills method. If there is no bill in the list, then display "**No bills to show**".  
  
**Note:** The above print statements should be present in the main method. Use   
**Console.WriteLine("{0,-8} {1,-8} {2,-8} {3,-8} {4}", "Sold By", "Sold To", "Bill No", "Amount", "Bill Date");**  
for printing the details of the bill.  
  
When the “bill” object is printed, it should display the following format  
Print format:  
**Display 1 digit after the decimal point in Double.**  
  
**Sample Input/Output:**  
  
Enter the name of the Expense:  
**Purchase**  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
Enter your choice:  
**3**  
No bills to show  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
Enter your choice:  
**2**  
Enter the bill number to be deleted:  
**1**  
Bill not found in expense  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
Enter your choice:  
**1**  
**Matt,Danny,187,3500,16-08-2017**  
Bill successfully added  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
Enter your choice:  
**1**  
**Brandon,Rob,387,2348,09-09-2017**  
Bill successfully added  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
Enter your choice:  
**1**  
**Leonard,Martin,101,1000,30-06-2017**  
Bill successfully added  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
Enter your choice:  
**3**  
Bills in Purchase  
Sold By  Sold To  Bill No  Amount   Bill Date  
Matt     Danny    187      3500.0   16-08-2017  
Brandon  Rob      387      2348.0   09-09-2017  
Leonard  Martin   101      1000.0   30-06-2017  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
Enter your choice:  
**2**  
Enter the bill number to be deleted:  
**356**  
Bill not found in expense  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
Enter your choice:  
2  
Enter the bill number to be deleted:  
**187**  
Bill successfully deleted  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
Enter your choice:  
**3**  
Bills in Purchase  
Sold By  Sold To  Bill No  Amount   Bill Date  
Brandon  Rob      387      2348.0   09-09-2017  
Leonard  Martin   101      1000.0   30-06-2017  
1.Add Bill  
2.Delete Bill  
3.Display Bills  
4.Exit  
Enter your choice:  
**4**  
**Expense - Requirement 3**

**Requirement 3:**  
In this requirement develop a feature in which you can search a List of Bills by amount and billDate.  
  
a) Create a Class **Bill** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_soldBy | string |
| \_soldTo | string |
| \_billNo | int |
| \_amount | double |
| \_billDate | DateTime |

Mark all the attributes as private.  
Create / Generate appropriate properties.  
Add a default constructor and a parameterized constructor to take in all attributes in the given order:  
**Bill( string \_soldBy, string \_soldTo, int \_billNo, double \_amount, DateTime billDate )**  
  
b) Create a class **BillBO**with the following methods,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public List<Bill> FindBill(List<Bill> billList,double amount) | This method accepts a list of bills and amount as arguments and returns a list of bills having amount less than or equal to the given amount. |
| public List<Bill> FindBill(List<Bill> billList,DateTime billDate) | This method accepts a list of bills and billDate as arguments and returns a list of bills that match with the given billDate. |

The mail details should be given as a comma-separated value in the below order,  
**\_soldBy, \_soldTo, \_billNo, \_amount, \_billDate**  
  
**Problem Overview:**  
First of all get the total number of bills from the user n.  
The next n line of input contains string, that corresponds to the bill details(which is comma seperated). Split that string and create a bill object and add that bill into the billList(which is maintained in Main method).  
The main theme for this requirement is filter the bills by users choice.  
This requirement contains menu driven,

1.By Bill Date

2.By Amount

If the user select option-1, then get the billDate from the user and filter the bill list by the billDate. Use FindBill(List,DateTime) method.

If the user select option-2, then get the amount from the user and filter the bill list by the amount. Use FindBill(List,double) method.

Finally display the bill details using following format.  
  
Print format:  
**Console.WriteLine("{0,-8} {1,-8} {2,-8} {3,-8} {4}","Sold By","Sold To","Bill No","Amount","Bill Date");**  
  
**Note:**The bill lists are displayed in the Main method.  
              If any other choice is selected, display "**Invalid choice**"  
              If search detail is not found, display "**No such bill is present**"

**Display one digit after the decimal point for Double Datatype.**  
  
  
**Sample Input and Ouput 1:**

Enter the number of bills:

**4**

**Harry,Joe,254,2500,08-02-2017**

**Oliver,Tina,77,8540,15-12-2017**

**Leonard,Martin,101,1000,30-06-2017**

**Lucas,Will,208,1200,25-04-2017**

Enter a search type:

1.By Bill Date

2.By Amount

**1**

Enter the Bill Date:

**15-12-2017**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sold By | Sold To | Bill No | Amount | Bill Date |
| Oliver | Tina | 77 | 8540.0 | 15-12-2017 |

**Sample Input and Ouput 2:**

Enter the number of bills:

**4**

**Harry,Joe,254,2500,08-02-2017**

**Oliver,Tina,77,8540,15-12-2017**

**Leonard,Martin,101,1000,30-06-2017**

**Lucas,Will,208,1200,25-04-2017**

Enter a search type:

1.By Bill Date

2.By Amount

**2**

Enter the Amount:

**1200**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sold By | Sold To | Bill No | Amount | Bill Date |
| Leonard | Martin | 101 | 1000.0 | 30-06-2017 |
| Lucas | Will | 208 | 1200.0 | 25-04-2017 |

**Sample Input and Ouput 3:**

Enter the number of bills:

**4**

**Harry,Joe,254,2500,08-02-2017**

**Oliver,Tina,77,8540,15-12-2017**

**Leonard,Martin,101,1000,30-06-2017**

**Lucas,Will,208,1200,25-04-2017**

Enter a search type:

1.By Bill Date

2.By Amount

**10**

Invalid choice

**Expense - Requirement 4**

**Requirement 4:**  
  
In this requirement, you need to sort the list of bills based on \_billNo, \_amount, and \_billDate.  
  
a) Create a class **Bill** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_soldBy | string |
| \_soldTo | string |
| \_billNo | int |
| \_amount | double |
| \_billDate | DateTime |

Mark all the attributes as private.  
Create / Generate appropriate properties.  
Add a default constructor and a parameterized constructor to take in all attributes in the given order:  
**Bill( string \_soldBy, string \_soldTo, int \_billNo, double \_amount, DateTime \_billDate )**  
  
b) Create the following static methods in the Bill class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static Bill CreateBill(string detail) | This method accepts a String. The bill detail separated by commas is passed as the argument. Split the details and create a bill object and returns it. |

The bill details should be given as a comma-separated value in the below order,  
\_soldBy, \_soldTo, \_billNo, \_amount, \_billDate  
  
c) The Bill class should implement the **IComparable** interface which sorts the Bill list based on bill number(\_billNo).While comparing, all the bill number() attributes in the list are unique.  
  
d) Create a class **AmountComparer** which implements **Comparer** interface and sort the Bill list based on amount. While comparing, all the amount attributes in the list are unique.  
  
e) Create a class **BillDateComparer** which implements **Comparer** interface and sort the Bill list based on date of bill(\_billDate). While comparing, all the date of bill(\_billDate). attributes in the list are unique.  
  
Get the number of Bills and bill details and create a bill list. Sort the Bills according to the given option and display the list.  
  
**Problem Overview:**  
First of all get the number of bills (n) from the user.  
The next n line contains string, that corresponds to the bill details(which is comma seperated). Split that string and create a bill object and add to the bill list(which is maintain in **Main**method).  
The main theme of this requirement is Sorting the bill list by user choice.  
This requirement contains menu driven,  
1.Sort by Bill No  
2.Sort by Bill Date  
3.Sort by Amount  
  
If the user select option-1, then sort the bill list based on the bill number(\_billNo) of the bill(by ascending order).  
If the user select option-2, then sort the bill list based on the bill date(\_billDate) of the bill(by ascending order).  
If the user select option-3, then sort the bill list based on the amount(\_amount) of the bill(by ascending order).  
  
After sorted the bill list display the bill list by using the following format.  
  
When the “bill” object is printed, it should display the following details. Override **ToString** method  
Print format:  
**Console.WriteLine("{0,-8} {1,-8} {2,-8} {3,-8} {4}","Sold By","Sold To","Bill No","Amount","Bill Date");**  
  
**Display one digit after the decimal point for Double datatype.**  
  
**Sample input & output 1**  
Enter the number of bills:  
**3  
James,Winn,807,4560,14-10-2017  
Matt,Danny,187,3500,16-08-2017  
Walter,Hank,471,7800,27-01-2017**  
Enter a type to sort:  
1.Sort by Bill No  
2.Sort by Bill Date  
3.Sort by Amount  
1  
Sold By  Sold To  Bill No  Amount   Bill Date  
Matt        Danny   187       3500.0       16-08-2017  
Walter   Hank      471       7800.0       27-01-2017  
James   Winn       807       4560.0       14-10-2017  
  
**Sample input & output 2**  
Enter the number of bills:  
**3  
James,Winn,807,4560,14-10-2017  
Matt,Danny,187,3500,16-08-2017  
Walter,Hank,471,7800,27-01-2017**  
Enter a type to sort:  
1.Sort by Bill No  
2.Sort by Bill Date  
3.Sort by Amount  
2  
Sold By   Sold To  Bill No  Amount   Bill Date  
Walter    Hank       471      7800.0       27-01-2017  
Matt         Danny   187      3500.0       16-08-2017  
James    Winn       807      4560.0       14-10-2017  
  
**Sample input & output 3**  
Enter the number of bills:  
**3  
James,Winn,807,4560,14-10-2017  
Matt,Danny,187,3500,16-08-2017  
Walter,Hank,471,7800,27-01-2017**  
Enter a type to sort:  
1.Sort by Bill No  
2.Sort by Bill Date  
3.Sort by Amount  
3  
Sold By  Sold To  Bill No  Amount   Bill Date  
Matt        Danny   187       3500.0       16-08-2017  
James   Winn        807      4560.0       14-10-2017  
Walter   Hank       471      7800.0        27-01-2017

**Expense-Requirement 5**

Now we are gonna find the person who has done the maximum sales from a list of bills based on the total amount incurred from the sale.  
  
**Requirement 5:**  
  
a)Create a Class **Bill** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_soldBy | string |
| \_soldTo | string |
| \_billNo | int |
| \_amount | double |
| \_billDate | DateTime |

Mark all the attributes as private.  
Create / Generate appropriate properties.  
Add a default constructor and a parameterized constructor to take in all attributes in the given order:   
**Bill( string \_soldBy, string \_soldTo, int \_billNo, double \_amount, DateTime \_billDate )**  
  
b) Create the following static method in **Bill** class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public static Bill CreateBill(string detail) | This method accepts a string which contains bill details separated by commas. Split the details and create a bill object from the details and return it. |
| public static string GetMaximumSoldBy(List<Bill> billList) | This method returns the name of the soldBy person who has the maximum amount of sales in the given bills. |

The bill details should be given as a comma-separated value in the below order,  
**\_soldBy, \_soldTo, \_billNo, \_amount, \_billDate**  
  
**Problem Overview:**  
First of all get the number of bills from the user n.  
The next n line of input contains the bill details(which is comma seperated). Split that string and create a bill object and add to the bill list(which is maintain in Main method).  
The main theme of the requirement is to predict the user, who is with maximum amount (calculate by \_soldBy)  
  
**Sample Input/Output:**

Enter the number of bills:

**5**

**Brandon,Rob,387,2348,09-09-2017**

**James,Winn,807,4560,14-10-2017**

**Brandon,Danny,187,7500,16-08-2017**

**Walter,Hank,471,7800,27-01-2017**

**James,Sam,398,4000,23-11-2017**

The highest sales is done by Brandon

**Expense - Requirement 6**

**Requirement 6:**  
  
In this requirement, given a list of bills, you need to find the amount billed on a Date using SortedDictionary.  
  
a) Create a Class **Bill** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_soldBy | string |
| \_soldTo | string |
| \_billNo | int |
| \_amount | double |
| \_billDate | DateTime |

Mark all the attributes as private.  
Create / Generate appropriate properties.  
Add a default constructor and a parameterized constructor to take in all attributes in the given order:  
**Bill(string \_soldBy, string \_soldTo, int \_billNo, double \_amount,DateTime \_billDate)**

b) Create the following static methods in the Bill class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static SortedDictionary<DateTime,double> AmountPerDate(List<Bill> list) | This method accepts a list of Bill as arguments and returns a**SortedDictionary** with the **\_billDate** as key and the amount billed on the date as value and returns the SortedDictionary. |

In the SortedDictionary have the billDate as key and calculate the amount billed on the date and keep the amount as value. Print the value sorted by billDate.  
  
The Bill details will be given as a comma-separated value in the below order,

**soldBy,** **soldTo** **,billNo,amount,** **billDate**

**Problem Overview:**  
First of all get the number of bills from the user n.  
The next n line of input contains the bill details(which is comma seperated). Split that string and create a bill object and add to the bill list(which is maintain in Main method).  
The main theme of the requirement is calculate the number of occurance of the bill based on the bill date of the bill.  
Please use SortedDictionary to do this requirement.  
Finally display the dictionary details by the following format.  
  
Print format:  
**Console.WriteLine("{0,-15} {1}","Date","Amount");**  
  
**Sample Input and Output 1:**

Enter the number of bills:

**6**

**Lucas,Will,208,1200,25-04-2017**

**Brandon,Rob,387,2348,16-08-2017**

**James,Winn,807,4560,14-10-2017**

**Matt,Danny,187,3500,25-04-2017**

**Walter,Hank,471,7800,14-10-2017**

**Dean,Sam,398,4000,14-10-2017**

|  |  |
| --- | --- |
| Date | Amount |
| 25-04-2017 | 4700.0 |
| 16-08-2017 | 2348.0 |
| 14-10-2017 | 16360.0 |