

Peer Evaluation for Lab 4 – Chapter 12

Your name: (Your lab is the one being evaluated)	Will Schultz
Name(s) of peer evaluator(s)	Kaiser
Date:	5/23/2018

Instructions

You should have already completed Lab 4. 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.

<i>In Class Exercises – Customer Class – Exercise 12.1 (1 – 4)</i>	
<p>Completed Customer Class?</p> <ul style="list-style-type: none"> • Instance variables are camelCase and private? Yes • Implements all properties/methods in the specification? Implements both property get and set? Property/Method names are TitleCase? Yes • Properties/Methods are public? Yes • Completed Customer Tests? Tests all properties and methods in the class? Yes • Screen shot is included? Yes • Created a class diagram in visual studio? Screen shot included? The class diagram is not from VS • Source code includes Customer class as well as test program? yes 	
<p>One thing that you learned from writing and testing the class:</p> <p>I learned a lot from this lab. I learned how to create a class in C#, the different things that make up a class, and how to write a console app to test my classes.</p>	
<p>Something that you'd like to continue working on:</p> <p>I would like to continue to work on everything! There is a lot of information here and I don't 100% understand everything, so I would like to spend more time working with my own classes to better understand it all.</p>	
<p>Time you spent completing and testing the class:</p> <p>It took me maybe 20 minutes to do this lab.</p>	

Class diagram, screen shot of testing, source code for testing and class:

Customer Class		
Fields:	Properties:	Methods:
firstName	FirstName	GetDisplayText()
lastName	LastName	ToString
Email	Email	

```
C:\Users\wills\OneDrive\Desktop\School\Spring 2018\CS233N Intermediate C#\lab 4 and 5\CustomerMaintenanceStart\CustomerTests\bin\Debug\Cus...
Testing both constructors
Default Constructor. expecting default values.,,,
overload constructor. expecting Jim Jimmerson,Jimjimmerson@hotmail.com : Jim Jimmerson, Jimjimmerson@hotmail.com

testing getters
first name, expecting Fred : Fred
last name, expecting Fredderson : Fredderson
email, expecting FredFredderson@aol.com : FredFredderson@aol.com

Testing Setters
expecting Will Willerson @ WillWillerson@gmail.com : Will,Willerson,,WillWillerson@gmail.com

testing to string method
expecting Michael, Michaelerson, MicaelMichaelerson@gmail.com : Michael Michaelerson, MicaelMichaelerson@gmail.com
expecting Michael, Michaelerson, MicaelMichaelerson@gmail.com : CustomerMaintenanceClasses.Customer
```

```
namespace CustomerTests
{
    class Program
    {
        static void Main(string[] args)
        {
            TestCustomerConstructors();
            TestCustomerGetter();
            TestCustomerSetter();
            TestToString();

            Console.WriteLine();
            Console.ReadLine();
        }

        static void TestCustomerConstructors()
        {
            Customer c1 = new Customer();//default constructor
            Customer c2 = new Customer("Jim", "Jimmerson", "Jimjimmerson@hotmail.com");
            Console.WriteLine("Testing both constructors");
            Console.WriteLine("Default Constructor. expecting default values." +
c1.GetDisplayText(", "));
            Console.WriteLine("overload constructor. expecting Jim
Jimmerson,Jimjimmerson@hotmail.com : "+c2.GetDisplayText(" "));
            Console.WriteLine("");
        }
        static void TestCustomerGetter()
        {
            Customer c1 = new Customer("Fred", "Fredderson", "FredFredderson@aol.com");
            Console.WriteLine("testing getters");
            Console.WriteLine("first name, expecting Fred : " + c1.FirstName);
            Console.WriteLine("Last name, expecting Fredderson : " + c1.LastName);
            Console.WriteLine("email, expecting FredFredderson@aol.com : "+c1.Email);
            Console.WriteLine("");
        }
    }
}
```

```

    }
    static void TestCustomerSetter()
    {
        Customer c1 = new Customer("Matt", "Matterson", "MattMatterson@gmail.com");
        Console.WriteLine("Testing Setters");
        c1.FirstName = "Will";
        c1.LastName = "Willerson";
        c1.Email = "WillWillerson@gmail.com";
        Console.WriteLine("expecting Will Willerson @ WillWillerson@gmail.com : " +
c1.GetDisplayText(", "));
        Console.WriteLine("");

    }
    static void TestToString()
    {
        Customer c1 = new Customer("Michael", "Michaelerson",
"MicaelMichaelerson@gmail.com");
        Console.WriteLine("testing to string method");
        Console.WriteLine("expecting Michael, Michaelerson,
MicaelMichaelerson@gmail.com : " + c1.ToString());
        Console.WriteLine("expecting Michael, Michaelerson,
MicaelMichaelerson@gmail.com : " + c1);

    }

}
}

```

```

public class Customer
{
    private string firstName;
    private string lastName;
    private string email;

    public Customer() { }

    public Customer(string firstName,string lastName,string email)
    {
        this.FirstName = firstName;
        this.LastName = lastName;
        this.Email = email;
    }
    public string FirstName
    {
        get
        {
            return firstName;
        }
        set
        {
            firstName = value;
        }
    }
    public string LastName
    {
        get

```

```

    {
        return lastName;
    }
    set
    {
        lastName = value;
    }
}
public string Email
{
    //can just do get;set;
    get
    {
        return email;
    }
    set
    {
        email = value;
    }
}
public string GetDisplayText(string sep)
{
    return FirstName + sep + lastName + "," + sep + email;
}
public string ToString()
{
    return GetDisplayText("\t");
}

```

Card Problem	
Completed Card Class?	
<ul style="list-style-type: none"> Instance variables are camelCase and private? 	Yes
<ul style="list-style-type: none"> Implements all properties/methods in the specification? Implements both property get and set where appropriate? Property/Method names are TitleCase? Properties/Methods are public? 	Yes
<ul style="list-style-type: none"> Completed Card Tests? Tests all properties and methods in the class? Screen shot is included? 	Yes
<ul style="list-style-type: none"> Created a class diagram in visual studio? Screen shot included? 	Yes
<ul style="list-style-type: none"> Source code includes Card class as well as test program? 	yes
<p>One thing that you learned from writing and testing the class:</p> <p>I learned about creating a card class library and linking it to another form.</p>	

Something that you'd like to continue working on:

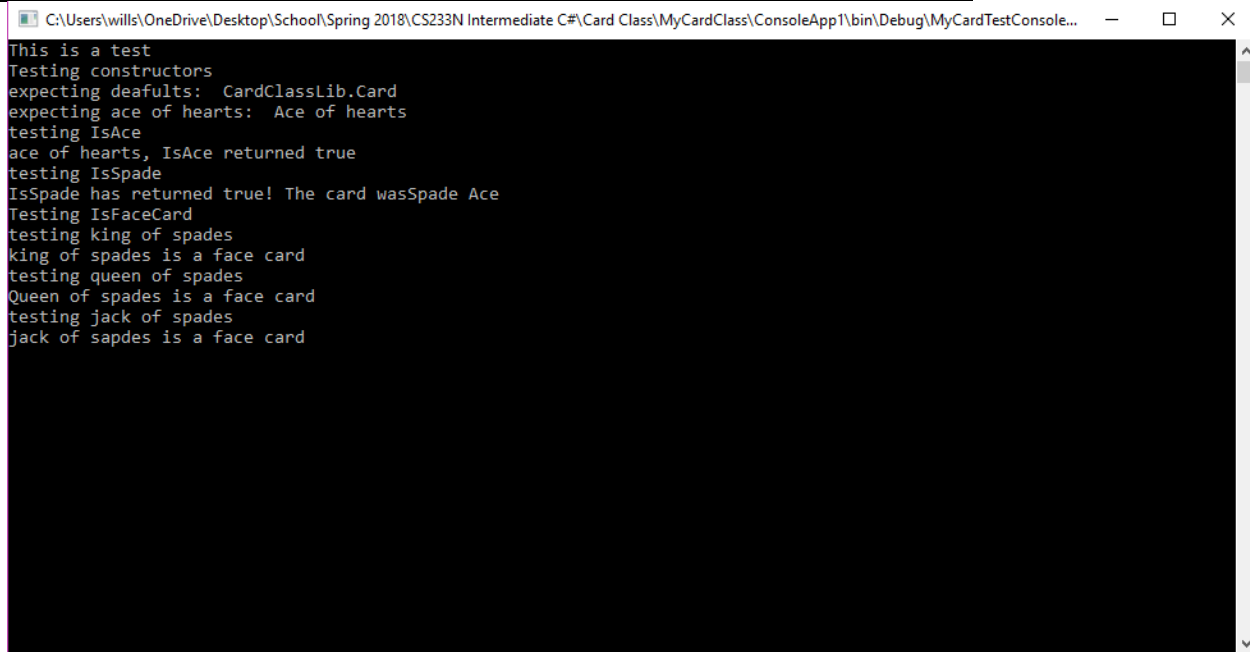
I had a lot of issues with setting up a blank form and linking it to my class library, that Kaiser did not have. I think I may have taken a wrong step in my setup and I would like to figure out how to do it correctly.

Time you spent completing and testing the class:

Because of the afore mentioned issues with my form I spent between 2 and 3 hours working on this.

Class diagram, screen shot of testing, source code for testing and class:

Card Class		
Fields:	Properties:	Methods:
suit	Suit	GetDisplayText()
cardValue	CardValue	IsAce
		IsSpade
		IsFaceCard



```
C:\Users\wills\OneDrive\Desktop\School\Spring 2018\CS233N Intermediate C#\Card Class\MyCardClass\ConsoleApp1\bin\Debug\MyCardTestConsole...
This is a test
Testing constructors
expecting deafults: CardClassLib.Card
expecting ace of hearts: Ace of hearts
testing IsAce
ace of hearts, IsAce returned true
testing IsSpade
IsSpade has returned true! The card wasSpade Ace
Testing IsFaceCard
testing king of spades
king of spades is a face card
testing queen of spades
Queen of spades is a face card
testing jack of spades
jack of sapdes is a face card
```

```
namespace CardClassLib
{
    public class Card
    {
        private string cardValue;
        private string suit;

        public Card() { }
```

```

public Card(string suit1, string cardValue1)
{
    cardValue = cardValue1;
    suit = suit1;
}
public string Suit
{
    get
    {
        return suit;
    }
    set
    {
        suit = value;
    }
}
public string CardValue
{
    get
    {
        return cardValue;
    }
    set
    {
        cardValue = value;
    }
}
public string GetDisplayText(string sep)
{
    return cardValue + sep + "of" + sep + suit;
}
public bool IsAce(string cardValue)
{
    if (cardValue == "Ace")
        return true;
    else
        return false;
}

public static bool IsSpade(string cardSuit)
{
    if (cardSuit == "Spade")
        return true;
    else
        return false;
}
public static bool IsFaceCard(string cardValue)
{
    if (cardValue == "King" || cardValue == "Queen" || cardValue == "Jack")
        return true;
    else
        return false;
}
}
}

```

```

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("This is a test");
        TestCardConstructors();
        TestIsAce();
        TestIsSpade();
        TestIsFaceCard();

        Console.ReadLine();
    }
    static void TestCardConstructors()
    {
        Card c1 = new Card();
        Card c = new Card("hearts", "Ace");
        Console.WriteLine("Testing constructors");
        Console.WriteLine("expecting deafults: " + c1);
        Console.WriteLine("expecting ace of hearts: " + c.GetDisplayText(" "));
    }

    static void TestIsAce()
    {
        Card c = new Card("hearts", "Ace");
        Card c2 = new Card("hearts", "King");
        Console.WriteLine("testing IsAce");
        string value = c.CardValue;

        if (c.IsAce(value) == true)
        {
            Console.WriteLine("ace of hearts, IsAce returned true");
        }
        else
            Console.WriteLine("something went wrong...");
    }
    static void TestIsSpade()
    {
        Card c = new Card("Spade", "Ace");
        Console.WriteLine("testing IsSpade");
        string suit = c.Suit;
        if (Card.IsSpade(c.Suit) == true)
            Console.WriteLine("IsSpade has returned true! The card was" + c.Suit + " "
+ c.CardValue);
        else
            Console.WriteLine("Something went wrong...");
    }
    static void TestIsFaceCard()
    {
        Card c1 = new Card("Spade", "King");
        Card c2 = new Card("Spade", "Queen");
        Card c3 = new Card("Spade", "Jack");

        Console.WriteLine("Testing IsFaceCard");
        Console.WriteLine("testing king of spades");
        if (Card.IsFaceCard(c1.CardValue) == true)

```

```

        Console.WriteLine("king of spades is a face card");
    else
        Console.WriteLine("Something went wrong...");

    Console.WriteLine("testing queen of spades");
    if (Card.IsFaceCard(c2.CardValue) == true)
        Console.WriteLine("Queen of spades is a face card");
    else
        Console.WriteLine("Something went wrong...");

    Console.WriteLine("testing jack of spades");

    if (Card.IsFaceCard(c3.CardValue) == true)
        Console.WriteLine("jack of spades is a face card");
    else
        Console.WriteLine("Something went wrong...");
}

}
}

```

<i>Programming style for all programs</i>			
Is proper indentation used?	Is each property/method indented properly?	Is each control structure indented properly?	Yes
Are comments used appropriately?			I don't think I added any comments this time
Do variable names use camel case? (camelCase for example)			Use
Do property/method names use Title Case (or Pascal Case?)			use

General comments and notes from the evaluator:

One thing that you learned from completing the evaluation:

I liked Kaisers use of just get; and set; as opposed to the longer way I wrote it. Also some of his code looked a lot my concise and practical.

Screen Shots and Source Code