**Peer Evaluation for Lab 2 – Chapters 13, 14 and 15**

|  |  |
| --- | --- |
| Your name: (Your lab is the one being evaluated) | **Steven Wilson** |
| Name(s) of peer evaluator(s) | **Me (I’ve been told we can do a self evaluation. I hope that’s true.)** |
| Date: | **04/28/2019** |

Instructions  
You should have already completed Lab 2. After you and a peer have evaluated your work, you will submit this evaluation along with screen shots and source code indicated in moodle. You may make corrections to your work as a result of the evaluation.

|  |  |
| --- | --- |
| ***In Class Exercises – 15-1 #3*** | |
| Completed Exercise?   * Created a class diagram in visual studio? Screen shot included? * Customer class implements the ICloneable interface?   + ICloneable is added to the class heading?   + Clone method is implemented with the signature expected by the framework? * Test calls Clone to make copies of the customer object. * Screen shot of test running is included? * Source code includes Customer class as well as test for Clone? | **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.** |
| ***In Class Exercises – 15-2 #2*** | |
| Completed Exercise?   * Created a class diagram in visual studio? Screen shot included? * CustomerList class implements the IEnumerable interface?   + IEnumerable is added to the class heading?   + GetEnumerator method is implemented with the signature expected by the framework? * Test fills a CustomerList and iterates through the list using a foreach loop? * Screen shot of test running is included? * Source code includes CustomerList class as well as test for the foreach loop? | **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.** |

|  |  |
| --- | --- |
| ***In Class Exercises – 13-1*** | |
| Completed Exercise?   * Created a class diagram in visual studio? Screen shot included? * CustomerList class includes delegate and event?   + Delegate ChangeHander is declared?   + Event Changed is declared?   + All methods that change the list “broadcast” the Changed event? * User interface responds to the Changed event?   + Method is written to handle the changed event, the signature of which matches the delegate?   + Form load event handler “wires up” the method to the event? * Screen shot of application running is included? * Source code includes CustomerList class as well as event handlers/methods from UI? | **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.**  **Yes.** |

|  |  |
| --- | --- |
| ***In Class Exercises – AlmostEmpty, Deck and Blackjack (You will complete this as you work on the UI for BlackJack in Lab 3)*** | |
| Completed Exercise?   * Created a class diagram in visual studio? Screen shot included? * Deck class includes delegate and event?   + Delegate EmptyHander is declared?   + Event AlmostEmpty is declared?   + Empty HandleEmpty method is created?   + Constructor “wires up” the AlmostEmpty event and the empty HandleEmpty method?   + Deal method “broadcasts” the AlmostEmpty event? * User interface responds to the AlmostEmpty event?   + Method is written to handle the AlmostEmpty event, the signature of which matches the delegate?   + Form load event handler “wires up” the method to the event? * Screen shot of application running is included? * Source code includes Deck class as well as event handlers/methods from UI that are related to handling the event? | **I wasn’t there for this, and didn’t realize we had to do it. Now I don’t have enough time, so I cannot.** |

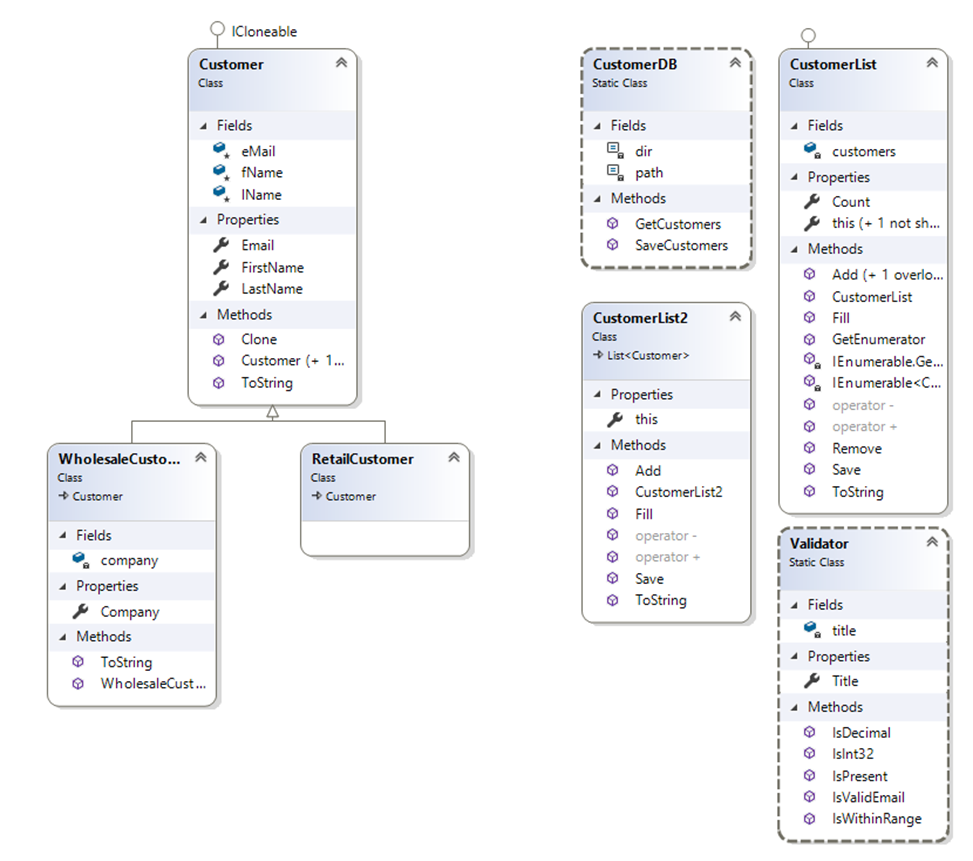
|  |  |
| --- | --- |
| ***Programming style for all programs*** | |
| Is proper indentation used? Is each property/method indented properly? Is each control structure indented properly? | **Yes.**  **Yes.** |
| Are comments used appropriately? | **Yes.** |
| Do variable names use camel case? (camelCase for example) | **Yes.** |
| Do property/method names use Title Case (or Pascal Case?) | **Yes.** |

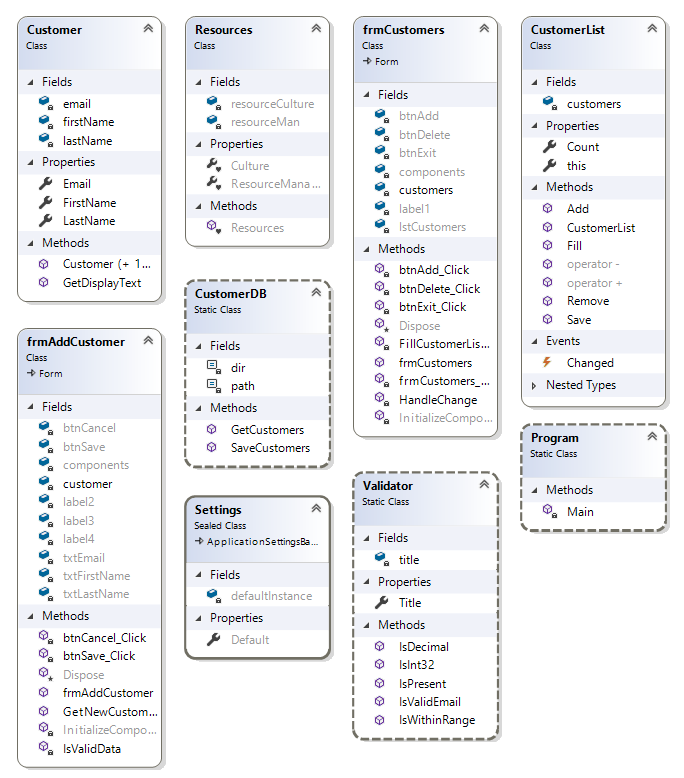
General comments and notes:

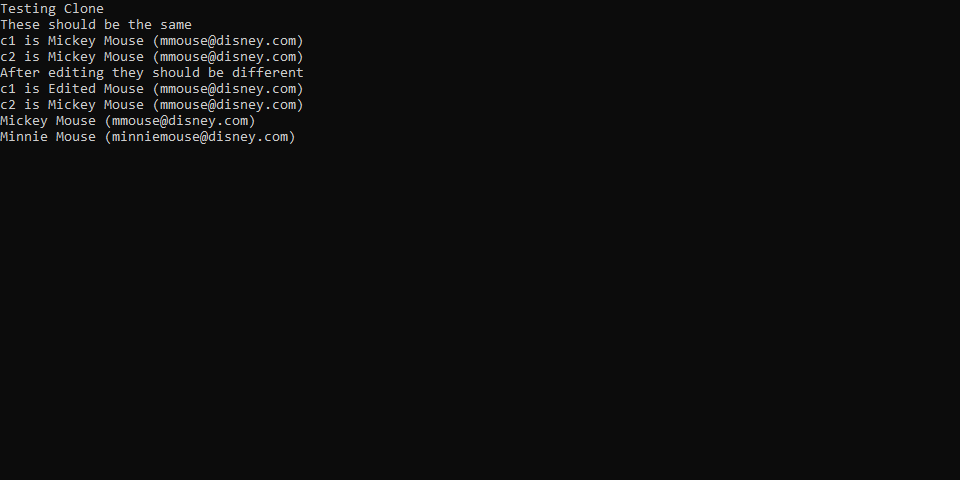
**Here’s the GitHub:**

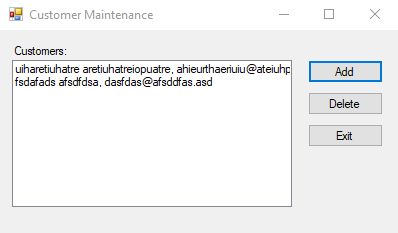
https://github.com/drazhok/CS234N

Screen Shots and Source Code





****

****

**Interfaces**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace CustomerMaintenanceClasses

{

public class CustomerList : IEnumerable<Customer>

{

private List<Customer> customers;

public CustomerList()

{

customers = new List<Customer>();

}

public int Count => customers.Count;

public void Fill()

{

customers = CustomerDB.GetCustomers();

}

public void Save()

{

CustomerDB.SaveCustomers(customers);

}

public void Add(Customer customer)

{

customers.Add(customer);

//Changed(this);

}

public void Add(string first, string last, string email)

{

Customer c = new Customer(first, last, email);

customers.Add(c);

//Changed(this);

}

public void Remove(Customer customer)

{

customers.Remove(customer);

//Changed(this);

}

public override string ToString()

{

string output = "";

foreach (Customer c in customers)

{

output += c.ToString() + "\n";

}

return output;

}

public Customer this[int i]

{

get

{

if (i < 0)

throw new ArgumentOutOfRangeException("i");

else if (i >= customers.Count)

throw new ArgumentOutOfRangeException("i");

return customers[i];

}

set

{

customers[i] = value;

//Changed(this);

}

}

public Customer this[string email]

{

get

{

foreach (Customer c in customers)

{

if (c.Email == email)

return c;

}

return null;

}

}

public static CustomerList operator + (CustomerList cl, Customer c)

{

cl.Add(c);

return cl;

}

public static CustomerList operator - (CustomerList cl, Customer c)

{

cl.Remove(c);

return cl;

}

public IEnumerator<Customer> GetEnumerator()

{

foreach (Customer customer in customers)

yield return customer;

}

IEnumerator<Customer> IEnumerable<Customer>.GetEnumerator()

{

throw new NotImplementedException();

}

IEnumerator IEnumerable.GetEnumerator()

{

throw new NotImplementedException();

}

}

}

**Events**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CustomerMaintenance

{

public partial class frmCustomers : Form

{

public frmCustomers()

{

InitializeComponent();

}

private CustomerList customers = new CustomerList();

private void frmCustomers\_Load(object sender, EventArgs e)

{

customers.Changed += new CustomerList.ChangeHandler(HandleChange);

customers.Fill();

FillCustomerListBox();

}

private void FillCustomerListBox()

{

lstCustomers.Items.Clear();

for (int i = 0; i < customers.Count; i++)

{

Customer c = customers[i];

lstCustomers.Items.Add(c.GetDisplayText());

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

frmAddCustomer addCustomerForm = new frmAddCustomer();

Customer customer = addCustomerForm.GetNewCustomer();

if (customer != null)

{

customers += customer;

}

}

private void btnDelete\_Click(object sender, EventArgs e)

{

int i = lstCustomers.SelectedIndex;

if (i != -1)

{

Customer customer = (Customer)customers[i];

string message = "Are you sure you want to delete "

+ customer.FirstName + " " + customer.LastName + "?";

DialogResult button = MessageBox.Show(message, "Confirm Delete",

MessageBoxButtons.YesNo);

if (button == DialogResult.Yes)

{

customers -= customer;

}

}

}

private void HandleChange(CustomerList customers)

{

customers.Save();

FillCustomerListBox();

}

private void btnExit\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CustomerMaintenance

{

public class CustomerList

{

private List<Customer> customers;

// declare the delegate and then the event here

public delegate void ChangeHandler(CustomerList customers);

public event ChangeHandler Changed;

public CustomerList()

{

customers = new List<Customer>();

}

public int Count

{

get

{

return customers.Count;

}

}

public Customer this[int i]

{

get

{

return customers[i];

}

set

{

customers[i] = value;

Changed(this);

}

}

public void Fill()

{

customers = CustomerDB.GetCustomers();

}

public void Save()

{

CustomerDB.SaveCustomers(customers);

}

public void Add(Customer customer)

{

customers.Add(customer);

Changed(this);

}

public void Remove(Customer customer)

{

customers.Remove(customer);

Changed(this);

}

public static CustomerList operator + (CustomerList c1, Customer c)

{

c1.Add(c);

return c1;

}

public static CustomerList operator - (CustomerList c1, Customer c)

{

c1.Remove(c);

return c1;

}

}

}