CISP 41

Programming in C#

Project Evaluation Sheet

Student Name: _____ Johnny Cho and Bryant Tunbutr_____ Project Number: _____5_____

Project Name: _____BtunbutrProject1_____ Visual Studio Version:___2010_____

Date Due: _____12/6/2012_____ Date Turned In: _____12/6/2012____

Above to be completed by student

Points (80 Possible)

Correctness/Efficiency:

Output is accurate — It did crash on my machine as well — -3

Meets all requirements _____

Provide appropriate user interface _____

Logic is efficient _____

Documentation/Coding Style:

Project can be open from the submitted zip file _____

Folder is present and contains all necessary project files (no extra files) _____

Use required coding template _____

Use proper naming and spacing _____

Submit all requested information _____

Test Cases:

List all required test cases _____

Provide output forms for important test cases _____

Other issues: It seems Johnny carried a heavier workload. _____

Good effort, but too much work is being done in the presentation tier.

Extra Credit: It doesn't line up nicely, but interesting approach in popping up those text boxes. +2

Timeliness: _____

Project Score: 79/80

CISP 41 Programming in C#

Project specification

This software is intended to summarize, calculate, and display costs for repairs for the Auto Repair Shop.

It is designed in Visual Studio 2010 using the C# coding.

It uses user input including the selected part to be repaired and number of hours.

It displays total cost, summary, and provides information about the program with an about button.

Used in the project are arrays, methods, calculations, multiple forms including summary and about and splash.

The project also uses multiple forms, 3 tiers of forms, multiple classes, and data processing.

CISP 41 **Programming in C#**

Project status

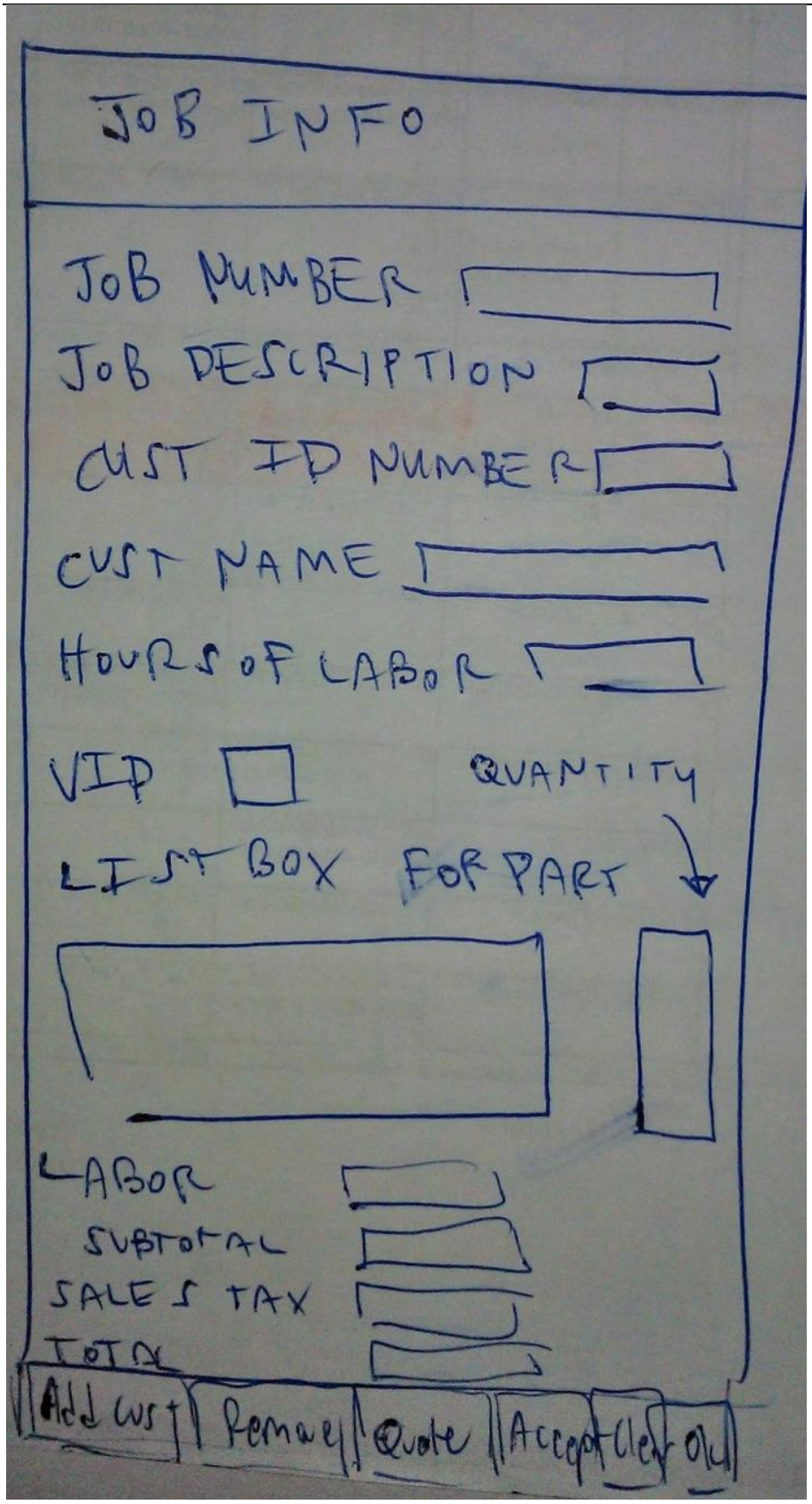
The project is completed and finalized

The vast majority of the project was created, developed and coded by Johnny Cho including design of forms, organization and use of data, arrays, and loops.

Bryant Tunbutr assisted in developing classes, calculations, proof reading, and developing documentation.

There was an attempt to do the extra credit "Save and retrieve customer data so new customer can be added and existing customers can be looked up by id" by the use of a database but that is incomplete.

Sketch of user interface



CISP 41 **Programming in C#**

Objects and Properties Plan for ____ _____Form

Object	Property	Setting
lblJobNum	Name Text	lblJobNum Job Number:
lblJobDescript	Name Text	lblJobDescript Job Description:
lblCustID	Name Text	lblCustID Customer ID:
lblCustName	Name Text	lblCustName Customer Name:
lblHoursLabor	Name Text	lblHoursLabor Hours of Labor:
listBox	Name SelectionMode	listBox MultiSimple
txtBoxParts0	Name	txtBoxParts0
txtBoxParts1	Name	txtBoxParts1
txtBoxParts2	Name	txtBoxParts2
txtBoxParts3	Name	txtBoxParts3
txtBoxParts4	Name	txtBoxParts4
lblLabor	Name Text	lblLabor Labor: $
lblSubtotal	Name Text	lblSubtotal Subtotal
lblSalesTax	Name Text	lblSalesTax Sales Tax
lblTotal	Name Text	lblTotal Total
btnQuote	Name Text	btnQuote &Quote
btnAccept	Name Text Enabled	btnAccept &Accept False
btnClear	Name Text	btnClear &Clear
btnOK	Name Text	btnOK &OK
checkBoxVIP	Name Text	checkBoxVIP VIP Customer
mnuPrintJobs	Name Text	Print Jobs
mnuExit	Name	mnuExit

	Text	**Exit**
mnuJobInfo	**Name** **Text**	**mnuJobInfo** **Job Information**
mnuSummary	**Name** **Text**	**mnuSummary** **Summary**
mnuAbout	**Name** **Text**	**mnuAbout** **About**
menuStrip1	**Name**	**menuStrip1**
printPreviewDialog1	**Name**	**printPreviewDialog1**
printDocument1	**Name**	**printDocument1**
btnExit	**Name** **Text**	**btnExit** **E&xit**
btnJobInformation	**Name** **Text**	**btnJobInformation** **&Job Information**
lblParts	**Name** **Text**	**lblParts** **Total Cost of Parts:**
lblLabor	**Name** **Text**	**lblLabor** **Total Cost of Labor:**
lblSalesTax	**Name** **Text**	**lblSalesTax** **Total Sales Tax:**
lblTotal	**Name** **Text**	**lblTotal** **Total:**
btnSummaryOk	**Name** **Text**	**btnSummaryOk** **OK**

CISP 41 **Programming in C#**

Event Plan for _____Form

Object	Event	Action - Pseudocode
btnQuote	Click	Declare the variables. try Convert the input Quantity to numeric. Checks if getSubtotalCost() is correct to display information in labels
btnClear	Click	Clear each text box except Summary fields. Set the focus in the first text box.
btnOK	Click	Exit the form. Uses clear() and sends information to array for updating.
getSubtotalCost()	General method	Computes quantity * price
loadListbox()	General method	Loads information from array to listbox
listBox	SelectedIndexChanged	Hides/unhides textboxes
btnAccept	click	Clears "everything" and saves information for printing
clear()	General method	General clearing textboxes- to be used in other methods
JobInfoForm	FormClosing	Uses clear() and sends information to array for updating.
mnuAbout	Click	Brings up the about form
mnuSummary	Click	Brings up the summary form
mnuExit	Click	This.close();
mnuPrintJobs	Click	Opens print preview

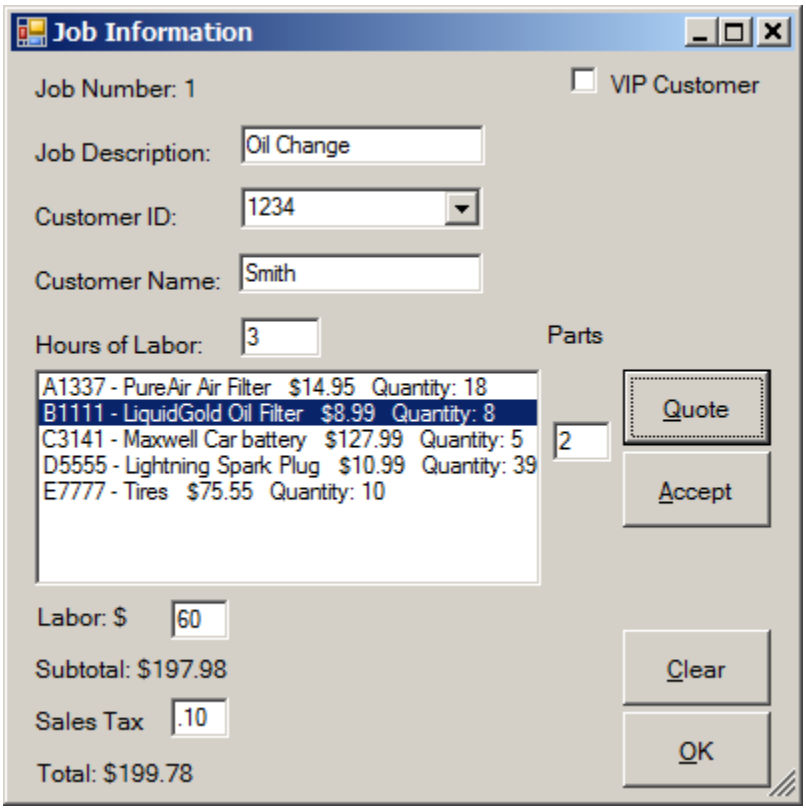
btnJobInformation	click	Brings up job information form
mnuJobInfo	click	Brings up job information form

CISP 41 Programming in C#

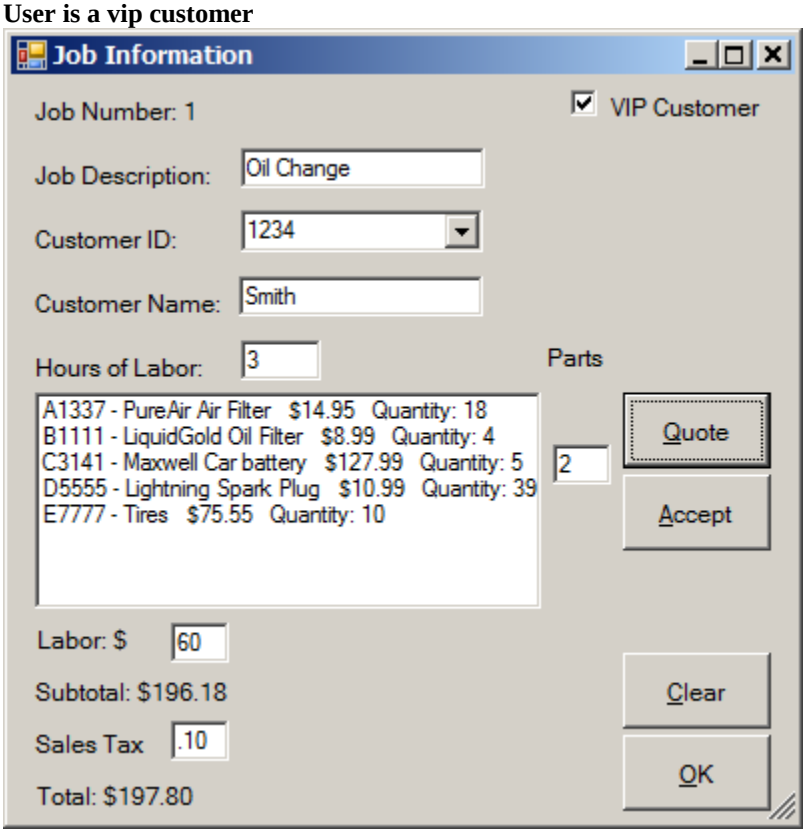
Test cases and captured screens

Test case #1

Hours = 3
Quantity = 2

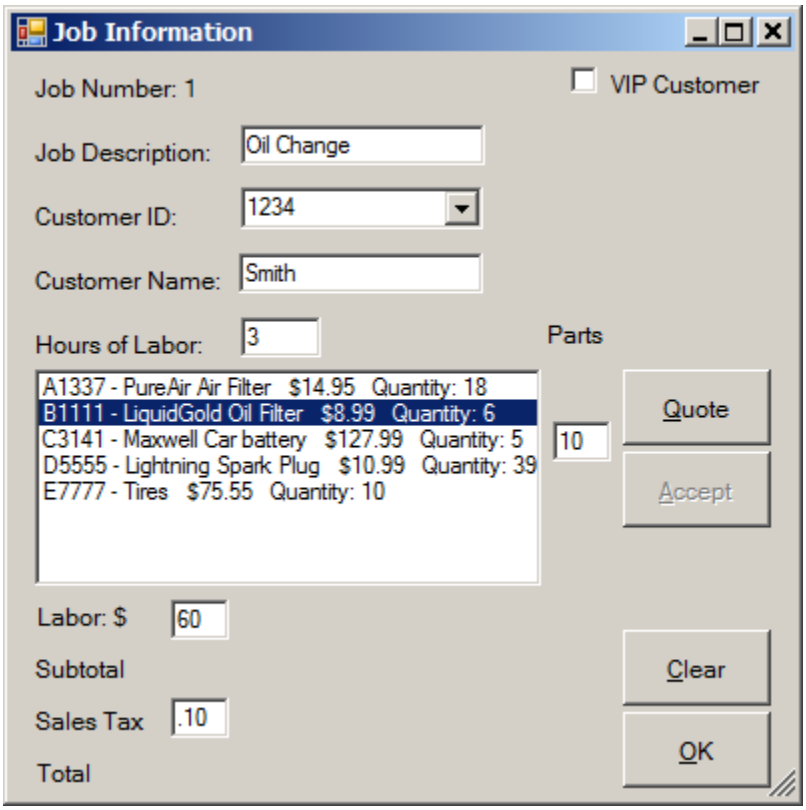


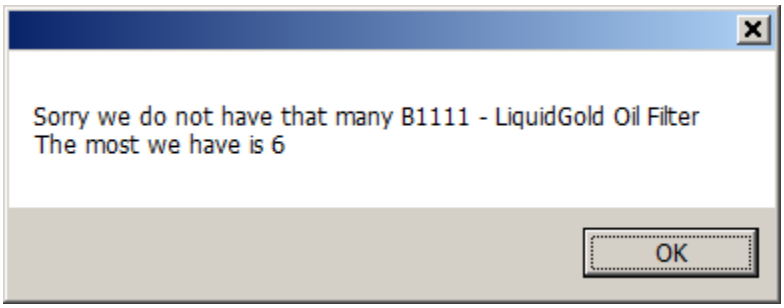
User is a vip customer



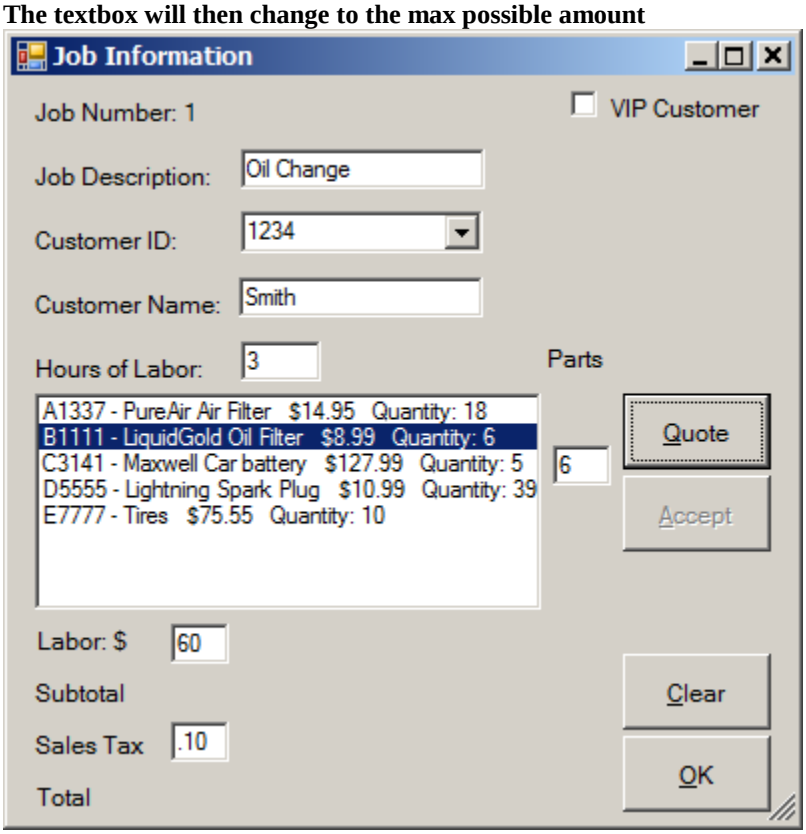
Test case #2

Customer wants more than there is available



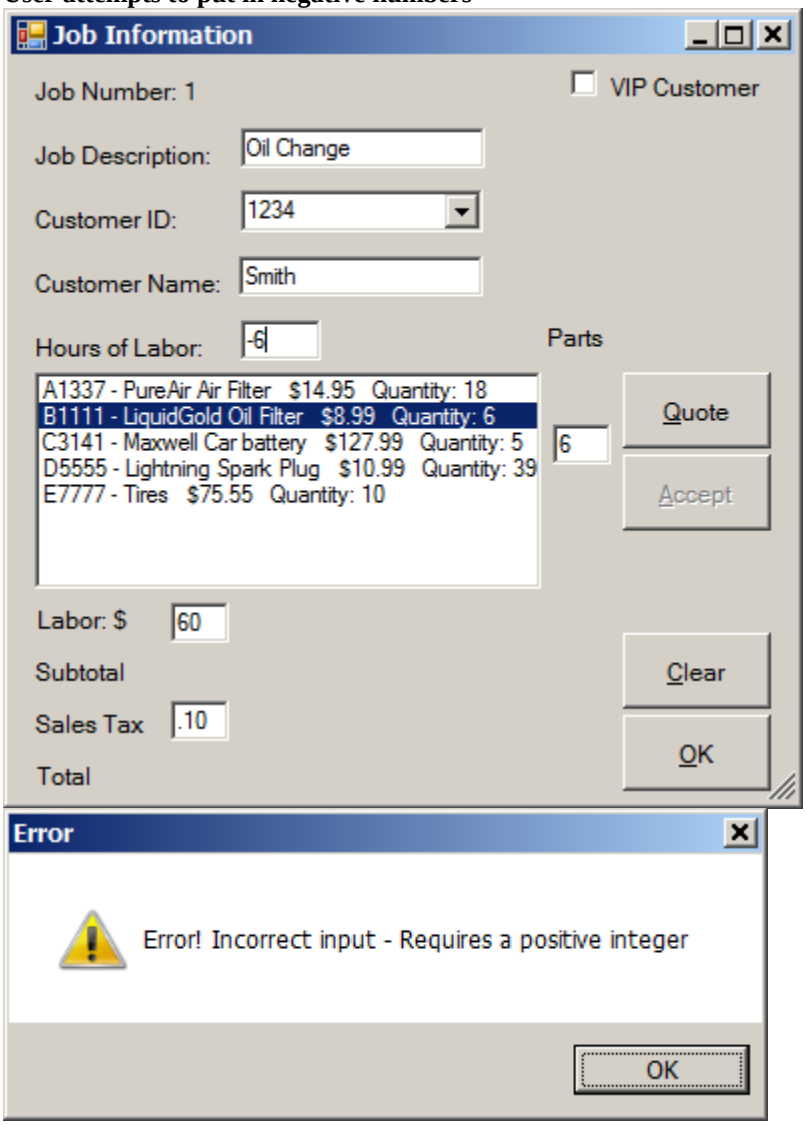


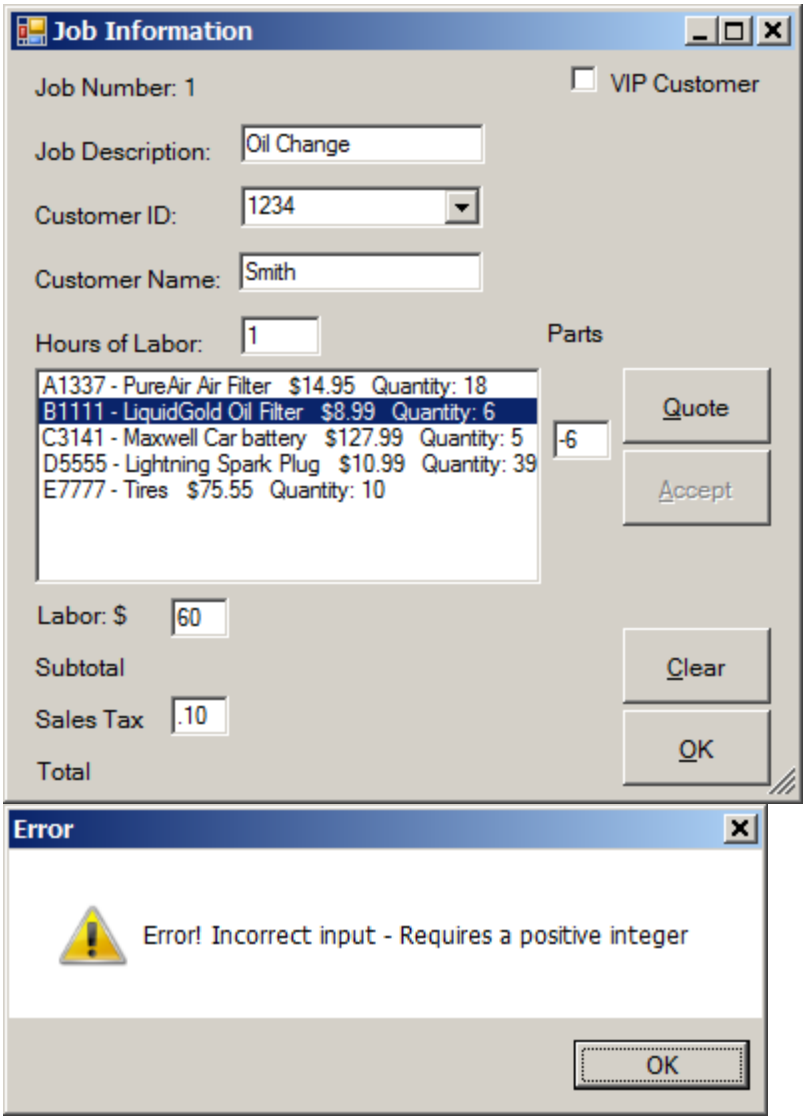
The textbox will then change to the max possible amount



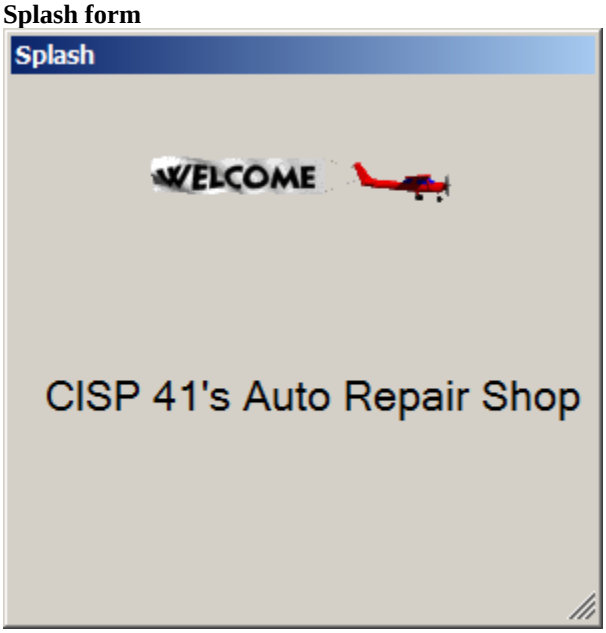
Test case #3

User attempts to put in negative numbers

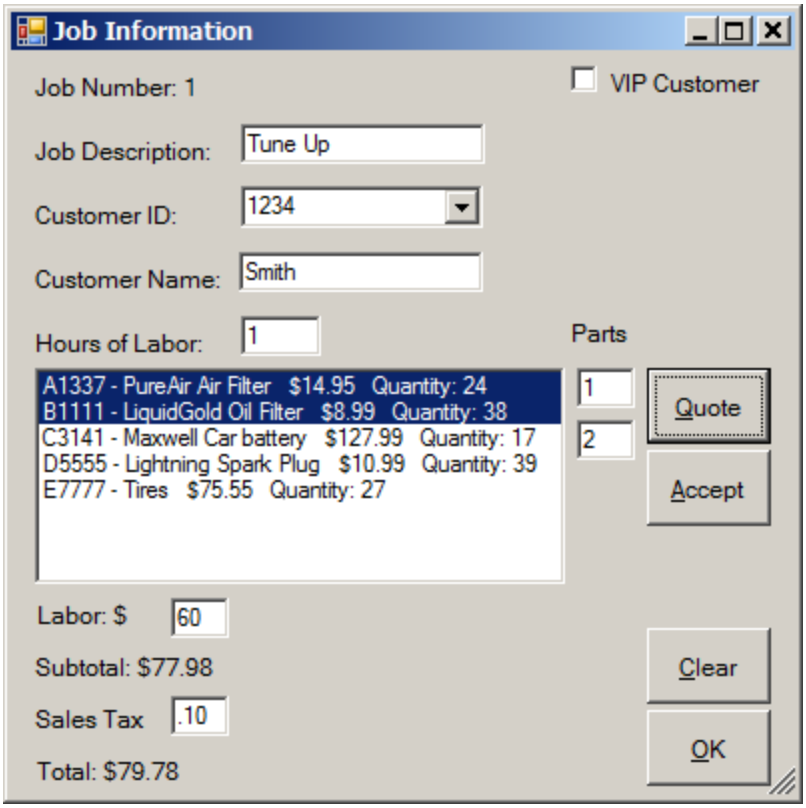




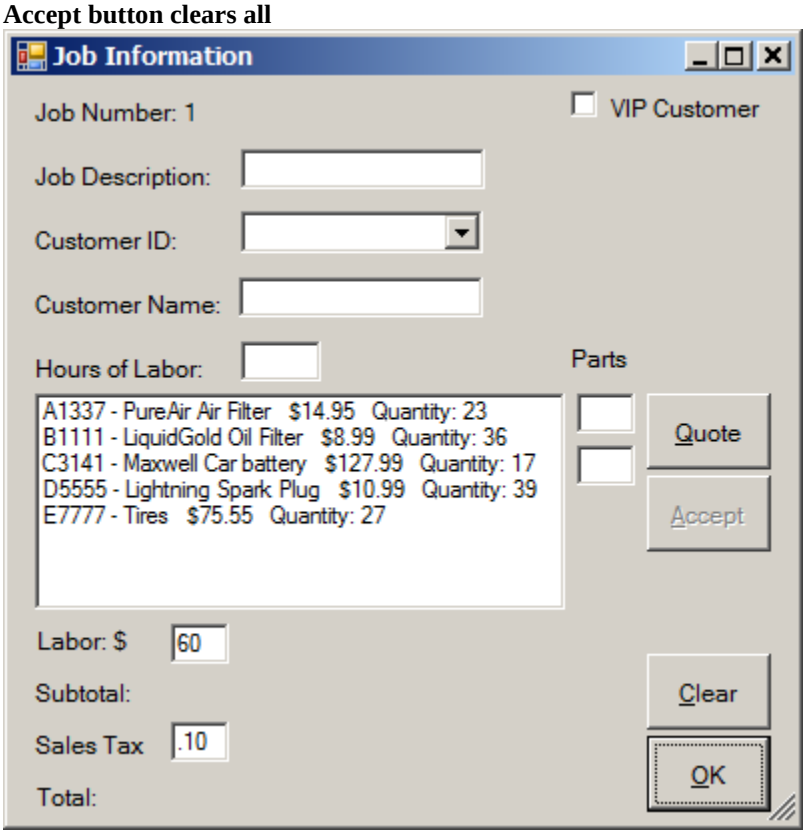
Splash form

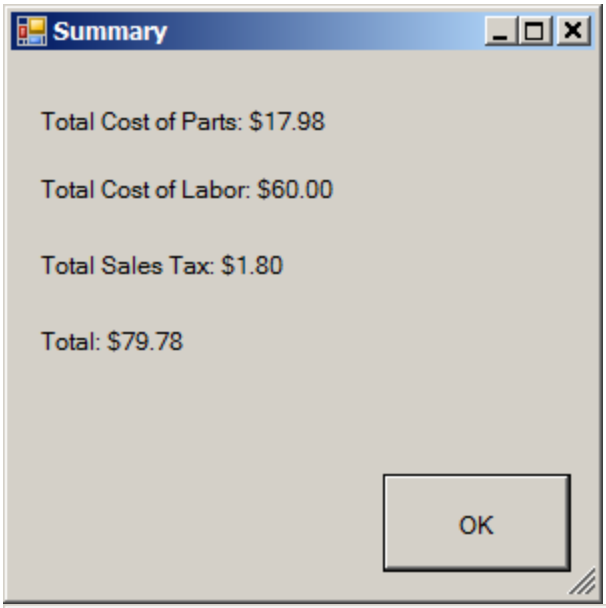


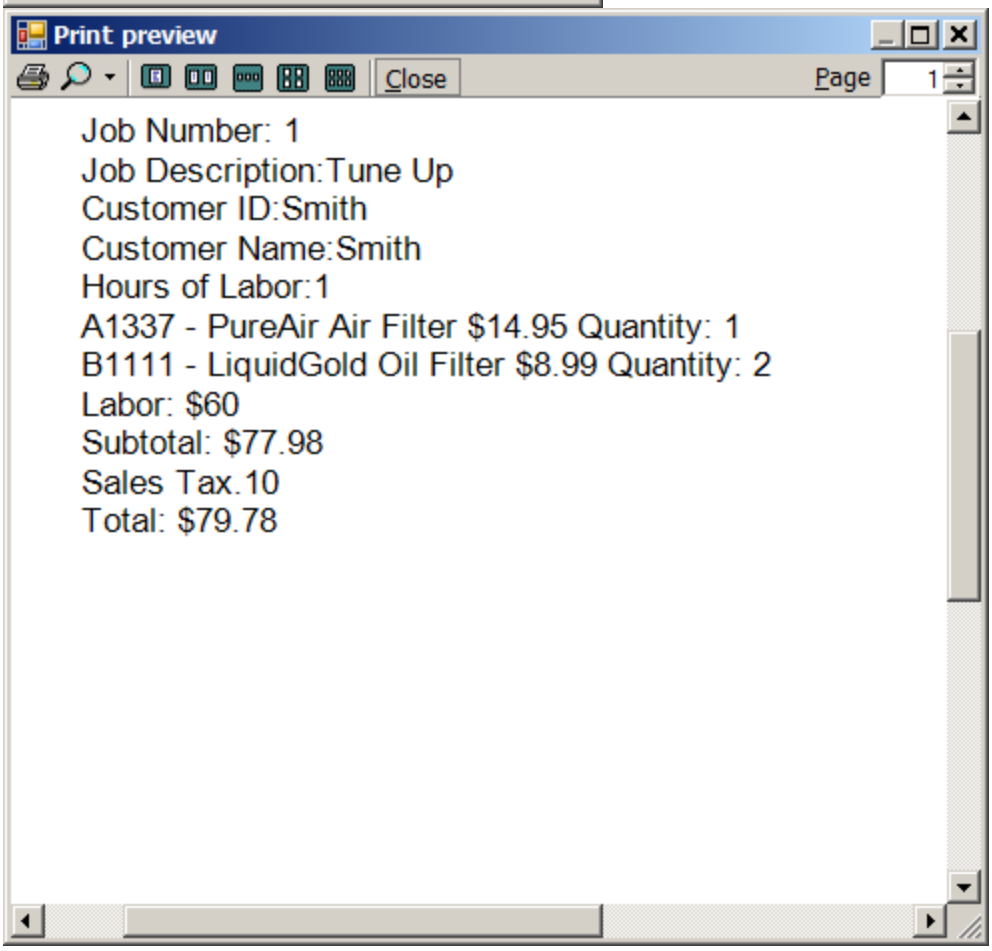
Printing and Summary



Accept button clears all




Summary
Total Cost of Parts: $17.98
Total Cost of Labor: $60.00
Total Sales Tax: $1.80
Total: $79.78
OK


Print preview
Close
Page
1
Job Number: 1
Job Description:Tune Up
Customer ID:Smith
Customer Name:Smith
Hours of Labor:1
A1337 - PureAir Air Filter $14.95 Quantity: 1
B1111 - LiquidGold Oil Filter $8.99 Quantity: 2
Labor: $60
Subtotal: $77.98
Sales Tax.10
Total: $79.78

Source code

Main form, Form1.cs

```
/*  Program:     Project 5
    Author:      Johnny Cho and Bryant Tunbutr
    Class:     Cisp 41
    Date:      12/2/2012
    Description:

    I certify that the code below is my own work.

   Exception(s): N/A
*/
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;


namespace Project5
{
    public partial class Form1 : Form
    {
        private StreamReader printReader;

        public Form1()
        {
            InitializeComponent();

        }
        JobInfoForm aJobInfoForm = new JobInfoForm();

        private void mnuJobInfo_Click(object sender, EventArgs e)
        {
            aJobInfoForm.ShowDialog();
        }
        private void btnJobInformation_Click(object sender, EventArgs e)
        {
            aJobInfoForm.ShowDialog();
        }
        private void mnuSummary_Click(object sender, EventArgs e)
        {
            SummaryForm aSummaryForm = new SummaryForm();
            aSummaryForm.ShowDialog();
        }
        private void mnuExit_Click(object sender, EventArgs e)
        {
            this.Close();
```

```
        }

        private void mnuPrintJobs_Click(object sender, EventArgs e)
        {
            printPreviewDialog1.Document = printDocument1;
            printPreviewDialog1.ShowDialog();

        }

        private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs
e)
        {
            Font printFont = new Font("Arial", 12);
            float lineHeightFloat = printFont.GetHeight();
            float hPrintLocation;
            float vPrintLocation;
            string printLine;

            hPrintLocation = e.MarginBounds.Left;
            vPrintLocation = e.MarginBounds.Top;

            printReader = new StreamReader("printOutput.txt");

            while (printReader.Peek() != -1)
            {
                printLine = printReader.ReadLine();

                e.Graphics.DrawString(printLine, printFont, Brushes.Black,
                    hPrintLocation, vPrintLocation);

                vPrintLocation += lineHeightFloat;
            }
            printReader.Close();

        }
        private void mnuAbout_Click(object sender, EventArgs e)
        {
            AboutBox1 aAboutBox = new AboutBox1();
            aAboutBox.ShowDialog();
        }
        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

User form, Form2.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace Project5
```

```
{
    public partial class JobInfoForm : Form
    {
        private StreamReader partsReader;
        private StreamWriter partsWriter;
        private StreamWriter printWriter;
        //private StreamReader custReader;
        //private StreamWriter custWriter;

        //declare 2d array to put stuff from txtfile
         string[,] partArray = new string[5, 3];
        //declare an array of textboxes
        public static TextBox[] txtBoxArray;

        //declare class level variables
        int jobNum = 1;
        decimal salesTaxDec;
        decimal laborDec;
        int hoursLaborInt;

        int quantityInteger;
        decimal subCostDec;

        getSubTotalClass SubtotalInheritance;

        //Class level variables for the summary form
        public static decimal accuParts;
        public static decimal accuLabor;
        public static decimal accuSalesTax;
        public static decimal accuTotal;

        //Below is the constructor
        public JobInfoForm()
        {
            InitializeComponent();

            lblJobNum.Text = "Job Number: " + jobNum.ToString();

            try
            {
                partsReader = new StreamReader("PartsInputOutput.txt");
            }
            catch
            {
                MessageBox.Show("Error! File not found");
            }

            int boundi = partArray.GetUpperBound(0);
            int boundj = partArray.GetUpperBound(1);
            // declare textboxes inside the textbox array
            txtBoxArray = new TextBox[5] { txtBoxParts0, txtBoxParts1, txtBoxParts2, txtBoxParts3,
txtBoxParts4 };

            //Read txtFile and place into array
            for (int i = 0; i <= boundi; i++)
            {
                for (int j = 0; j <= boundj; j++)
                {
                    partArray[i, j] = partsReader.ReadLine();
                }
            }
        }
        private void loadListbox()
        {
```

```
            //adds strings from array to the listbox
            listBox.Items.Clear(); //clears the whole textbox so it will not keep adding items when
opening/closing
            listBox.Items.Add(partArray[0, 0] + "   " + "$" + partArray[0, 1] + "   " + "Quantity: " +
partArray[0, 2] );
            listBox.Items.Add(partArray[1, 0] + "   " + "$" + partArray[1, 1] + "   " + "Quantity: " +
partArray[1, 2]);
            listBox.Items.Add(partArray[2, 0] + "   " + "$" + partArray[2, 1] + "   " + "Quantity: " +
partArray[2, 2]);
            listBox.Items.Add(partArray[3, 0] + "   " + "$" + partArray[3, 1] + "   " + "Quantity: " +
partArray[3, 2]);
            listBox.Items.Add(partArray[4, 0] + "   " + "$" + partArray[4, 1] + "   " + "Quantity: " +
partArray[4, 2]);

        }
        private void JobInfoForm_Load(object sender, EventArgs e)
        {
            loadListbox();
        }
        decimal sumDec;
        private decimal getSubtotalCost()
        {
            foreach (int i in listBox.SelectedIndices)
            {
                if (int.Parse(txtBoxArray[i].Text) < 0)
                {
                    MessageBox.Show("Error! Incorrect input - Requires a positive integer", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    txtBoxArray[i].Focus();
                    return -1;
                }
                if(int.Parse(txtBoxArray[i].Text) > decimal.Parse(partArray[i, 2]))
                {
                    MessageBox.Show("Sorry we do not have that many " + partArray[i, 0].ToString() +
"\nThe most we have is " + partArray[i, 2].ToString());
                    return -1;
                }
                if ((int.Parse(txtBoxArray[i].Text) <= decimal.Parse(partArray[i, 2]) ) &&
(int.Parse(txtBoxArray[i].Text) >= 0))
                {
                    //Enable Accept Button
                    btnAccept.Enabled = true;

                    partArray[i, 2] = (decimal.Parse(partArray[i, 2]) -
int.Parse(txtBoxArray[i].Text)).ToString();

                    //decimal cost = int.Parse(txtBoxArray[i].Text) * decimal.Parse(partArray[i, 1]);
                    //sumDec = sumDec + cost;

                    quantityInteger = int.Parse(txtBoxArray[i].Text);
                    subCostDec = decimal.Parse(partArray[i, 1]);
                    if (checkBoxVIP.Checked == false)
                    {
                        SubtotalInheritance = new getSubTotalClass(quantityInteger, subCostDec, sumDec);
                    }
                    if (checkBoxVIP.Checked == true)
                    {
                        SubtotalInheritance = new vipClass(quantityInteger, subCostDec, sumDec);
                    }

                    sumDec = SubtotalInheritance.ComputeCost;

                }
            }
```

```
            return sumDec;
        }
        private void btnQuote_Click(object sender, EventArgs e)
        {
            try
            {
                salesTaxDec = decimal.Parse(txtBoxSalesTax.Text);
                laborDec = decimal.Parse(txtBoxLabor.Text);
                hoursLaborInt = int.Parse(tboxHourOfLabor.Text);

                if (hoursLaborInt <= 0)
                {
                    MessageBox.Show("Error! Incorrect input - Requires a positive integer", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    tboxHourOfLabor.Focus();
                    return;
                }
                computeLaborCostClass acomputeLaborCost =
            new computeLaborCostClass(hoursLaborInt, laborDec);

                decimal computeLaborCost = acomputeLaborCost.ComputeLaborCost;

                if (getSubtotalCost() == -1)
                {
                    //stop computation if user wants too big of a quantity
                    foreach (int i in listBox.SelectedIndices)
                    {
                        if (int.Parse(txtBoxArray[i].Text) > decimal.Parse(partArray[i, 2]))
                        {
                            txtBoxArray[i].Text = partArray[i, 2];
                        }
                    }
                    return;
                }
                //increases job number
                jobNum = jobNum + 1;
                //Display labels
                lblSubtotal.Text = "Subtotal: " + (sumDec + computeLaborCost).ToString("C");
                lblTotal.Text = "Total: " + ((sumDec * salesTaxDec) + sumDec +
computeLaborCost).ToString("C");
                //accumulate
                accuParts = accuParts + sumDec;
                accuLabor = accuLabor + computeLaborCost;
                accuSalesTax = accuSalesTax + (sumDec * salesTaxDec);
                accuTotal = accuTotal + ((sumDec * salesTaxDec) + sumDec + computeLaborCost);

            }
            catch
            {
                tboxHourOfLabor.Focus();
                MessageBox.Show("Error! Incorrect input", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
            }
        }
        //Make the textboxes visible / invisible
        private void listBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            bool[] visibleArray;
            visibleArray = new bool[5] { false, false, false, false, false };

            if (listBox.SelectedItems.Count > 0)
            {
                for (int j = 0; j < listBox.Items.Count; j++)
                {
```

```
                if (visibleArray[j] == false)
                {
                    txtBoxArray[j].Visible = false;
                }
            }
            for (int i = 0; i < listBox.SelectedItems.Count; i++)
            {
                visibleArray[(listBox.SelectedIndices[i])] = true;
                //txtBoxArray[(listBox.SelectedIndices[i])].Clear();
                txtBoxArray[(listBox.SelectedIndices[i])].Visible = true;
                //MessageBox.Show(listBox.SelectedIndices[i].ToString());
            }
        }
    }
    private void btnAccept_Click(object sender, EventArgs e)
    {
        printWriter = new StreamWriter("printOutput.txt", true);

        printWriter.WriteLine(lblJobNum.Text);
        printWriter.WriteLine(lblJobDescript.Text + tBoxJobDescript.Text);
        printWriter.WriteLine(lblCustID.Text + tBoxCustName.Text);
        printWriter.WriteLine(lblCustName.Text + tBoxCustName.Text);
        printWriter.WriteLine(lblHoursLabor.Text + tboxHourOfLabor.Text);

        foreach (int i in listBox.SelectedIndices)
        {
            printWriter.WriteLine(partArray[i, 0] + " $" + partArray[i, 1] + " Quantity: " +
(txtBoxArray[i].Text));
        }
        printWriter.WriteLine(lblLabor.Text + txtBoxLabor.Text);
        printWriter.WriteLine(lblSubtotal.Text);
        printWriter.WriteLine(lblSalesTax.Text + txtBoxSalesTax.Text);
        printWriter.WriteLine(lblTotal.Text);
        printWriter.WriteLine();



        loadListbox();
        clear();
        printWriter.Close();
        btnAccept.Enabled = false;
    }
    private void clear()
    {
        tBoxJobDescript.Focus();
        tBoxJobDescript.Clear();
        CBoxCustID.Text = "";
        tBoxCustName.Clear();
        tboxHourOfLabor.Clear();
        txtBoxParts0.Clear();
        txtBoxParts1.Clear();
        txtBoxParts2.Clear();
        txtBoxParts3.Clear();
        txtBoxParts4.Clear();
        lblJobNum.Text = "Job Number:" + jobNum.ToString();
        lblSubtotal.Text = "Subtotal: ";
        lblTotal.Text = "Total: ";
        sumDec = 0;

    }
    private void btnClear_Click(object sender, EventArgs e)
    {
        clear();
        lblJobNum.Text = "Job Number: " + jobNum.ToString();
```

```
        }
        private void OkCloseForm()
        {
            partsReader.Close();
            //File.Delete("PartsInputOutput.txt");
            partsWriter = new StreamWriter("PartsInputOutput.txt");

            //updates file upon closing
            int boundi = partArray.GetUpperBound(0);
            int boundj = partArray.GetUpperBound(1);
            for (int i = 0; i <= boundi; i++)
            {
                for (int j = 0; j <= boundj; j++)
                {
                    partsWriter.WriteLine(partArray[i, j]);
                }
            }
            partsWriter.Close();

            this.Close();
        }
        private void btnOK_Click(object sender, EventArgs e)
        {
            OkCloseForm();
        }

        private void JobInfoForm_FormClosing(object sender, FormClosingEventArgs e)
        {
            OkCloseForm();
        }
        /*
        private void btnAddCust_Click(object sender, EventArgs e)
        {
            if (CBoxCustID.Text != "" && tBoxCustName.Text != "")
            {
                CBoxCustID.Items.Add(CBoxCustID.Text);
            }
            else
            {
                MessageBox.Show("Please Enter an ID and name.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
            }
        }
        private void btnRemoveCust_Click(object sender, EventArgs e)
        {
            if (CBoxCustID.Text != "" )
            {
                CBoxCustID.Items.Remove(CBoxCustID.Text);
            }
            else
            {
                MessageBox.Show("Please select a Customer ID", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
            }
        }*/
    }
}
```

getSubTotalClass.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Project5
{
    class getSubTotalClass
    {

        protected int quantityInteger;
        protected decimal computeCostDec, subCostDec, sumDec;

        public int quantity
        {
            get
            {
                return quantityInteger;
            }
            private set
            {
                quantityInteger = value;
            }
        }

        public decimal sum
        {
            get
            {
                return sumDec;
            }
            private set
            {
                sumDec = value;
            }
        }

        public decimal subCost
        {
            get
            {
                return subCostDec;
            }
            private set
            {
                subCostDec = value;
            }
        }

        public decimal ComputeCost
        {
            get
            {
                return computeCostDec;
```

```
            }
            private set
            {
                computeCostDec = value;
            }
        }

        public getSubTotalClass(int quantityInteger,
            decimal subCostDec, decimal sumDec)
        {
            // Constructor.
            quantity = quantityInteger;
            subCost = subCostDec;
            sum = subCostDec;
            computeCost();
        }

        protected virtual void computeCost()
        {
            computeCostDec = subCostDec * quantityInteger;
        }

    }
}
```

vipClass.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Project5
{
    class vipClass : getSubTotalClass
    {
        const decimal vipDecimal = .9M;

        public vipClass(int quantityInteger, decimal subCostDec, decimal sumDec)
            : base (quantityInteger, subCostDec, sumDec)
        {
            // Call the base class constructor and pass arguments.
        }


        //Method in the derived class that overrides the method in the base class
        protected override void computeCost()
        {
            // Find the ExtendedPrice.
            computeCostDec = subCostDec * quantityInteger * vipDecimal;
        }

    }
}
```

computeLaborCostClass.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Project5
{
    class computeLaborCostClass
    {

        protected int hoursLaborInt;
        protected decimal laborDec, computeLaborCostDec;

        public int hoursLabor
        {
            get
            {
                return hoursLaborInt;
            }
            set
            {
                hoursLaborInt = value;
            }
        }

        public decimal labor
        {
            get
            {
                return laborDec;
            }
            set
            {
                laborDec = value;
            }
        }

        public decimal ComputeLaborCost
        {
            get
            {
                return computeLaborCostDec;
            }
        }

        public computeLaborCostClass(int hoursLaborInt,
            decimal laborDec)
        {
            // Constructor.
            hoursLabor = hoursLaborInt;
            labor = laborDec;
```

```
            computeLaborCost();
        }

        protected virtual void computeLaborCost()
        {
            computeLaborCostDec = laborDec * hoursLaborInt;
        }

    }
}
```

Splash.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Project5
{
    public partial class Splash : Form
    {
        public Splash()
        {
            InitializeComponent();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Summary.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Project5
{
    public partial class SummaryForm : Form
    {
        public SummaryForm()
        {
            InitializeComponent();
            lblParts.Text = "Total Cost of Parts: " + JobInfoForm.accuParts.ToString("C");
            lblLabor.Text = "Total Cost of Labor: " + JobInfoForm.accuLabor.ToString("C");
            lblSalesTax.Text = "Total Sales Tax: " + JobInfoForm.accuSalesTax.ToString("C");
            lblTotal.Text = "Total: " + JobInfoForm.accuTotal.ToString("C");
        }

        private void btnSummaryOk_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void SummaryForm_Load(object sender, EventArgs e)
        {

        }
    }
}
```