**Identify classes with appropriate data fields, instantiated objects and file structures required to implement the chosen system. (P2.1)**

**1. Instance field members for the Customer class**

|  |  |  |
| --- | --- | --- |
| **Data Description** | **Type** | **Attributes** |
| Customer Name | String | name |
| IC No of the customer | String | ic |
| Customer’s address | string | address |
| Customer’s phone number | String | phone |
| ID Card Number | String | idcardNo |
| Initial point of the program | Int | point |
| Class Transaction | Transaction | TransactionforSale1 |

**Object of Customer class:** objCustomer, theCust

**File Structure:** text file name CustomerInformation.txt and CustomerTransaction.txt.

**2. Instance field members for the Transaction class**

|  |  |  |
| --- | --- | --- |
| **Data Description** | **Type** | **Attributes** |
| Total Price of items | int | totalitem |
| Total litre of fuel | int | totallitrefuel |
| Total price of transaction | int | TotalPriceTransaction |

**Object of Transaction class:**  TransactionforSale1, objTransaction

**File Structure:** none.

**Produce a complete and logical class diagram with appropriate behaviors following UML notations. (P2.2)**



**Produce object diagram and define the relationship between objects for your system that meet with the design requirements. (P3.2)**



**Produce pseudocode for all the methods in the identified classes and application program. (P2.2)**

**1.0 Class Customer**

1.1 Method customerReg (Transaction aTransaction)

Begin

Assign object TransactionforSale1 for class Transaction

End

1.2 Method getCalculatetotalPriceforFuel()

Begin

Call method CalculatetotalPriceforFuel() from class Transaction

End

1.3 Method getcalculatetotalPrice()

Begin

Call method calculatetotalPrice() from class Transaction

End

1.4 Method getCalculateconvertToPoint()

Begin

Call method CalculateconvertToPoint() from class Transaction

End

1.5 Method calculateCurrentPoint()

Begin

Add Point with method CalculateconvertToPoint() from class Transaction

End

**2.0 Class Transaction**

2.1 Method CalculatetotalPriceforFuel()

Begin

Multiply total liter of fuel of 3

End

2.2 Method calculatetotalPrice()

Begin

Add total price of item with total price of fuel

End

2.3 Method CalculateconvertToPoint()

Begin

Add total point of fuel with total point of item.

End

**3.0 Main Menu Form**

3.1 btnRegister

Begin

Open Register form

End

3.2 btnBuyItem

Begin

Open Buy Item form

End

3.3 btnRedeemPoint

Begin

Open Redeem Point form

End

3.4 btnReport

Begin

Open Report Menu form

End

3.5 btnExit

Begin

Close application

End

**4.0 Register form**

4.1 btnRegisterCustomer

Begin

Instantiate object named objCustomer of class Customer

Read Name

Read IC no

Read Address

Read Phone

Read ID Card No

Read Point

If Name or IC No or Address or Phone or ID Card No or Point == “-”

Display message box “Please complete the form”

Else

All customer details will be saved in CustomerInformation.txt text file

Message Box displayed “You has been registered successfully “

End if

End

4.2 btnReset

Begin

The form will be cleared

End

4.3 btnExit

Begin

Application will be closed

End

4.4 btnMainMenu

Begin

Display Main Menu form

End

**5.0 Buy Item Form**

5.1 btnSearch

Begin

Read ID Card No

Search for customer Information in CustomerInformation.txt text file by using ID Card No

All Customer’s details display in text boxes

End

5.2 btnCalculate

Begin

Instantiate object named objCustomer of class Customer

Instantiate object named objTransaction of class Transaction

Read total liter of fuel

Read total price of item

Call method getCalculatetotalPriceforFuel()

Call method getcalculatetotalPrice()

Display total price in textbox

End

5.3 btnCalBalance

Begin

Instantiate object named objCustomer of class Customer

Instantiate object named objTransaction of class Transaction

Read payment

Read total price

If payment >= total price

Calculate balance = payment – total price

Display balance in text box

Else

Display message box “Payment should be more than total price”

End if

End

5.4 btnCollectPoint

Begin

If total price == “-” or payment == “-”

Display message box “Please complete the transaction”

Else

Instantiate object named objCustomer of class Customer

Instantiate object named objTransaction of class Transaction

Read total liter of fuel

Read total price of item

Call method getCalculateconvertToPoint();

Display Collected point in text box.

End if

End

5.5 btnUpdate

Begin

Instantiate object named objCustomer of class Customer

Instantiate object named objTransaction of class Transaction

Call method calculateCurrentPoint()

Display Current Total Point in text box

Read Customer Details

Read Current Total Point

The Customer Detail and Current Collected Point is saved in CustomerInformation.txt

End

5.6 btnViewReceipt

Begin

If ID Card No == “- “

Display message box “Please complete the transaction “

Else

Instantiate object named objCustomer of class Customer

Open Transaction Receipt form

Read Customer Details

Read total liter of fuel

Read total price for items

Read Payment

Read Total Price for Transaction

Read Balance

All the Customer details and Transaction detail will be save in CustomerTransaction.txt text file

End if

End

5.7 btnMainMenu

Begin

Display Main Menu form

End

5.8 btnExit

Begin

Application will be closed

End

**6.0 Transaction Receipt Form**

6.1 btnView

Begin

Display Transaction Detail from Buy Item form

End

6.2 btnClear

Begin

The form will be cleared

End

6.3 btnMainMenu

Begin

Display Main Menu form

End

6.4 btnExit

Begin

Application will be closed

End

**7.0 Redeem Point form**

7.1 btnSearch

Begin

Read ID Card No

Search for customer Information in CustomerInformation.txt text file by using ID Card No

All Customer’s details display in text boxes

End

7.2 btnRedemPoint

Begin

If ID Card No == “-”

Display message box “Please enter ID Card No”

Group box for item is disabled

Else

Read Current Total Point

End if

If Current Total Point < 100

Display message box “You do not have enough point for the redemption. Please try again later!”

Group box for items is disabled

Else

Display message box “You are allowed for the redemption”

Group box for items is enabled

End If

End If

End

7.3 btnEnter

Begin

Read point for the redemption

If point for the redemption < 100

Display message box “The point for the redemption should be more than 100”

Else

If point for the redemption >= 100 and pointToRedeem <= 200

Combo box for item 1 is enabled

Combo box for item 2 is disabled

Combo box for item 3 is disabled

Combo box for item 4 is disabled

Else If point for the redemption >= 201 and pointToRedeem <= 300

Combo box for item 1 is disabled

Combo box for item 2 is enabled

Combo box for item 3 is disabled

Combo box for item 4 is disabled

Else If point for the redemption >= 301 and pointToRedeem <= 400

Combo box for item 1 is disabled

Combo box for item 2 is disabled

Combo box for item 3 is enabled

Combo box for item 4 is disabled

Else If point for the redemption >= 401

Combo box for item 1 is disabled

Combo box for item 2 is disabled

Combo box for item 3 is disabled

Combo box for item 4 is enabled

End if

End if

End

7.4 btnRedeemnow

Begin

Instantiate object named objCustomer of class Customer

Instantiate object named objTransaction of class Transaction

Read current total point

Read point for redemption

Calculate remaining point = current total point – point for redemption

Display remaining point in text box

Read Customer details

Read remaining point

Update customer details and remaining point in CustomerInformation.txt text file

End

7.5 btnViewRedemption

Begin

Instantiate object named objCustomer of class Customer

All customer’s redemption information will be displayed in list box

All customer information and customer’s redemption information will be saved in RedemptionReport.txt text file

End

7.6 btnClear

Begin

The Redeem Point will be Cleared

End

7.7 btnMainMenu

Begin

Display Main Menu form

End

7.8 btnExit

Begin

Application will be closed

End

**8.0 Customer Report form**

9.1 btnView

Begin

Instantiate object named objCustomer of class Customer

Read Customer’s details from CustomerInformation.txt text file

Display Customer’s detail in list box

End

**9.0 Transaction Report form**

10.1 btnView

Begin

Read Transaction report from CustomerTransaction.txt text file

Display Transaction report in list box.

End

**10.0 Redemption Report form**

11.1 btnView

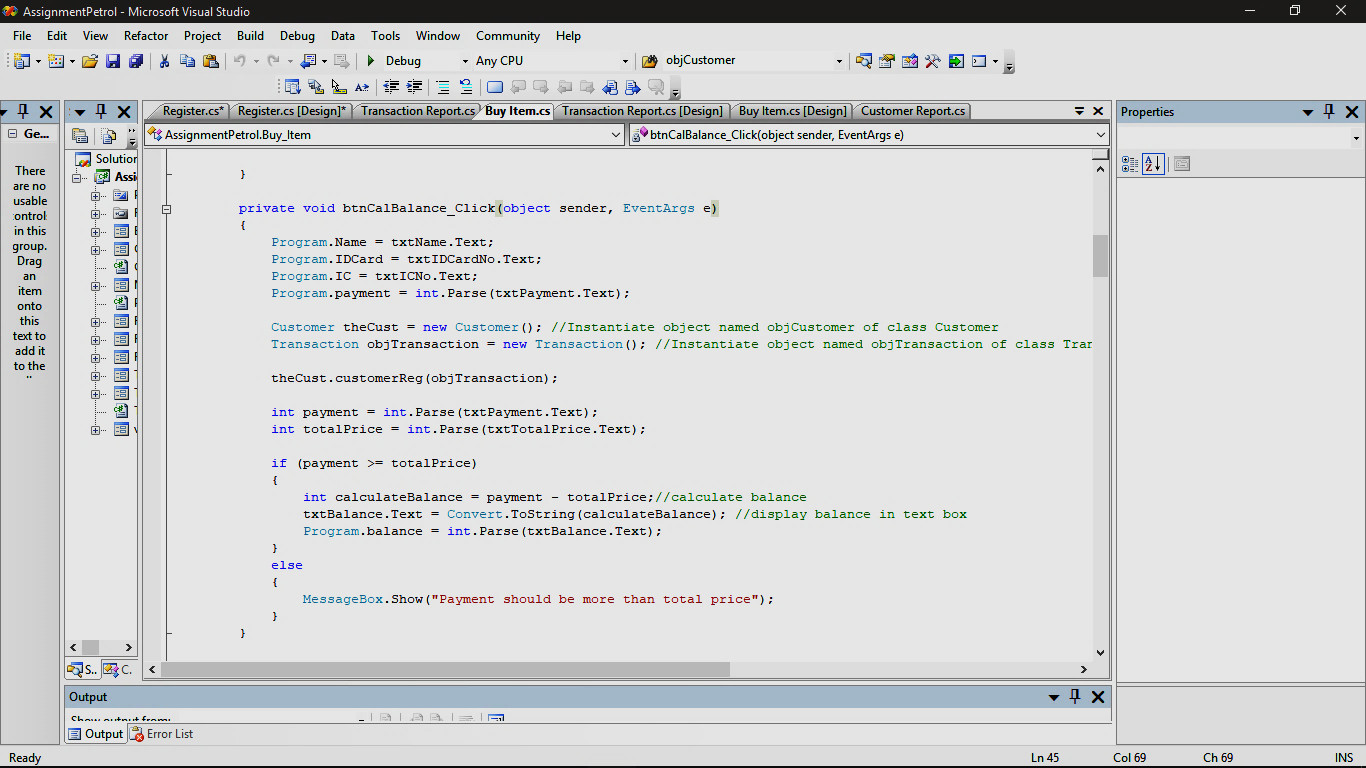
Begin

Read Redemption report from CustomerTransaction.txt text file

Display Redemption report in list box.

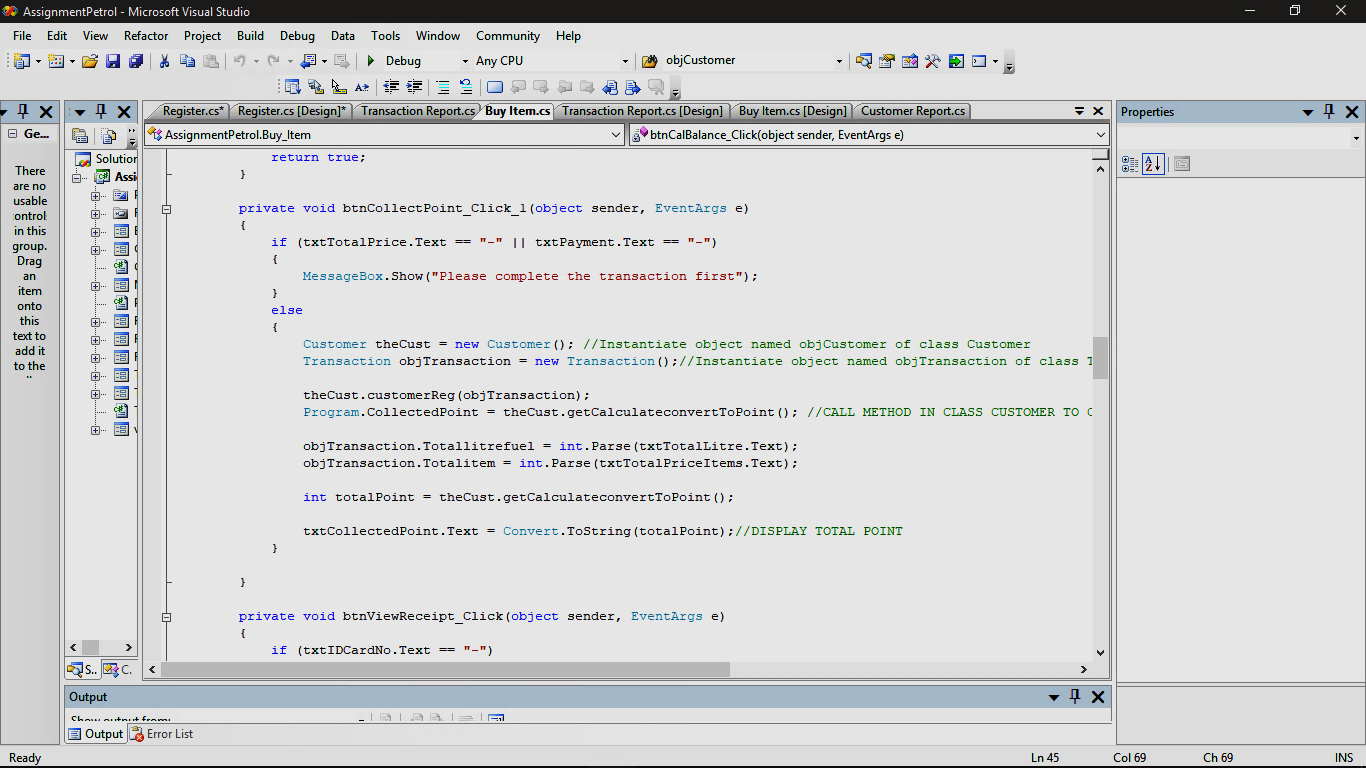
End’

**Explain the implemented control structures (selection and repetition control structures) in your system and support with appropriate print screen(s). (P3.3)**

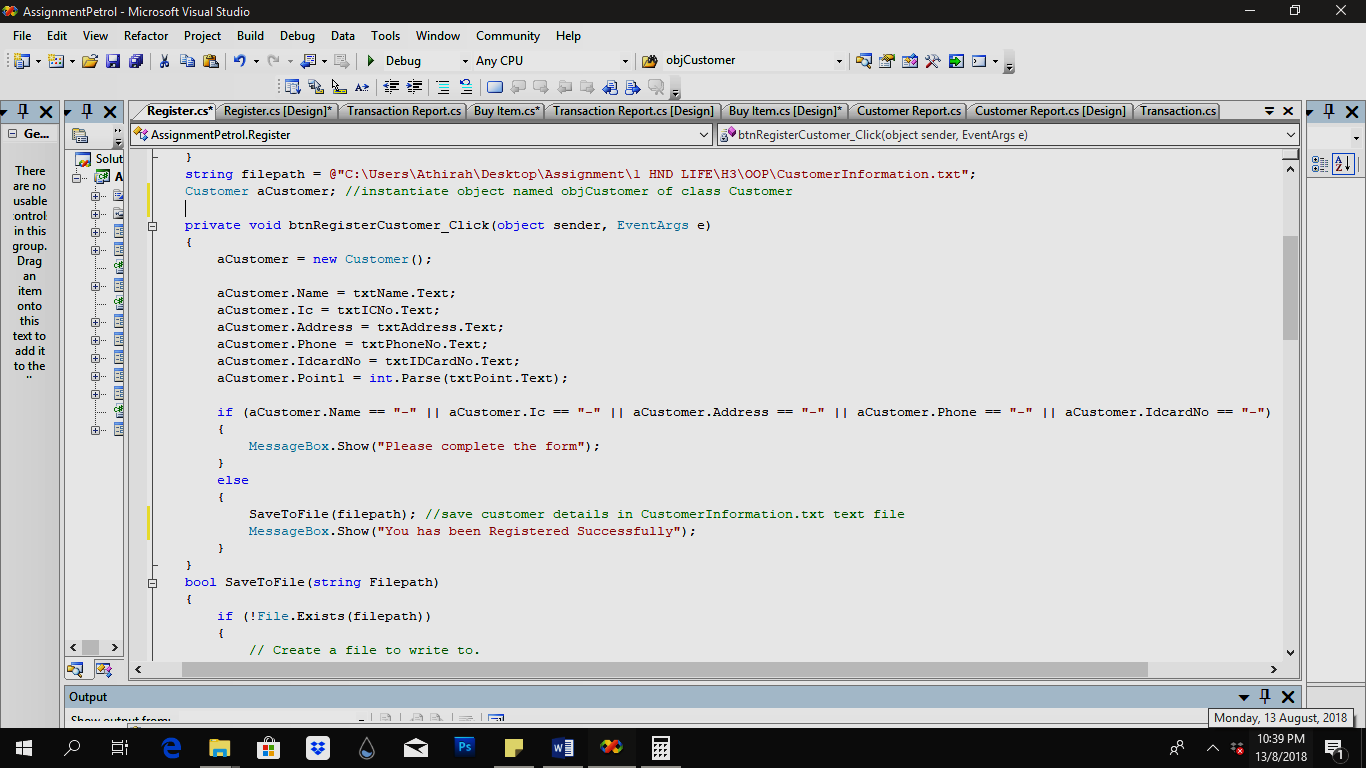


**Figure 1**

The figure above shows one of the selection structure in button calculate balance. If the user enters payment more than total price of the transaction, the system will calculate balance of the payment. Then, it will display the answer in the text box. But, if the user enters the payment less then total price, the system will display message box “Payment should be more than total price”.

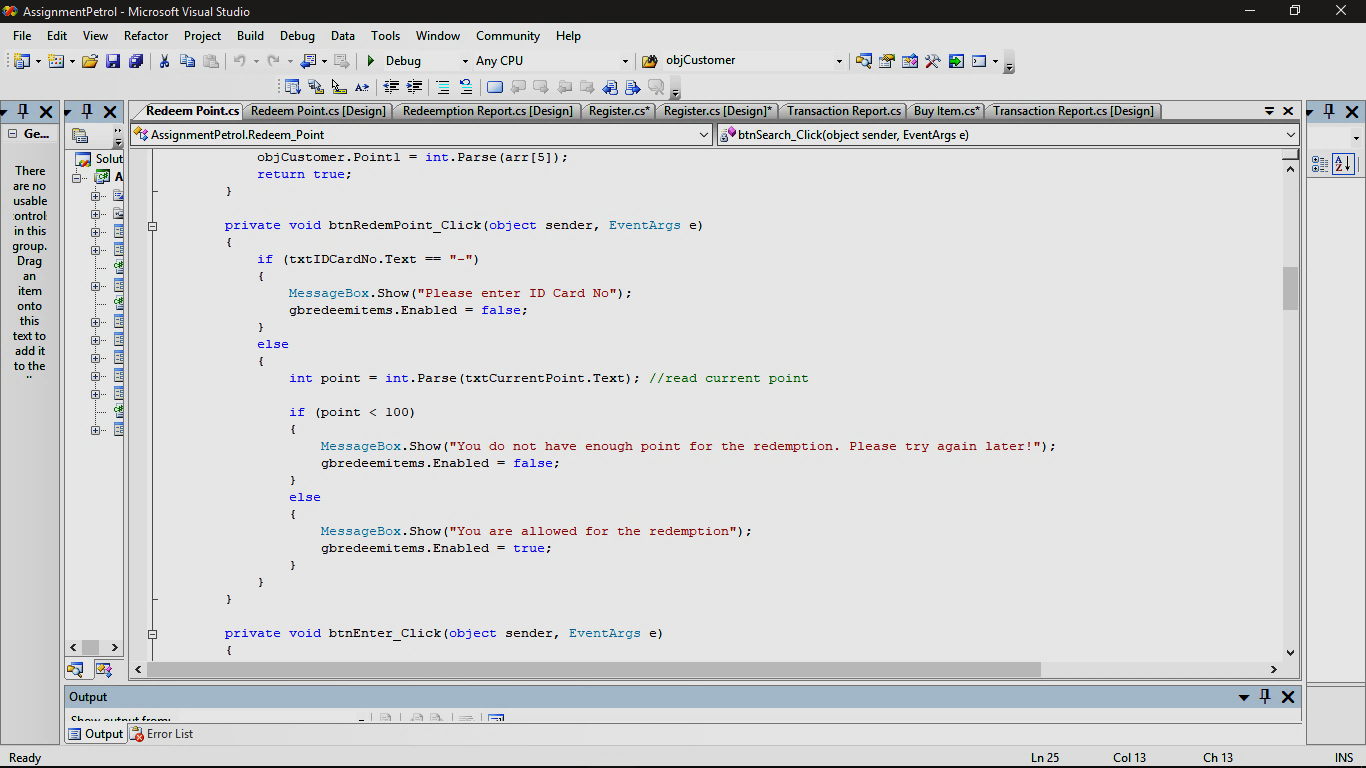


**Figure 2**

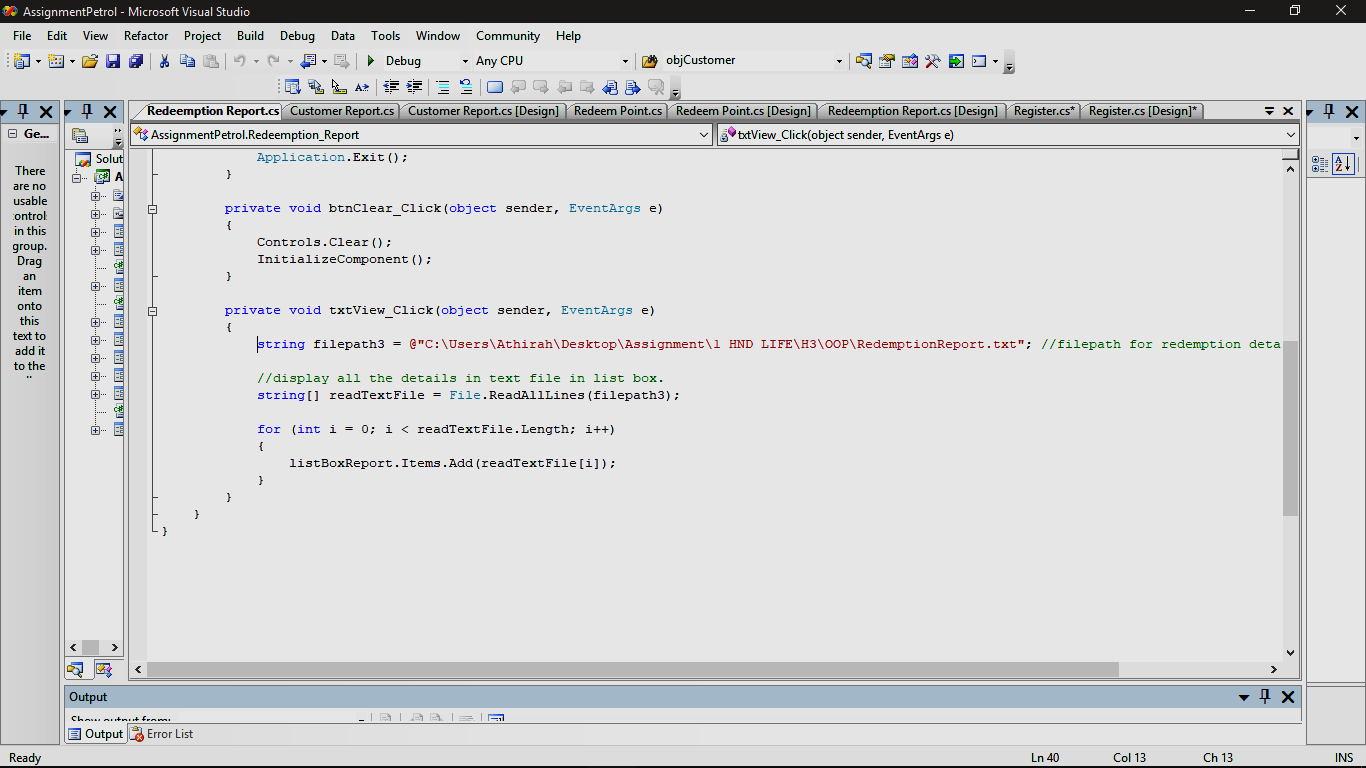
The figure above shows the other selection control structure in the system. The selection structure is in button CollectPoint. If the users do not enter the calculate button for the transaction and the text box for the total price of the transaction is null and if the user do not enter the payment, the system will display message box “Please complete the transaction first”. Else, the system will calculate the total point from the total liter of fuel and total price of items. Then, it will display the collected point from the transaction in the text box.

**Figure 3**

The figure above shows another selection control structure which is in register customer button. If the users do not fill the customer name, IC no, address, phone number and ID Card no in the text box, the system will display message box “Please complete the form”. But, if the users fill all the customer’s details, the message box “You has been Registered Successfully” and it will save all the customer’s details in CustomerTransaction.txt text file.

 **Figure 4**

The figure above shows another selection control structure which is in redeem point button. if the users do not enter the ID Card No, the system will display message box “Please enter ID Card No”. But, if the users enter the ID Card No, the system will display the current point and the point. If the current point is less than 100, the message box displayed “You do not have enough point for the redemption. Please try again later!”. But if the current point is more than and equal to 100, the system will display message box “You are allowed for the redemption”.

 **Figure 5**

The figure above shows one of the repetition control structures. It uses for loop control structure. The for loop is in view redemption report button. The for loop is set up to read the length in the text files that is lesser than total length of the text file. If the text file is less than total length, it will keep looping and display all the redemption report in the list box.

**Describe the purpose of each component in the Integrated Development Environment (IDE) that you use for the system development which includes editor space, toolbox, solution explorer, compiler and debugger. (P3.4)**

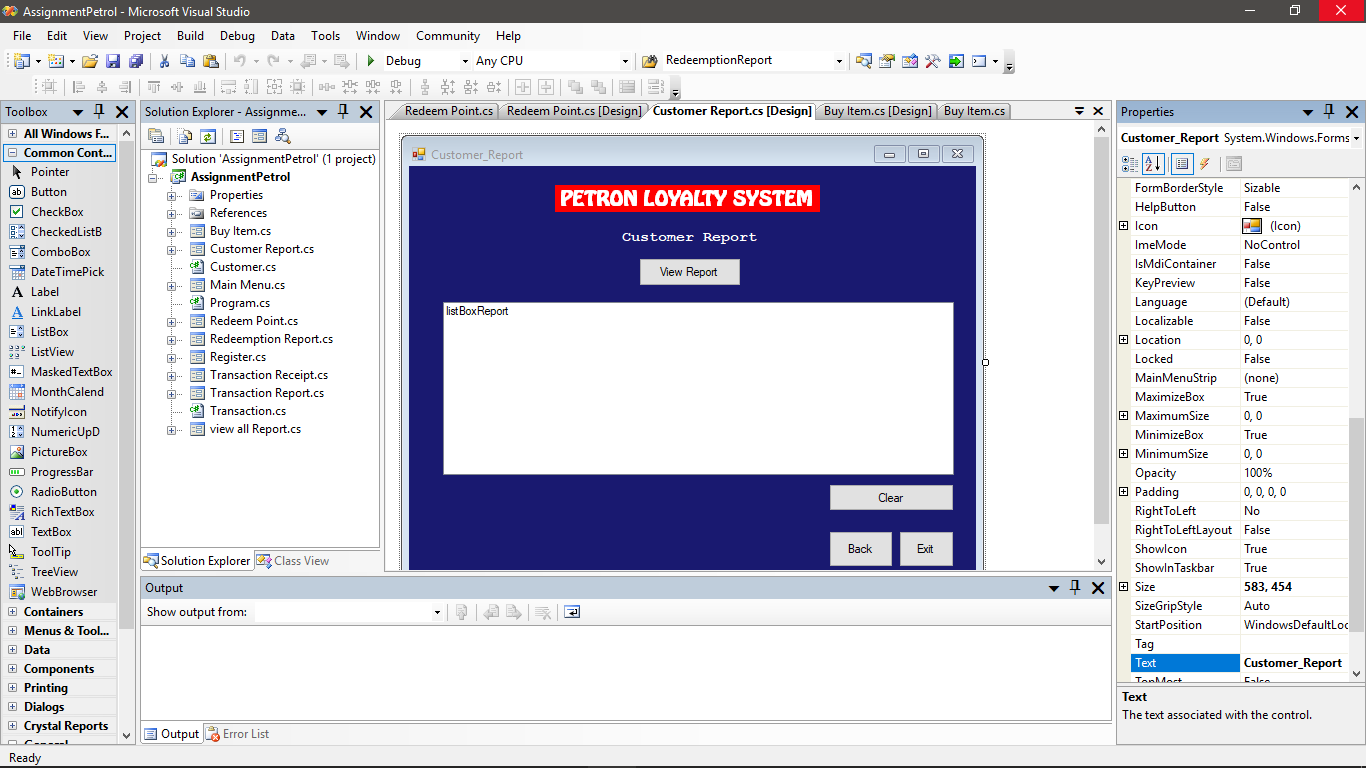
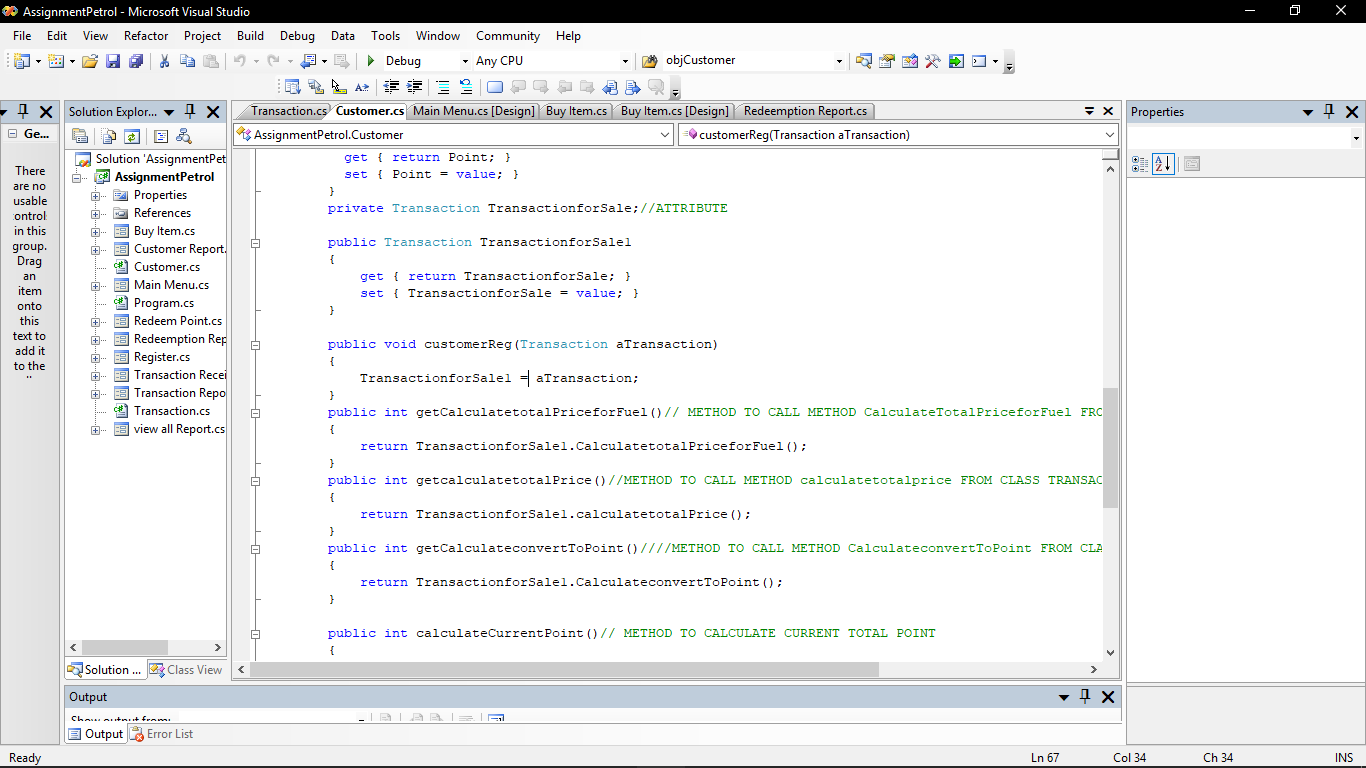


Figure 6

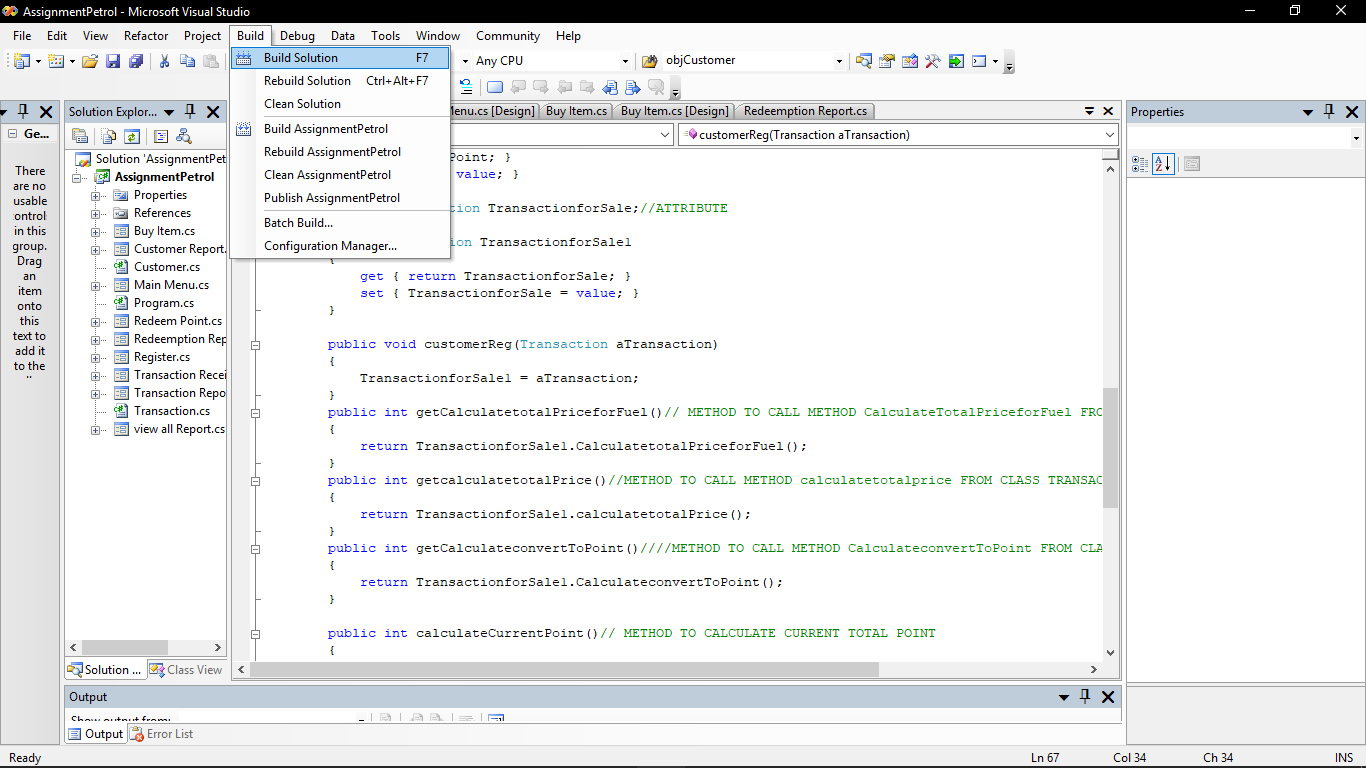
This is according to Docs.microsoft.com. (2017), Integrated Development Environment (IDE) is a creative launching pad that is used to build code, edit, debug and publish software. It is feature-rich program that can be used for many aspects of software development. IDE in Visual Studio provides many features includes compiler, debugger, editor space, toolbox and solution explorer as the Figure 6 shows above.

From the Figure 1, the first feature in IDE is toolbox window. The purpose of toolbox window is to provides a list of controls and component that the developer can use to design the interface of a software using Windows Form. The toolbox window has several tabs that provide different set of components. The example of the components is button, textbox, list box, combo box, label, check box, date time picker, radio button, picture box, month calendar, group box and many more. All these components use in designing the interface of the software to make the software more efficient and user-friendly.

The other feature that IDE provide is solution explorer window. Solution explorer window is a window that show an expandable list of projects, each project’s references and each project’s components. The purpose of solution explorer window is to let the developer to view, navigate and manage the code files and window form. It is because solution explorer provide button which include view code button, view designer button and properties button. It also helps developer in order to organize the code by separating all the files into solutions and projects.

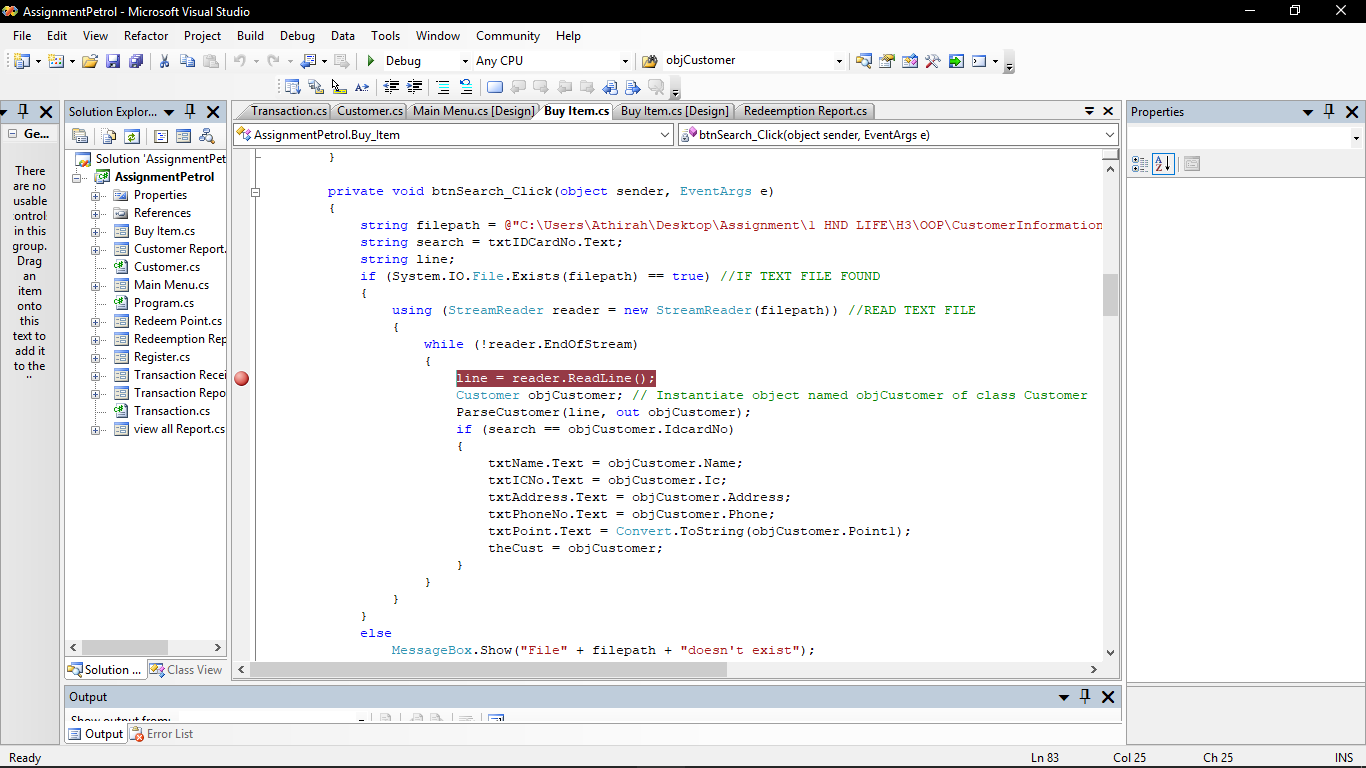
 **Figure 7**

From the figure 7, the other component in the Integrated Development Environment (IDE) is editor space. Editor space is a powerful text editor. The editor is used to write and manage code and text. It also provides many features to make the developer easy to write code. From the editor space, the developer can find and replace text or code by using Find and Replace if the code need to be replaced. Other than that, the editor space also provides IntelliSense where when the developer enters the name or code, the editor will display a drop-down list of the members of the form or class of object. For example, if the developer enter object such as objCustomer and enter dot (.), a list of method or attributes of the object from class will be listed. So, it will ease the developer to choose which is the suitable code for the object. It also will provide a list of all parameters that developers want to use for the development. (Docs.microsoft.com. (2017)).



**Figure 8**

From the figure 8, the other component in IDE in Visual Studio is compiler. Compiler is feature where it processes written statement in editor space and translate it into machine language that computer understand. Build solution is one of the compiler in IDE. The function of the build solution is it will transform the code into the computer language and produce error list if there is any error in the code. So, from that the developer will know where is the error if the system is not running in right way or cannot be ran. Then, from the error list the developer can fix the code to make the system running well.



**Figure 9**

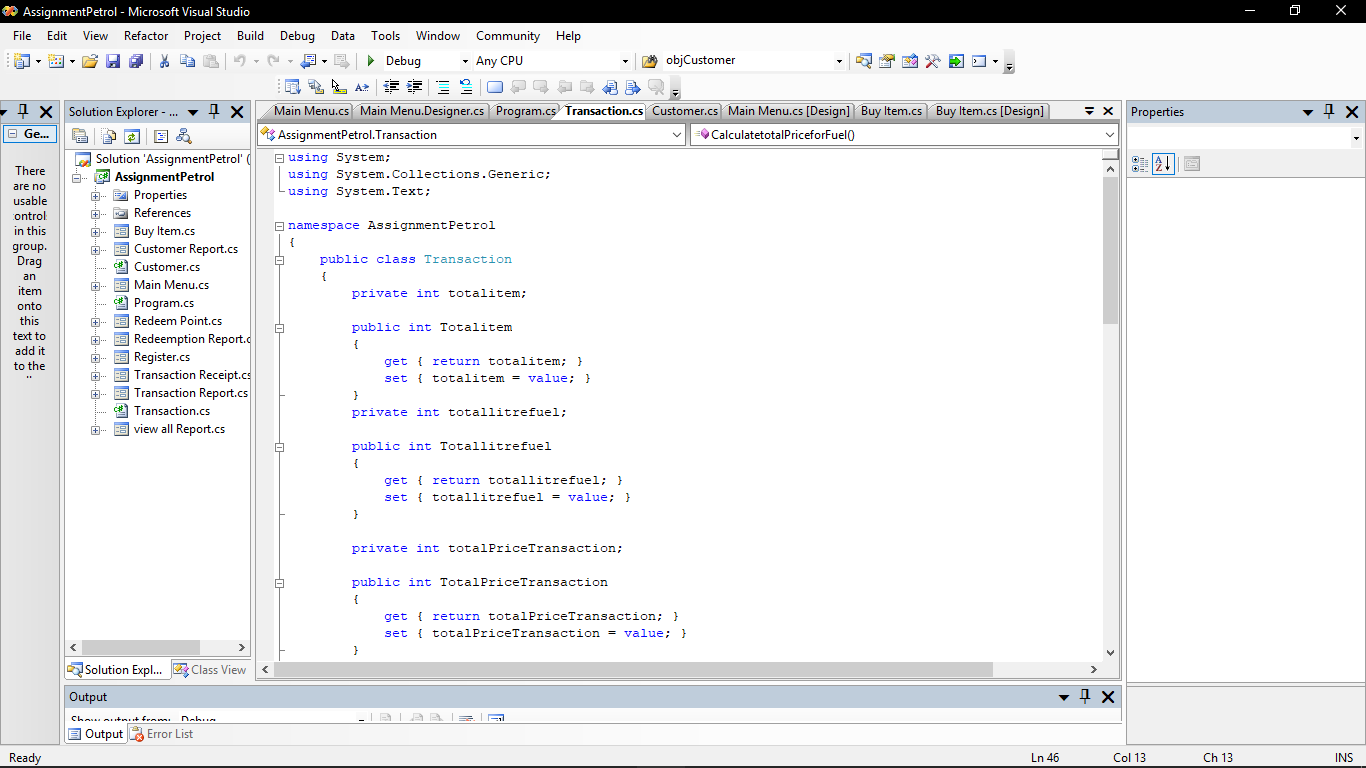
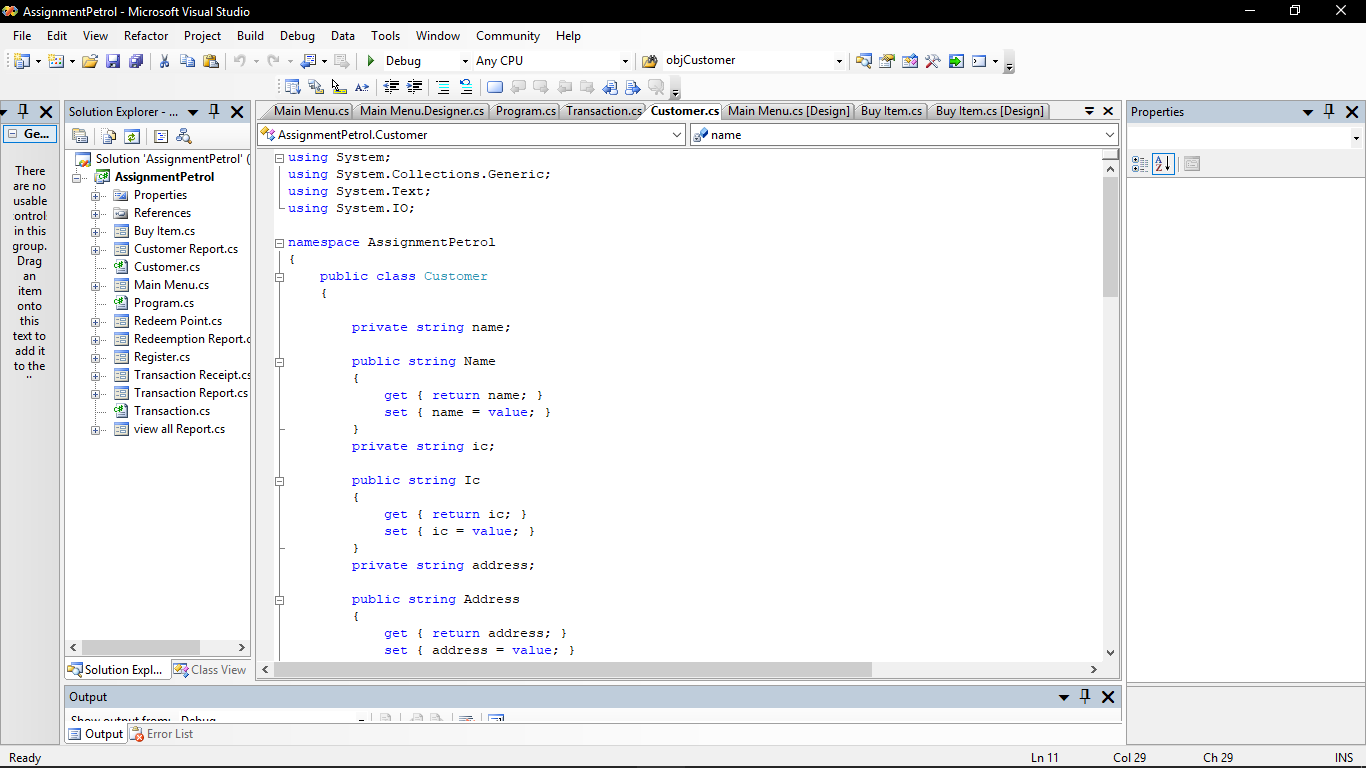
From the figure 9, the last component in IDE is debugger. The purpose of the debugger is it allows the observation of the run-time behavior of the program and locate logic errors. It also allows the developer to break the execution of the code to go through the code one by one to see the state of code variables. Breakpoint is one of the debugging techniques. To set a Breakpoint in source code, the developer should put the cursor on the line of code and press F9. A red dot in left margin will appear and the line of the code will be colored as the Figure 9. When the code is run in debugger, the execution will stop at the breakpoint before the code on that line is executed. Then, the developer need to press F11 to go through line by line of the code which it will be in yellow color. Then, the developer can see what is the value that the line hold. From that, the developer can detect error and easy for them to fix it. (Msdn.microsoft.com. (n.d.))

**Explain the different OOP concepts applied in your developed system clearly with appropriate examples and diagram. (M1.1)**

The developed system is Customer Loyalty Program System. The system is using object-oriented programming (OOP) concept. According to NeONBRAND. (n.d.), OOP is a programming language that uses classes and object to create models based on the real-world environment. There are few OOP approach concept that has been applied in the Customer Loyalty Program System which is it contain class, object, attributes, method, access modifier and constructor. In OOP approach, it uses a few relationships to relate the classes which composition, association, aggregation and inheritance. For Customer Loyalty Program System, it uses association relationship.

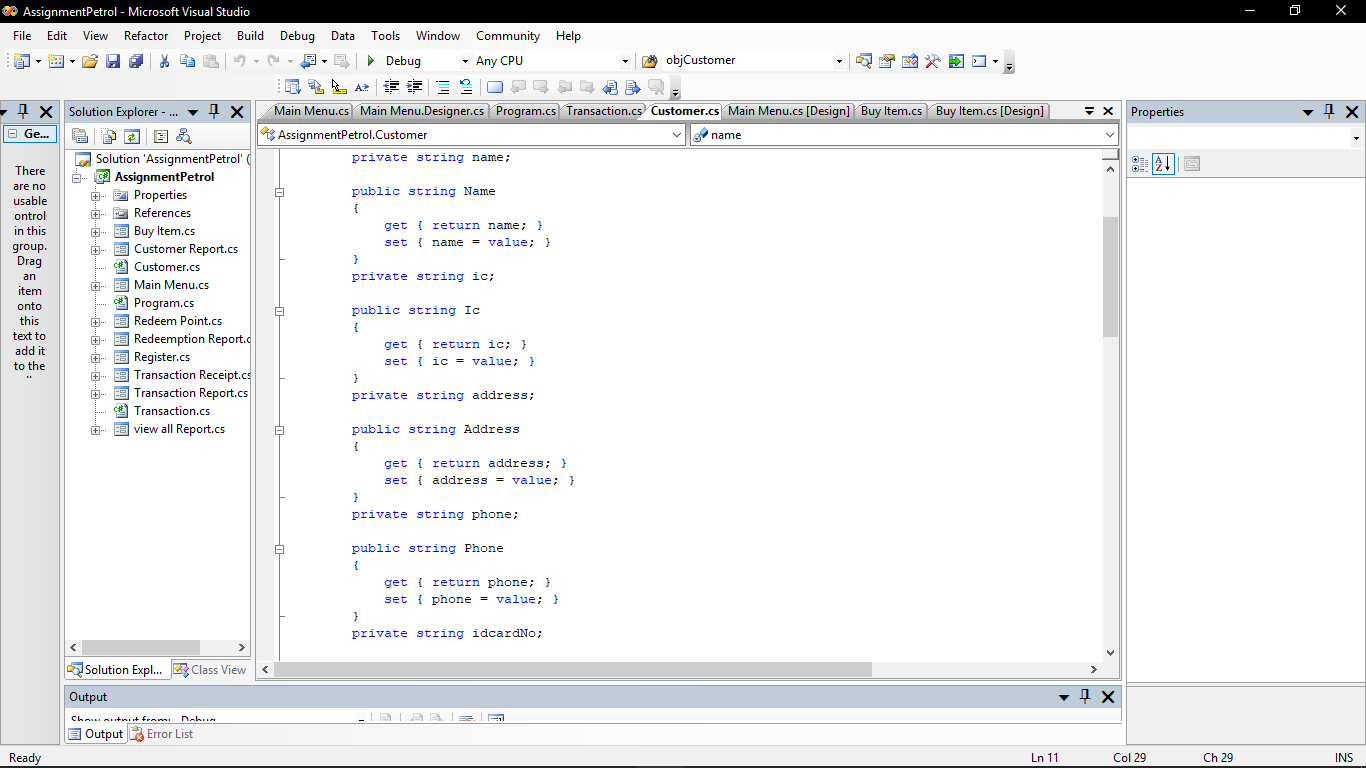
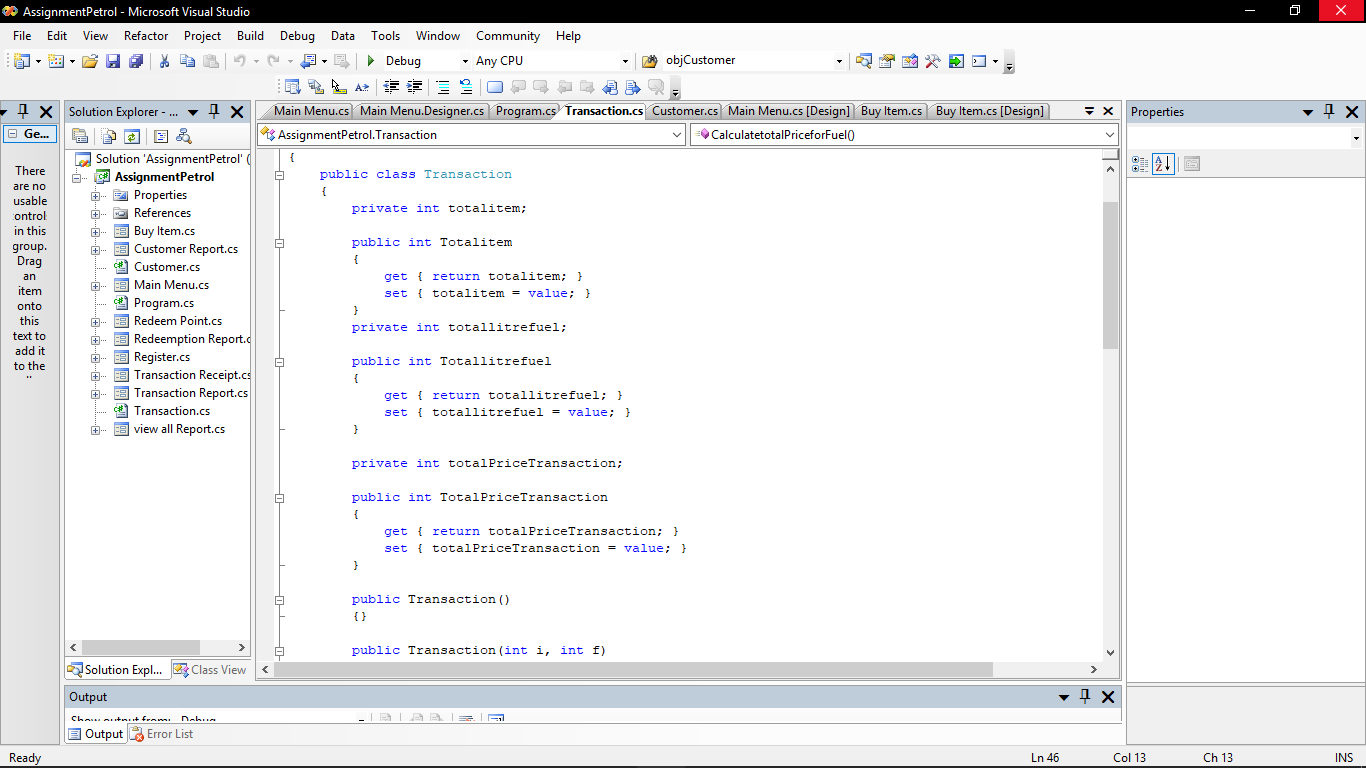
1.Class

In OOP approach that has been applied in the developing the system, it contains classes. Class is a template for creating objects which consist method and variable. Each object that is created from the class in an instance of the class. Class also contain constructor. In the developed system, there are two classes which class Customer and class Transaction.



2. Attribute

Other than that, OOP approach contains attributes. All the attribute is being declared in class. Attributes is referred to the data of a class. Attributes can be the type of primitive such as int, double, char, float, string and many more. There are many attribute that has been used in the developed system which is name, IC, address, phone no, ID Card no, current point, total price of item, total liter of fuel and total price of the transaction. Below is the example of the coding that is declared in the class Customer and Transaction.

3. Object

The other concept in OOP approach that has been implemented in the system is object. Object is created from the class. The object is created to create an instance of a class. This operation is also referred as instantiation. When the object is created, the object will be allocated in the computer memory. The object that has been used in the system is objCustomer, theCust and objTransaction. The object is being declared as below.

Customer theCust = new Customer();

Customer objCustomer = new Customer();

Transaction objTransaction = new Transaction();

4. Method

The other concept in OOP approach that has been used in the system is method. Method is referred to behavior of a class. Mostly, method is declared as public and it hold calculation that will be used in the form. The method will be called in the form by using object as it is declared as public that make it easy to be accessed inside or outside of the class. One of the method that has been implemented in the system is calculatetotalPrice(). Below is the coding of that method.

public int calculatetotalPrice()

{

return totalPriceTransaction = totalitem + (totallitrefuel \* 3);

}

5. Access modifier

Another OOP concept that has been implemented in the system is access modifier. Access modifier is referred to the keyword that indicate how an attribute or method can be accessed. There are two access modifier that has been used in the system which is public and private. Private is used as the attribute can be accessed by only the method or attribute which is are from the same class. While public is used to make the method or attributes can be accessed by code inside the class or outside of the class. Below is the example of access modifier that has been used for the method in system.

public int CalculatetotalPriceforFuel()

{

int totalPriceFuel;

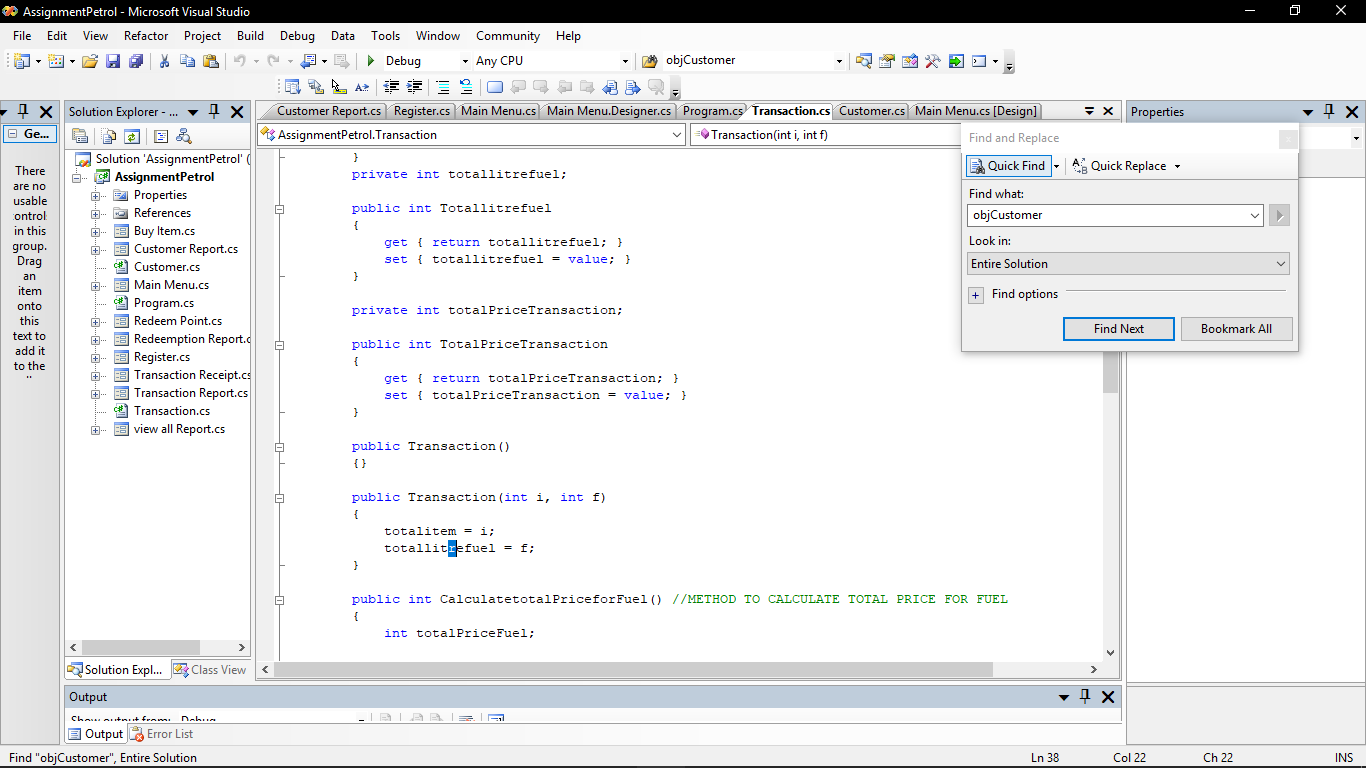
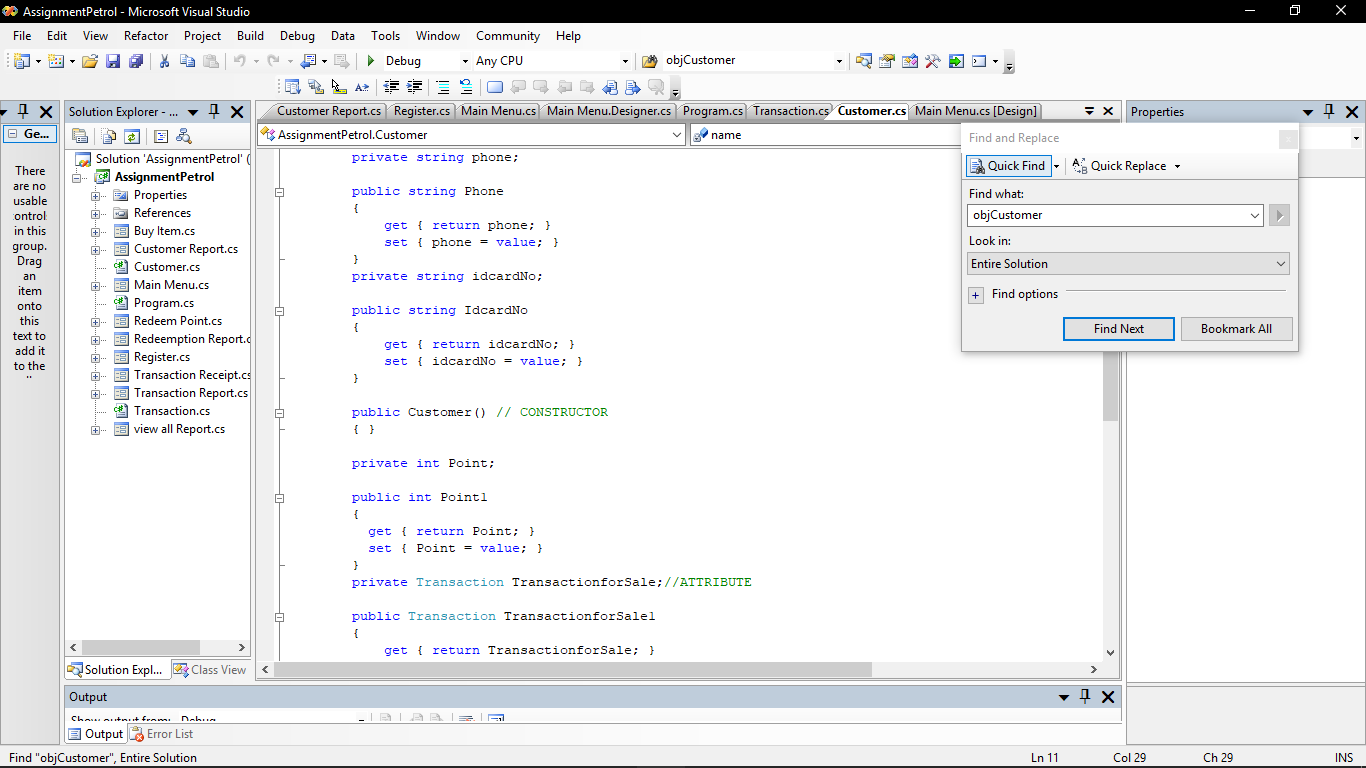
totalPriceFuel = totallitrefuel \* 3;

return totalPriceFuel;

}

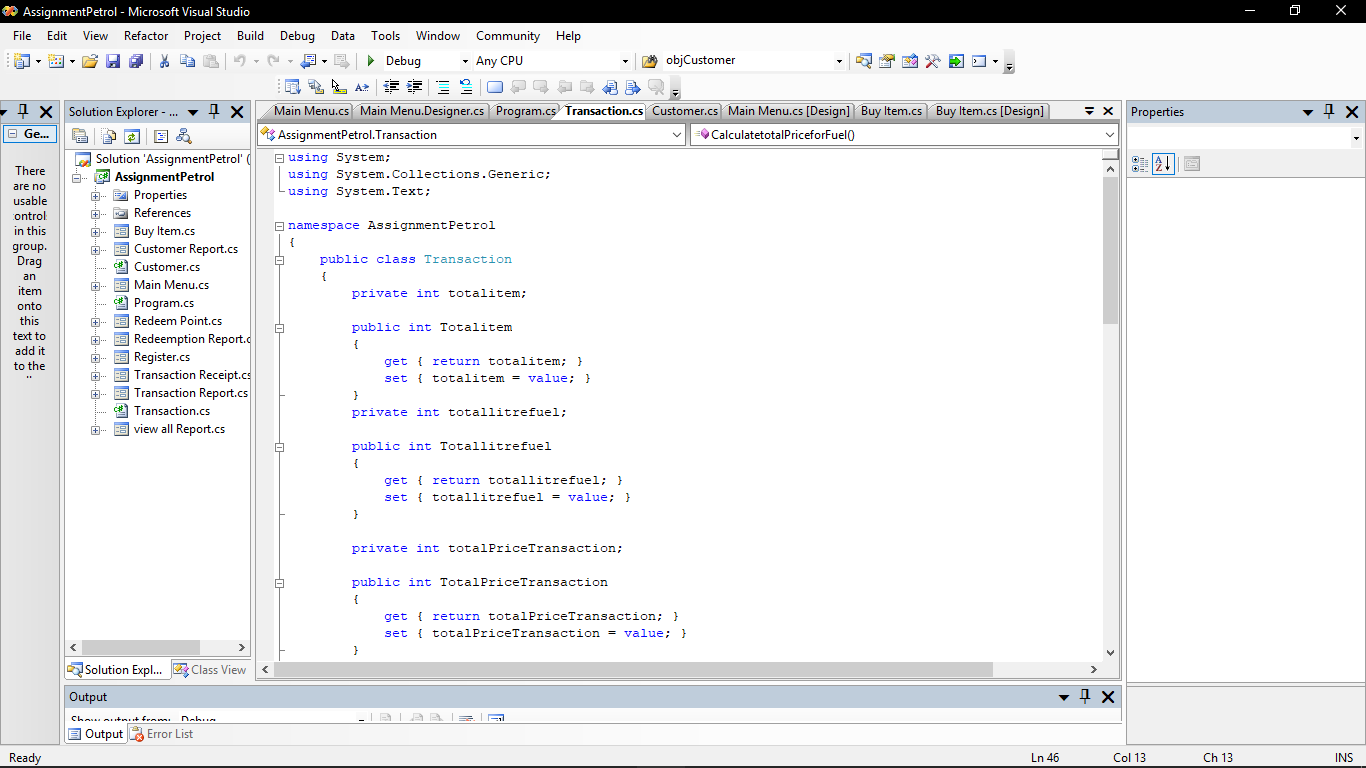
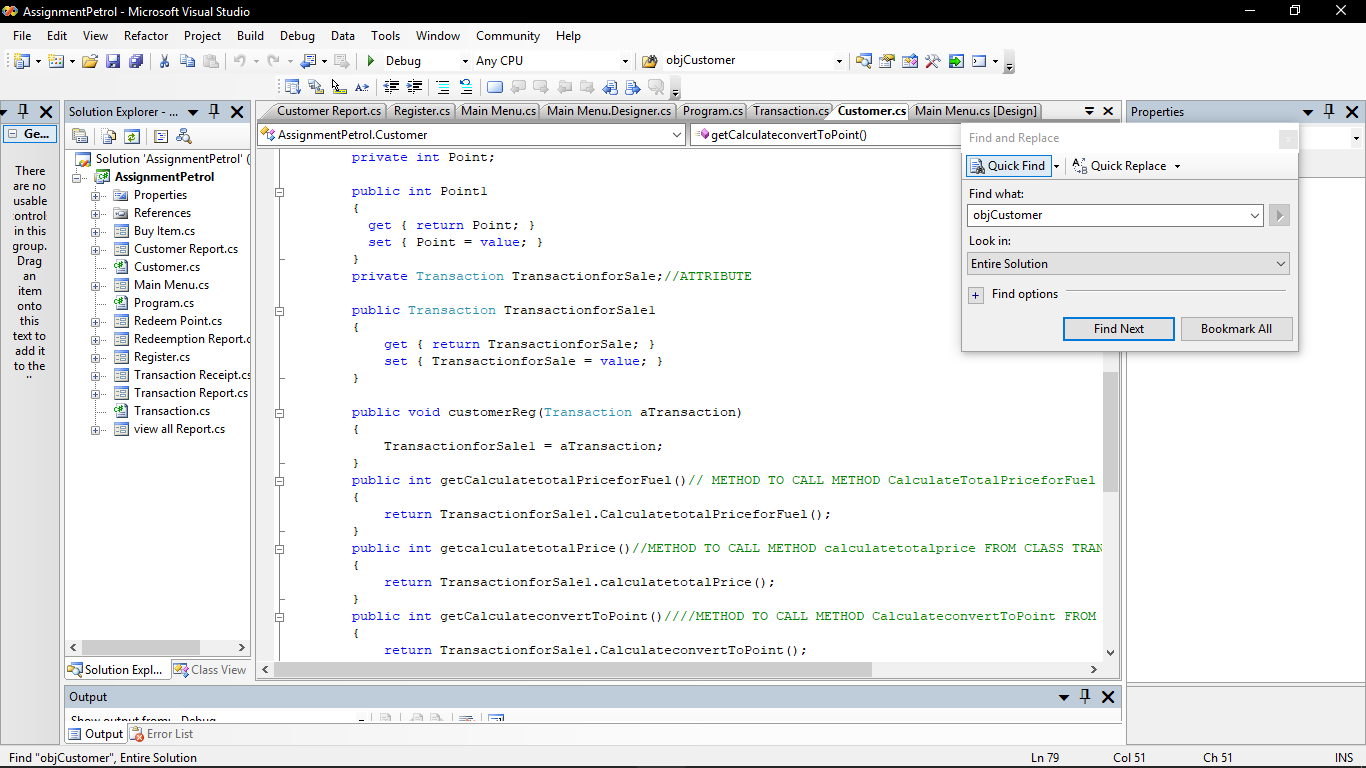
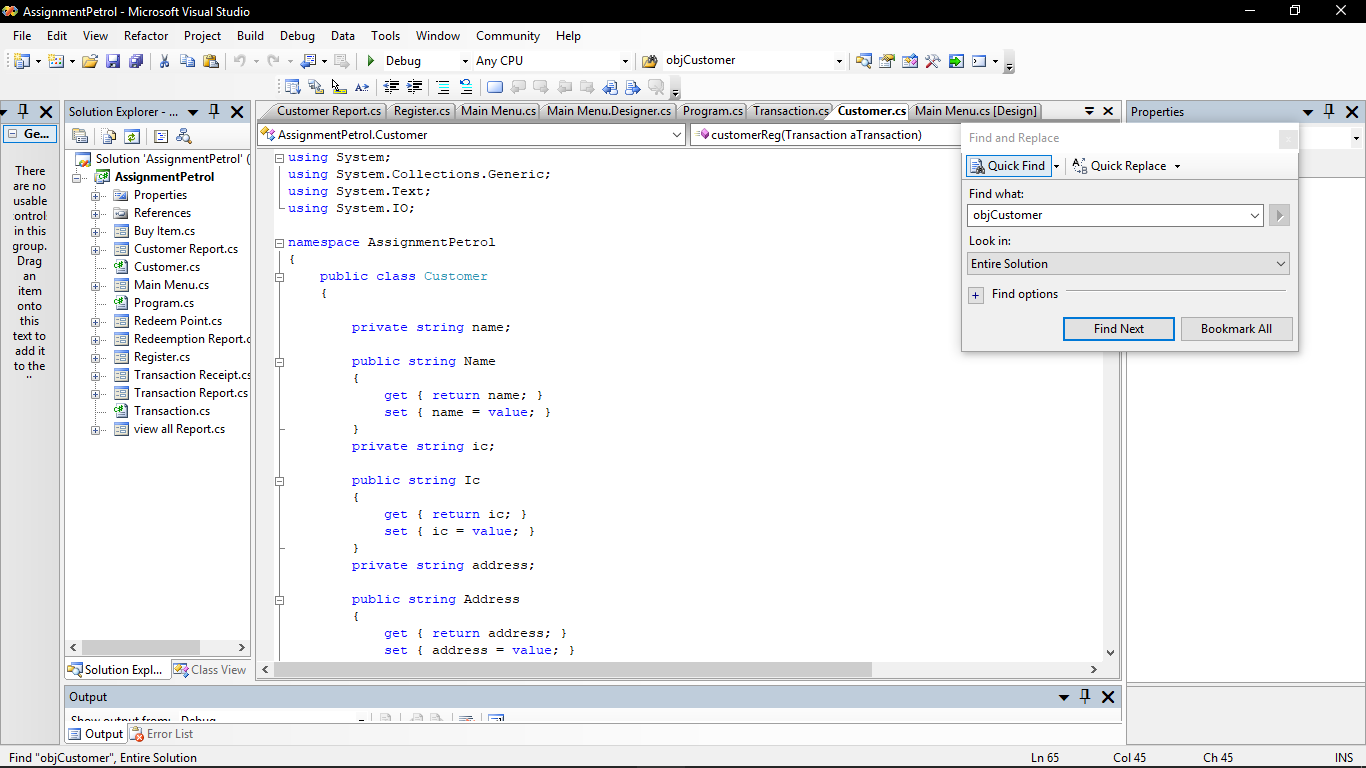
6. Constructor

The other OOP concept that has been use in the system is constructor. Constructor is a method of class or structure that initializes an object of that type. It is also an instance method that usually has the same name as the class and can be used to set values of the members of an object. Following is the coding of the constructor that has been implemented in the system.



7. Relationship

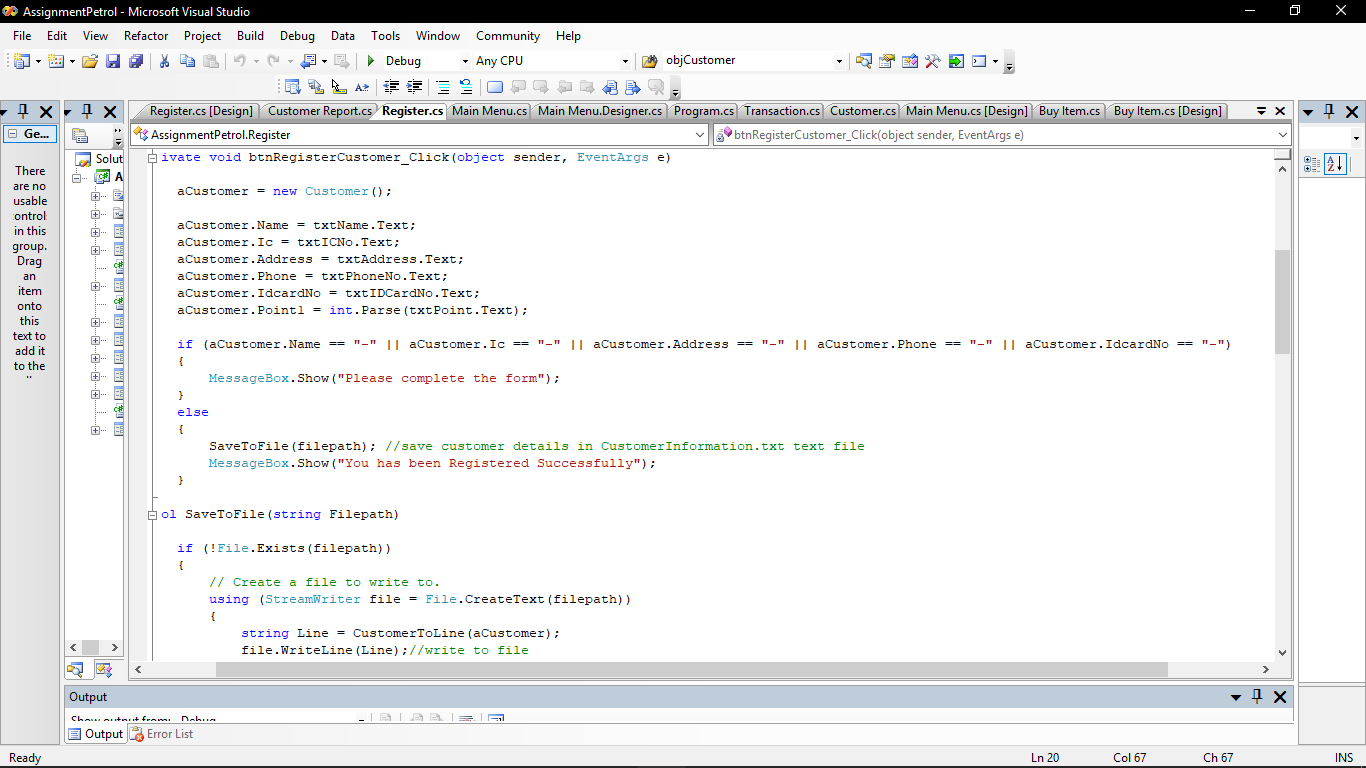
The last concept of OOP approach that has been applied in the developed system is relationship type which is simple association. Simple association is defines as a relationship between classes of objects that allows one object instance to cause another to perform an action. For example, in the system, there are two classes which Customer and Transaction. Both of the classes are associated to each other where the classes can be used without each other and there is no single owner between the classes. The class Customer will need class Transaction while Transaction need Customer. Below is the diagram that show how the both classes are associate to each other.



Customer objCustomer = new Customer();// both of the classes have their own life cycle

Transaction objTransaction = new Transaction();//they are independent of each other

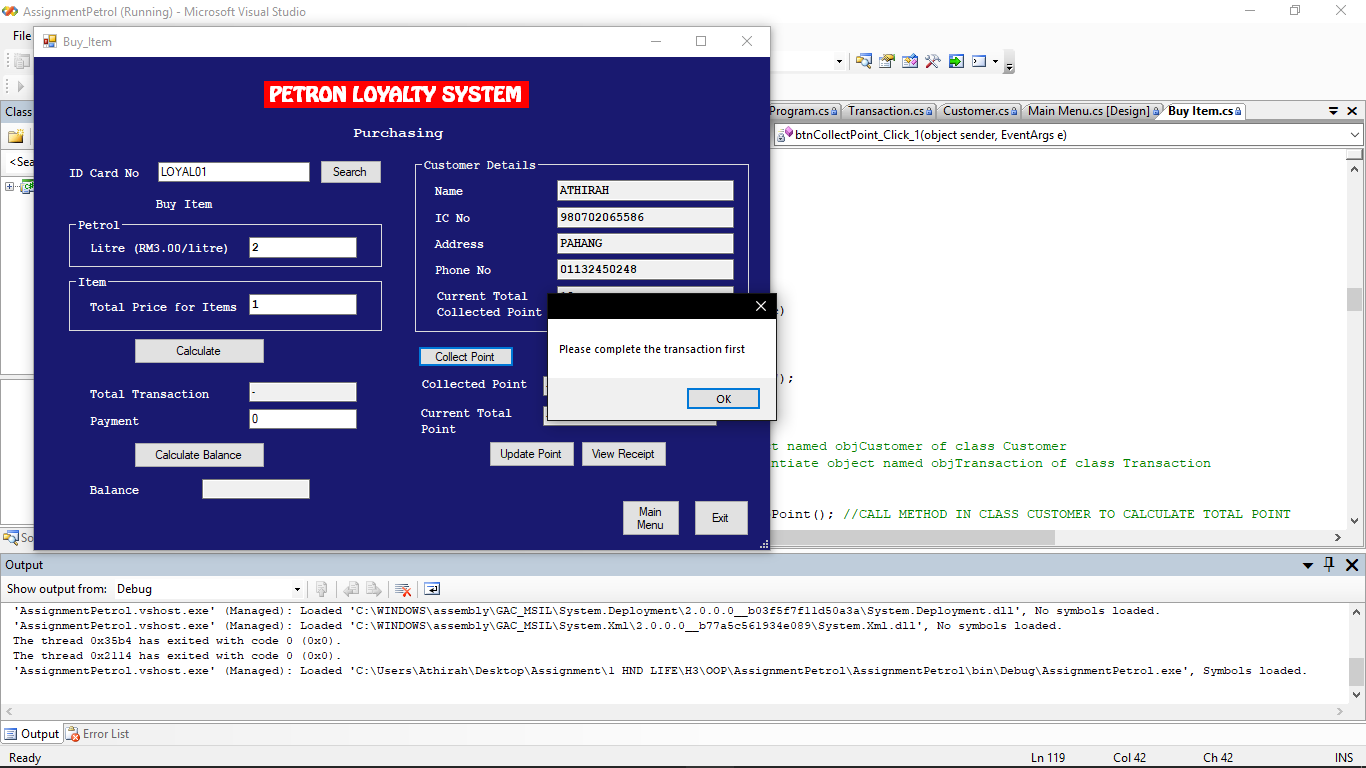
**Implement and justify error-handling or precaution imposed in your system to prevent any failure caused by inappropriate inputs from user. Provide print screen(s) for the implemented error-handling. (D2.4)**

1.

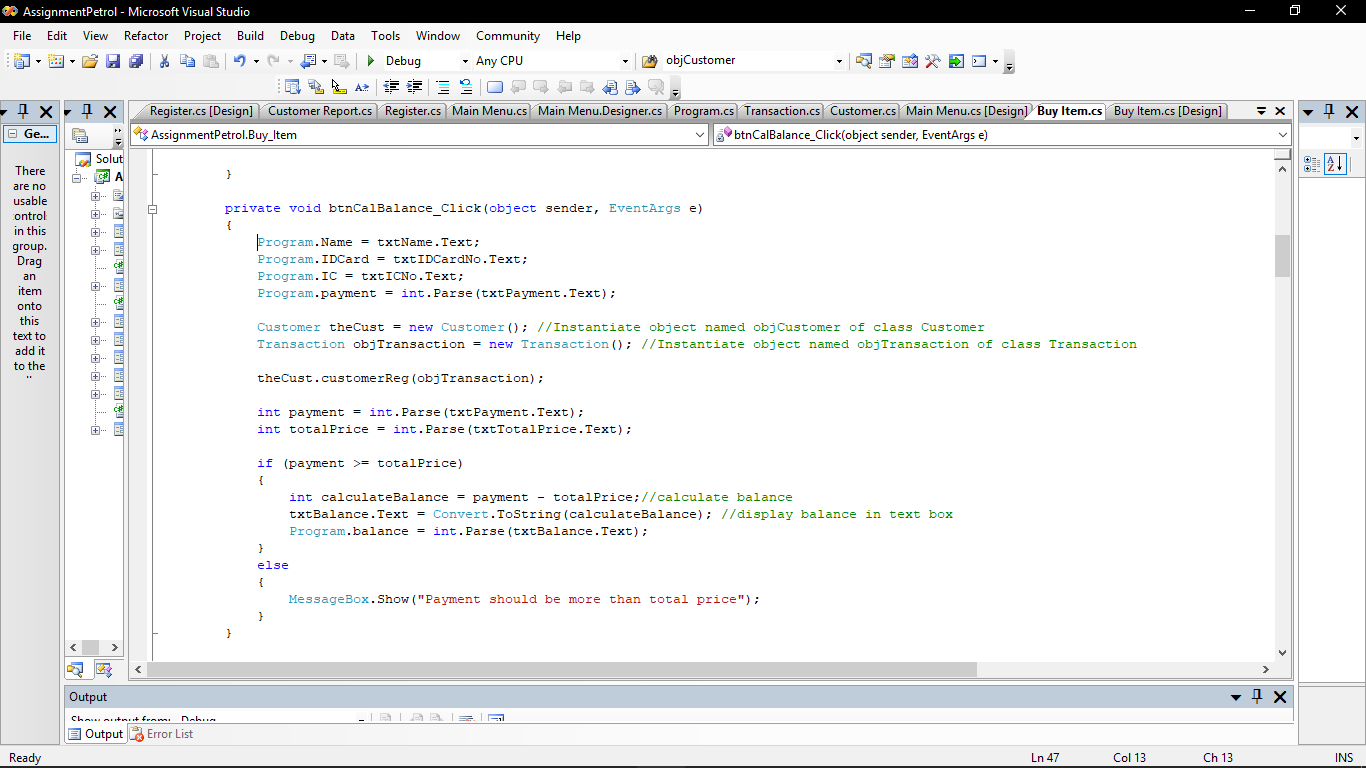


Above is the screenshot of the first error-handling that is implemented in the Customer Loyalty Program. The error-handling is implemented in the Register form where the users need to fill in all the customers’ details. If the users do not fill in one of the customers’ detail and click the register button, the message box will appear with message “Please complete the form”. But if user fill in all the customers’ details in text box, the system will save the customers’ detail in text file named CustomerInformation.txt. This error-handling is implemented to from the date save to be saved in null value.

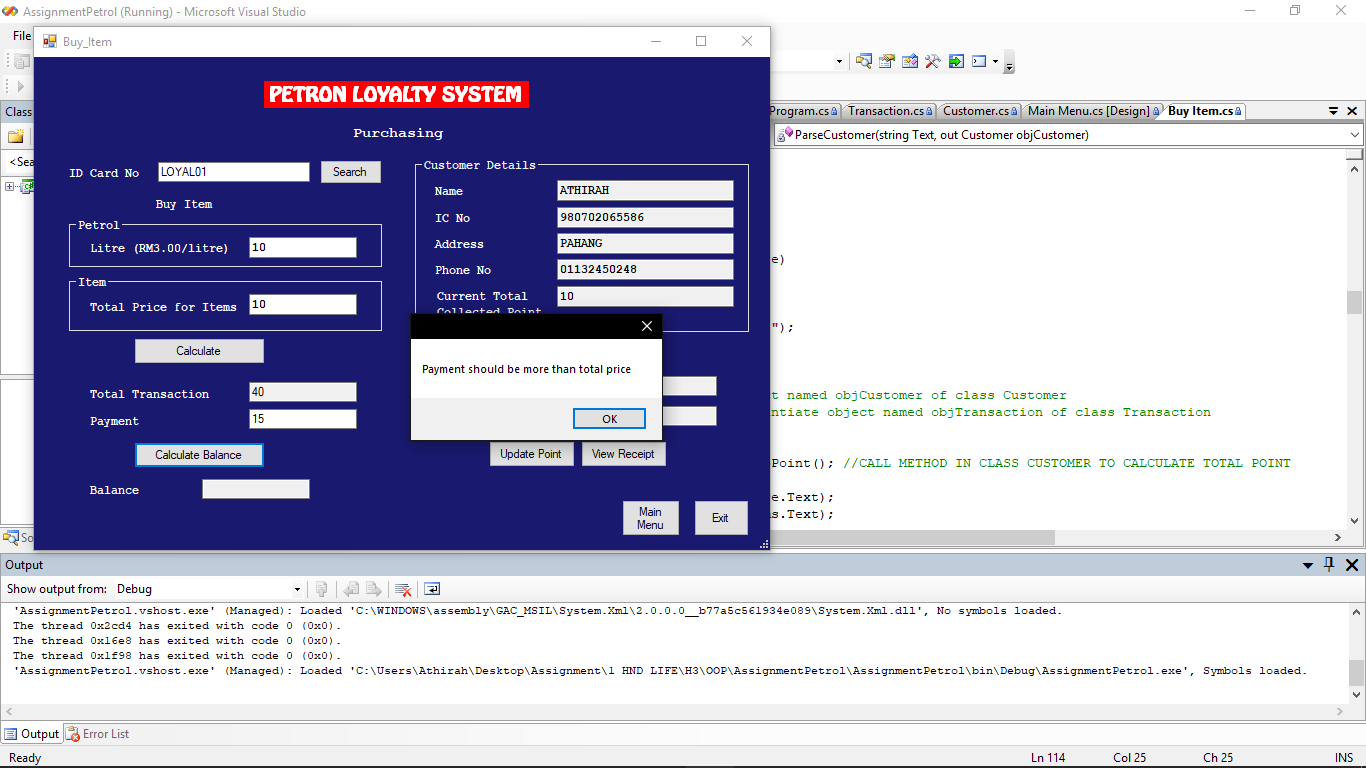
2.



Above is the screenshot of another error-handling that is implemented in the system. The error-handling is implemented in Buy Item form where user need to complete the transaction of the sale before click the button Collect Point to collect the point of the transaction. If user do not complete the transaction which the users do not enter the button calculate, but just straight forward by clicking the Collect Point button, the message box will display message “Please complete the transaction first”. But if the users complete the transaction, the total collected point from the transaction will be calculated and displayed in the text box. This error-handling is implemented because the total point cannot be calculated if the total price of the transaction is not calculated.



3.



Above is the last error-handling that is implemented in the system. The error handling is implement in the Buy Item form. It happens in the button Calculate Balance where the users need to enter the payment more than or equal to the total price. If the total price is 40, then the user only enters 15 for the payment, the message box will appear with a message “Payment should be more than total price”. But when user enter the payment more than total price, the system will calculate the balance of the payment. This error-handling is implemented because it if the payment is less than total price, the balance of the transaction will be in negative answer.