**Week 7, Classes based on other classes.  Inheritance. -**

Things to do this week:

* **Reading Quiz 8**
  + Read chapter 14 in the C# text.
  + Complete Reading Quiz 8. Each reading quiz consists of 10 multiple choice questions taken from the topic. You may take the reading quiz 4 times and the average of your scores will be used in the calculation of your grade. Each quiz is open book but because you only have 40 minutes to complete each attempt, you won't complete the quiz if you haven't read the materials prior to attempting the quiz.
* **Lab 7**
  + Complete exercise 14 - 1, parts 1 - 4.  Modify the console application you wrote to test the WholesaleCustomer and RetailCustomer classes.  Test the constructor for each class as well as any properties or methods that were added to each class.
  + EXTRA CREDIT:  Complete exercise 14 - 2.  Re-rum the console application you wrote to test the new implementation of the CustomerList class.  Note that there are no new methods or properties so you shouldn't have to change the test program at all.  Add code to the tester for the CustomerList class that adds several WholesaleCustomers and several RetailCustomers to a CustomerList and then prints the list.
  + Complete the BlackJackHand class as discussed in class.  Write a console application to test the class.
  + EXTRA CREDIT:  Create a console application in which one player plays blackjack against the dealer.
  + Participate in the forum - Lab 7 Questions as necessary.
  + Complete a peer evaluation for your work in lab 7.  Make any corrections necessary based on your peer evaluation.
  + Submit lab 7.
* **Lab 8**
  + Create a GUI application in which at least one player plays blackjack against the dealer.
  + Participate in the forum - Lab 8 Questions as necessary.
  + Complete a peer evaluation for your work in lab 8.  Make any corrections necessary based on your peer evaluation.
  + Submit lab 8.

**Lab 7 Instructions –**

The objective of this lab is to familiarize you with creating classes that use inheritance in C#.  This information is contained in chapter 14 of your text.

Complete the programming problems described below.  For each of the problems, make a copy of the solutions you created for labs 6 and 7.  Modify the new copy as you complete labs 8 and 9.

* Complete exercise 14 - 1, parts 1 - 4.  Modify the console application you wrote to test the WholesaleCustomer and RetailCustomer classes.  Test the constructor for each class as well as any properties or methods that were added to each class.
* Complete the BlackJackHand class as discussed in class.  Write a console application to test the class.
  + BJHand
    - BJHand() *>>* *calling the default constructor on base class, if it’s empty “BJHand () {}”*
    - int Score *>> Algorithm, not a running total, recalculate every time method is called.* 
      * *int Score = 0;*
      * *public int Score()* ***method, if you’re doing a property, just a getter not a setter, score is an instance variable.***
      * *foreach (Card c in cards)* ***make sure “cards” is protected in hand class***

{

*if (c.IsFaceCard())*

*score += 10;*

*else*

*score += c.Value;*

*}*

*If (HasAce && score <= 11)* ***don’t need object in front if within own class***

*{*

*Score += 10;*

*}*

* + - bool HasAce() *>> 1 line of code, refer to HasCard (Value) from Concentration*
    - bool IsBusted() *>> if score is over 21*
* ~~Create a console application in which one player plays blackjack against the dealer.~~

A maximum of 20 points will be awarded for the lab:

* The problems completed as a group in class will earn a maximum of 12 points.
* All problems will earn a maximum of 20 points.

In class section students should:

* Sketch a class diagram that illustrates the specification of the class.
* Complete the implementation, test and debug the class in Visual Studio.  Add a class diagram
* Download the peer evaluation form for lab 7.  Complete the peer evaluation with a classmate using the form as a guide.  Include with the peer evaluation document:
  + screen shots illustrating the functionality of the application in problem
  + the source code for the classes and test programs that you wrote in each problem
  + class diagram for each class you
* Upload the peer evaluation document you created in moodle.

**Lab 8 Instructions –**

The objective of this lab is to allow you to create an Object Oriented Windows Forms application that is slightly more complex in C#.

Complete the a GUI version of BlackJack that allows at least one user to compete against the Dealer.  Use the classes you created and tested in earlier labs as well as the user interface you designed several weeks ago.

A maximum of 20 points will be awarded for the lab.

In class section students should:

* Create an Action Control Event (ACE) chart for the application.
* Create a list of form level variables the application will use.
* Create a list of methods that the application will use.
* Write an algorithm in pseudocode that describes the logic involved in any method or event handler that is logically complex
* Complete the implementation, test and debug the application in Visual Studio.
* Download the peer evaluation form for lab 9.  Complete the peer evaluation with a classmate using the form as a guide.  Include with the peer evaluation document:
  + screen shots illustrating the functionality of the applicatioin in problem
  + the source code for the methods, event handlers and variable declarations that you wrote to solve the problem.  It is NOT necessary to include source code for your classes UNLESS you created new classes that were not part of earlier labs.
* Upload the peer evaluation document you created in moodle.

**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;**

**using CardClassLibrary;**

**namespace BlackJack**

**{**

**public partial class BlackJack : Form**

**{**

**private Deck deck;**

**private BlackJackHand player, dealer;**

**public BlackJack()**

**{**

**InitializeComponent();**

**}**

**private void LoadPlayerHand()**

**{**

**for (int i = 0; i < player.NumCards; i++)**

**{**

**LoadPlayerCard(i);**

**}**

**for (int i = player.NumCards; i < 5; i++)**

**{**

**HidePlayerCard(i);**

**}**

**}**

**private void LoadDealerHand()**

**{**

**for (int i = 0; i < dealer.NumCards; i++)**

**{**

**LoadDealerCard(i);**

**}**

**for (int i = dealer.NumCards; i < 5; i++)**

**{**

**HideDealerCard(i);**

**}**

**}**

**private void LoadPlayerCard(int i)**

**{**

**PictureBox card = (PictureBox)this.Controls["pCard" + i];**

**card.Image = Image.FromFile(System.Environment.CurrentDirectory + "\\Cards\\" + player.GetCard(i).Filename);**

**card.Visible = true;**

**}**

**private void LoadDealerCard(int i)**

**{**

**PictureBox card = (PictureBox)this.Controls["dCard" + i];**

**card.Image = Image.FromFile(System.Environment.CurrentDirectory + "\\Cards\\" + dealer.GetCard(i).Filename);**

**card.Visible = true;**

**}**

**private void LoadDealerCardBack(int i)**

**{**

**PictureBox card = (PictureBox)this.Controls["dCard" + i];**

**card.Image = Image.FromFile(System.Environment.CurrentDirectory + "\\cards\\" + "black\_back.jpg");**

**}**

**public void HidePlayerCard(int i)**

**{**

**PictureBox card = (PictureBox)this.Controls["pCard" + i];**

**card.Visible = false;**

**card.Enabled = false;**

**}**

**private void HideDealerCard(int i)**

**{**

**PictureBox card = (PictureBox)this.Controls["dCard" + i];**

**card.Visible = false;**

**card.Enabled = false;**

**}**

**private void newHandButton\_Click\_1(object sender, EventArgs e)**

**{**

**deck = new Deck();**

**player = new BlackJackHand();**

**dealer = new BlackJackHand();**

**deck.Shuffle();**

**player.Add(deck.Deal());**

**dealer.Add(deck.Deal());**

**player.Add(deck.Deal());**

**dealer.Add(deck.Deal());**

**/\***

**LoadPlayerCard(0);**

**LoadDealerCard(0);**

**LoadPlayerCard(1);**

**LoadDealerCard(1);**

**\*/**

**LoadPlayerHand();**

**LoadDealerHand();**

**}**

**private void hitButton\_Click(object sender, EventArgs e)**

**{**

**player.Add(deck.Deal());**

**LoadPlayerHand();**

**}**

**private void standButton\_Click(object sender, EventArgs e)**

**{**

**dealer.Add(deck.Deal());**

**LoadDealerHand();**

**}**

**}**

**}**