

Marty Peltz

SOFTWARE LIFE CYCLE REPORT – FOR LAB ASSIGNMENT 06

PHASE 1: SPECIFICATION (“What do we build?”)

Problem Specification

Create a program that stores a restaurant health inspector's reports in a text file. First, the program allows an inspector to enter multiple records, one per restaurant.

The records are entered using the form shown in Figure 1. As shown, each record includes four fields: a restaurant name, its address, its type of cuisine, and its health grade.

Second, after typing data for one restaurant record on the form, the inspector clicks on the button **Add to List**. The record is added to the end of a linked list in memory (that is, the linked list of restaurants records is not sorted by any field value). Simultaneously, fields in the Windows Form are emptied.

Third, when after entering the last restaurant record the inspector clicks on the button **Save List to File**, the records from the linked list are saved into a new file named **Restaurant_Reports.txt**.

Fourth, the inspector verifies data entered by him by clicking on the button **Load from File**. All records from the file appear in the read-only TextBox **Restaurant reports** in the following format (one per line):

Restaurant name; Address; Type; Health Grade

For example, a report with two records will appear one per line as:

Cosmo's Cucina; 804 W Vine St, Kalamazoo, MI; Asian; A

Bruno's Pizza & Restaurant; 1528 W Michigan Ave, Kalamazoo, MI; Fast Food; C

Finally, the inspector clicks on the button **Quit** to terminate the program (and to remove the form from the display).

Code Documentation

During lab last week, we discussed how to use Doxygen to generate documentation for your code. For this assignment, you must include documentation for your code as generated by Doxygen. You should have comments for every class, property, and method that **you** write (do not worry about the form designer code that is auto-generated by Visual Studio). Be sure to use the `///` (triple-dash) type of comments recognized by Doxygen. By default, Doxygen outputs documentation as html files. Include this folder of html files along with your code and SLC report when you submit the assignment online. You do not need to submit a hard copy of the documentation.

PHASE 2: DESIGN

2. Modules and Their Basic Structure

1. Class: Program

a. implements the form application

2. Class: LinkedList

**This class sets up the properties for the First and Last node
This class uses the method insert at back to be called for linked
list**

3. Class: Node

**implements properties the input data and the next to continue to
next node in the linked list**

4. Class: Form application

**Includes the Form with the 4 required text boxes 4 required
buttons and the richtextfield.**

For each button:

**Save to list: the button takes the input values in the text boxes and
uses the method from linkedlist insertatback to enter the values
into a list.**

**Save to file: This button will use StreamWriter to take the data
currently in the list and add it to the text file created in the
program debug file.**

**LoadFromFile: This button will take the data currently in the text
file and display it to the richtextfield using StreamReader.**

The quit button will close the program when pushed.

5. Class: RestaurantData

implements the properties of the 4 given inputs from the textboxes

PHASE 3: RISK ANALYSIS (“What can go wrong, and how bad can it be?”)

No risks (to timetable, cost, human health, etc.) are affected.

PHASE 4: VERIFICATION (“Are the algorithms correct?”)

The correct data displays and the given errors that occur are correct. This was done
by testing with the given test code and running in the command prompt.

PHASE 5: CODING

First Refinement-----

1) Class: LinkedLists

```
public LinkedLists()  
    sets first and last node to null  
public Node FirstNode  
    Property for firstNode  
public Node LastNode  
    Property for lastNode  
public void InsertAtBack(return type)  
    method that inserts the new objects to the back of the list
```

2) Class: Node

```
public RestaurantData Data  
    property for data  
public Node Next  
    property for next  
public Node(return)  
    Returns property call
```

3) Class: Program

Empty

4) Class: RestaurantData

```
public string Name  
    property for name  
public string Address  
    property for address  
public string Type  
    property for type  
public string Grade  
    property for grade
```

5) Class: Form Application

Methods:

```
    Addtolist  
        RestaurantData resturant = new RestaurantData(input  
data)  
        call insertatback method from linkedlists  
    Savelisttofile  
        StreamWriter _____ = new StreamWriter  
        if = null close project  
        else  
            Write objects to file in {0}; {1}; {2}; {3}\n  
format  
    LoadFromFile  
        StreamReader _____ = new StreamReader  
        while(_____.endofstream == false)
```

```

        load file from text document
quit
this.close()

```

Final Refinement-----

```

{
    /// <summary>
    /// This is the Class that defines the Properties for the objects of Name, Address,
    Type, and Grade.
    /// </summary>
    class RestaurantData
    {
        /// <summary>
        /// Property for Name
        /// </summary>
        public string Name { get; set; }
        /// <summary>
        /// Property for Address
        /// </summary>
        public string Address { get; set; }
        /// <summary>
        /// Property for Type
        /// </summary>
        public string Type { get; set; }
        /// <summary>
        /// Property for Grade
        /// </summary>
        public string Grade { get; set; }
        /// <summary>
        /// method that sets up the Properties for Class RestaurantData
        /// </summary>
        /// <param name="n">string returned for Name</param>
        /// <param name="a">string returned for Address</param>
        /// <param name="t">string returned for Type</param>
        /// <param name="g">string returned for Grade</param>
        public RestaurantData(string n, string a, string t, string g)
        {
            Name = n;
            Address = a;
            Type = t;
            Grade = g;
        }
    }
}
{
    /// <summary>
    /// This is the Class that implements the LinkedList for reading through the
    different textboxes in the windows form
    /// This class has methods for properties, inserting at back and empty strings
    /// </summary>
    class LinkedLists

```

```

{
    /// <summary>
    /// property for the first node
    /// </summary>
    private Node firstNode;
    /// <summary>
    /// property for the last node
    /// </summary>
    private Node lastNode;
    /// <summary>
    /// This is the linked list method that sets first node and last node to null
    /// </summary>
    public LinkedLists()
    {
        firstNode = lastNode = null;
    }
    /// <summary>
    /// property implementation for first node
    /// </summary>
    public Node FirstNode
    {
        get
        {
            return firstNode;
        }
        set
        {
            firstNode = value;
        }
    }
    /// <summary>
    /// property implementation for last node
    /// </summary>
    public Node LastNode
    {
        get
        {
            return lastNode;
        }
        set
        {
            lastNode = value;
        }
    }
    /// <summary>
    /// Method that inserts a new node at the back of the linked list
    /// </summary>
    /// <param name="input">returns the input as the new node</param>
    public void InsertAtBack(ResistantData input)
    {
        if (lastNode == null)
            firstNode = lastNode = new Node(input, null);
        else
            lastNode = lastNode.Next = new Node(input, null);
    }
    /// <summary>
    /// Method that determines if the string is empty
    /// </summary>

```

```

    /// <returns>if the string is empty it sets firstnode to null</returns>
    public bool IsEmpty()
    {
        return FirstNode == null;
    }
}

namespace RestaurantHealthR
{
    /// <summary>
    /// Class that Sets up the properties for the given data and moving to the next node
    /// </summary>
    class Node
    {
        /// <summary>
        /// property for Data
        /// </summary>
        public RestaurantData Data { get; set; }
        /// <summary>
        /// property for next
        /// </summary>
        public Node Next { get; set; }
        /// <summary>
        /// method that implements the properties
        /// </summary>
        /// <param name="dataValue">returns Data</param>
        /// <param name="nextNode">returns Next</param>
        public Node(RestaurantData dataValue, Node nextNode)
        {
            Data = dataValue;
            Next = nextNode;
        }
    }
}

{
    /// <summary>
    /// This is the class for the Form application it includes methods for the five
    different buttons
    /// AddToList button, SaveListToFile button, LoadFromFile button, Quit button, and
    Clear Button
    /// </summary>
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        LinkedList setList = new LinkedList();
        /// <summary>
        /// Method that Adds the given data in the text boxes to the list
        /// this method implements the 4 text boxes and inserts at back calling the
insert at back method
        /// It also clears the Textboxes above
        /// </summary>

```

```

/// <param name="sender"></param>
/// <param name="e"></param>
private void AddToListButton_Click(object sender, EventArgs e)
{
    //richTextBox1.Text = String.Format (RNt ext box + "; ");
    //richTextBox1.Text = String.Format (Addr esst ext box + "; ");
    //richTextBox1.Text = String.Format (Typet ext box + "; ");
    //richTextBox1.Text = String.Format (HGt ext box + "\n");
    Rest aur ant Dat a r est ur ant = new Rest aur ant Dat a(RNt ext box.Text ,
Addr esst ext box.Text , Typet ext box.Text , HGt ext box.Text);
    set Li st . I nser t At Back(r est ur ant);

    RNt ext box. Reset Text ();
    Addr esst ext box. Reset Text ();
    Typet ext box. Reset Text ();
    HGt ext box. Reset Text ();
}
/// <summary>
/// Method that Saves the list generated to a text file called
Rest aur ant _Report s.txt
/// This method determines if the list is null, if not then it writes the data to
the text file in a specific format with ; and spaces
/// It also clears the Textboxes above
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void SaveListToFileButton_Click(object sender, EventArgs e)
{
    StreamWriter savelist = new StreamWriter ("Rest aur ant _Report s.txt");
    if (set Li st . I sEmpty())
    {
        savelist . Cl ose();
        ri chText Box. Text = St ri ng. For mat ("No Dat a I n Li st");
    }
    else
    {
        Node current = set Li st . Fi rst Node;
        Rest aur ant Dat a typeObj ect;

        while(current != null)
        {
            typeObj ect = current . Dat a;

            savelist . Wi te(typeObj ect . Name + "; ");
            savelist . Wi te(typeObj ect . Addr ess + "; ");
            savelist . Wi te(typeObj ect . Type + "; ");
            savelist . Wi te(typeObj ect . Grade + "; ");
            current = current . Next;
        }
        savelist . Cl ose();
    }
    RNt ext box. Reset Text ();
    Addr esst ext box. Reset Text ();
    Typet ext box. Reset Text ();
    HGt ext box. Reset Text ();
}
/// <summary>
/// This method takes the data from the text file Rest aur ant _Report s.txt

```

```

/// and displays it into the richtext box in the form application
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void LoadFromFileButton_Click(object sender, EventArgs e)
{
    StreamReader loadfile = new StreamReader("Restaurant_Reports.txt");
    string restaurant = loadfile.ReadLine();

    while (loadfile.EndOfStream == false)
        restaurant = String.Format("{0}/n{1}", restaurant, loadfile.ReadLine());

    richTextBox.Text = restaurant;
}
/// <summary>
/// this is the method for the quit button that closes the program if clicked
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void QuitButton_Click(object sender, EventArgs e)
{
    this.Close();
}

/// <summary>
/// this is the method for the clear button it will clear all the text boxes if
clicked
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void ClearButton_Click(object sender, EventArgs e)
{
    FirstName.ResetText();
    Address.ResetText();
    Type.ResetText();
    HGT.ResetText();
}
}
}

```

static class Program

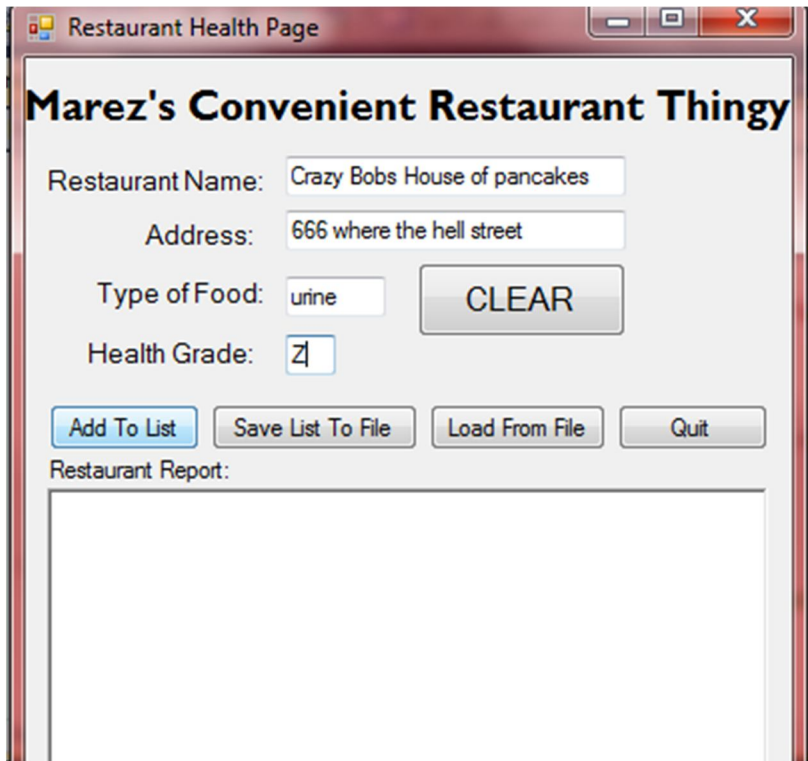
```

{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
    }
}
}

```

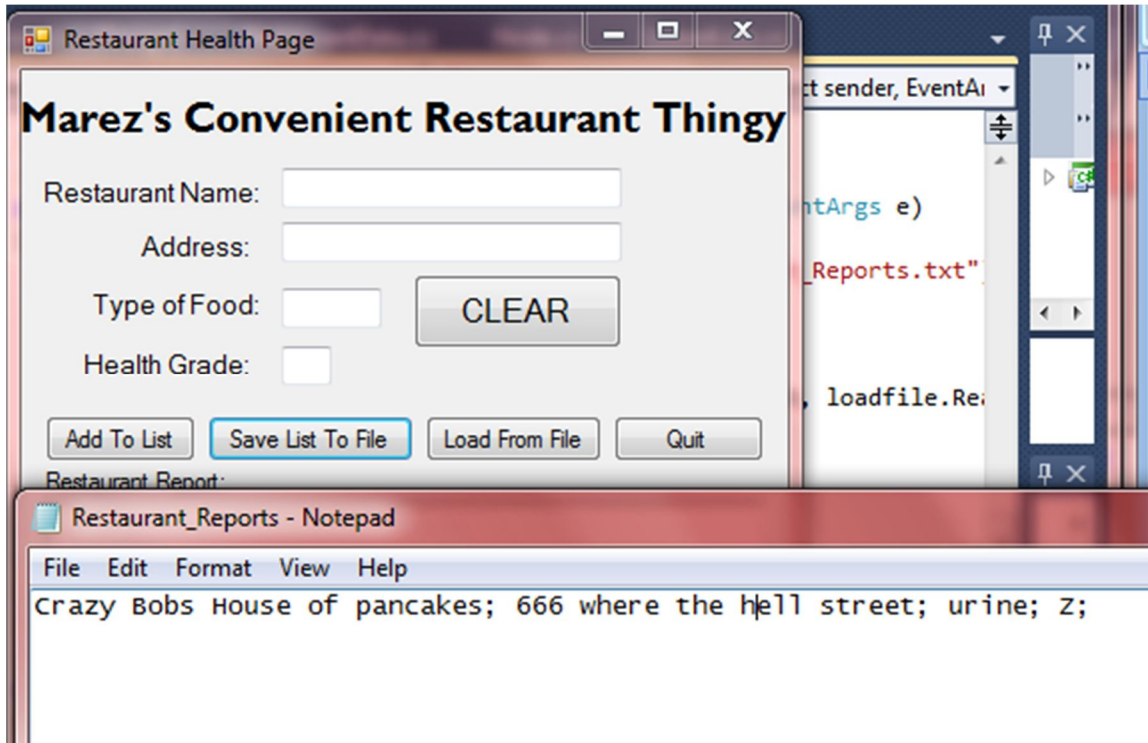

PHASE 6: TESTING (“Did we build it correctly?”)

The program, ran and tested fine, the windows application successfully took data in, formed it into a linked list when the save to list button was clicked. That list was successfully added to the text file when addtofile button was pressed. The richtextbox correctly displayed output when prompted to display the current data on the textfile.



The screenshot shows a Windows application window titled "Restaurant Health Page". The window contains a form titled "Marez's Convenient Restaurant Thingy". The form has the following fields and controls:

- Restaurant Name:
- Address:
- Type of Food:
- Health Grade:
- Buttons:
- Restaurant Report:



PHASE 7: REFINING THE PROGRAM (“Add bells and whistles to the program”)

A clear button was added to clear all the text files, i also did this whenever one of the buttons was clicked so new data could be implemented

PHASE 8: PRODUCTION

I prepared a copy of the entire program for Lab TA's evaluation, as specified by the TA. Then, I sent electronically the copy to the Lab TA, and printed off a physical copy to hand in to the lab TA.

PHASE 9: MAINTENANCE

I will use the TA's feedback in order to make modifications to improve the program. The program will be maintained as necessary.