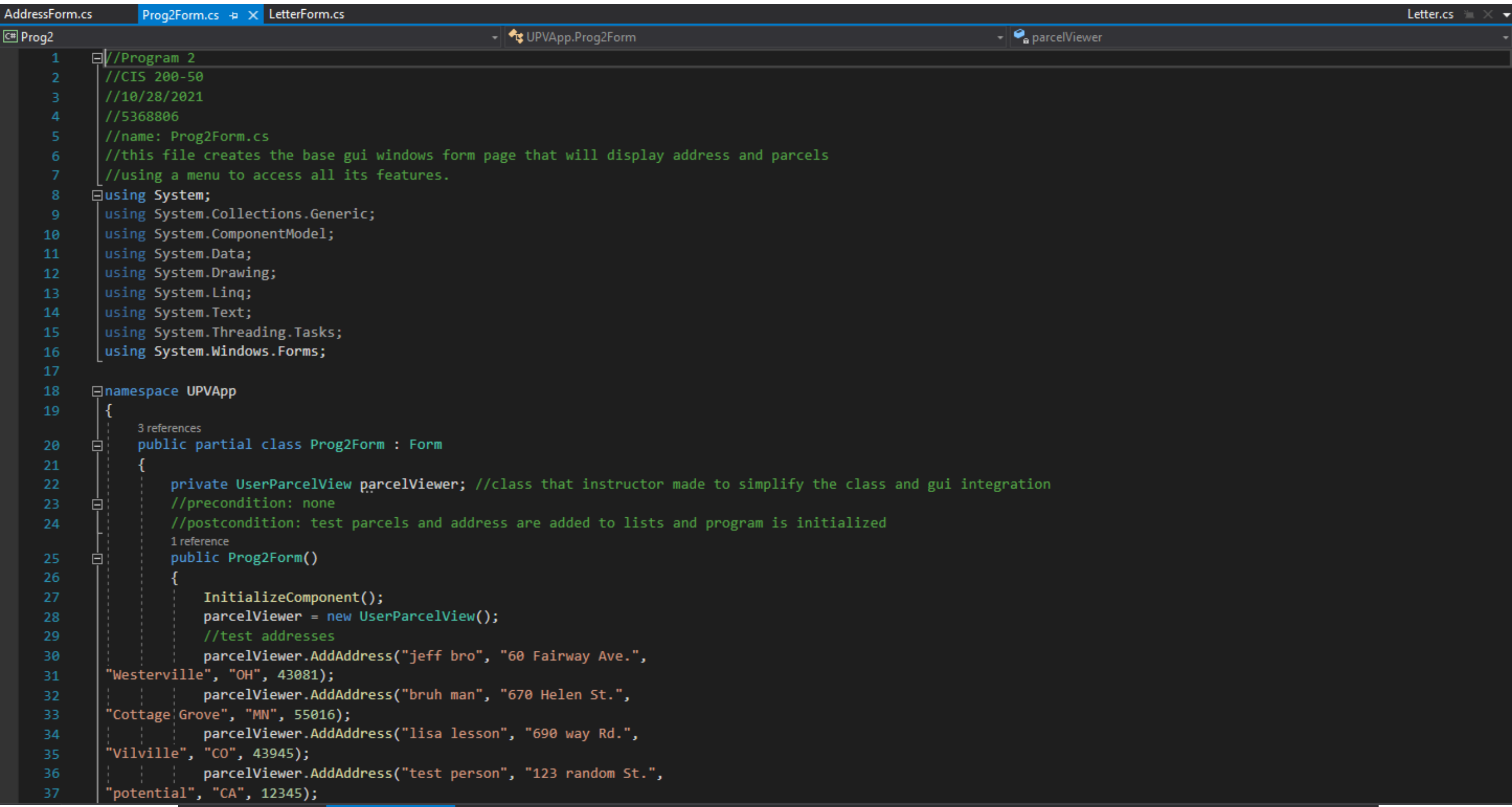
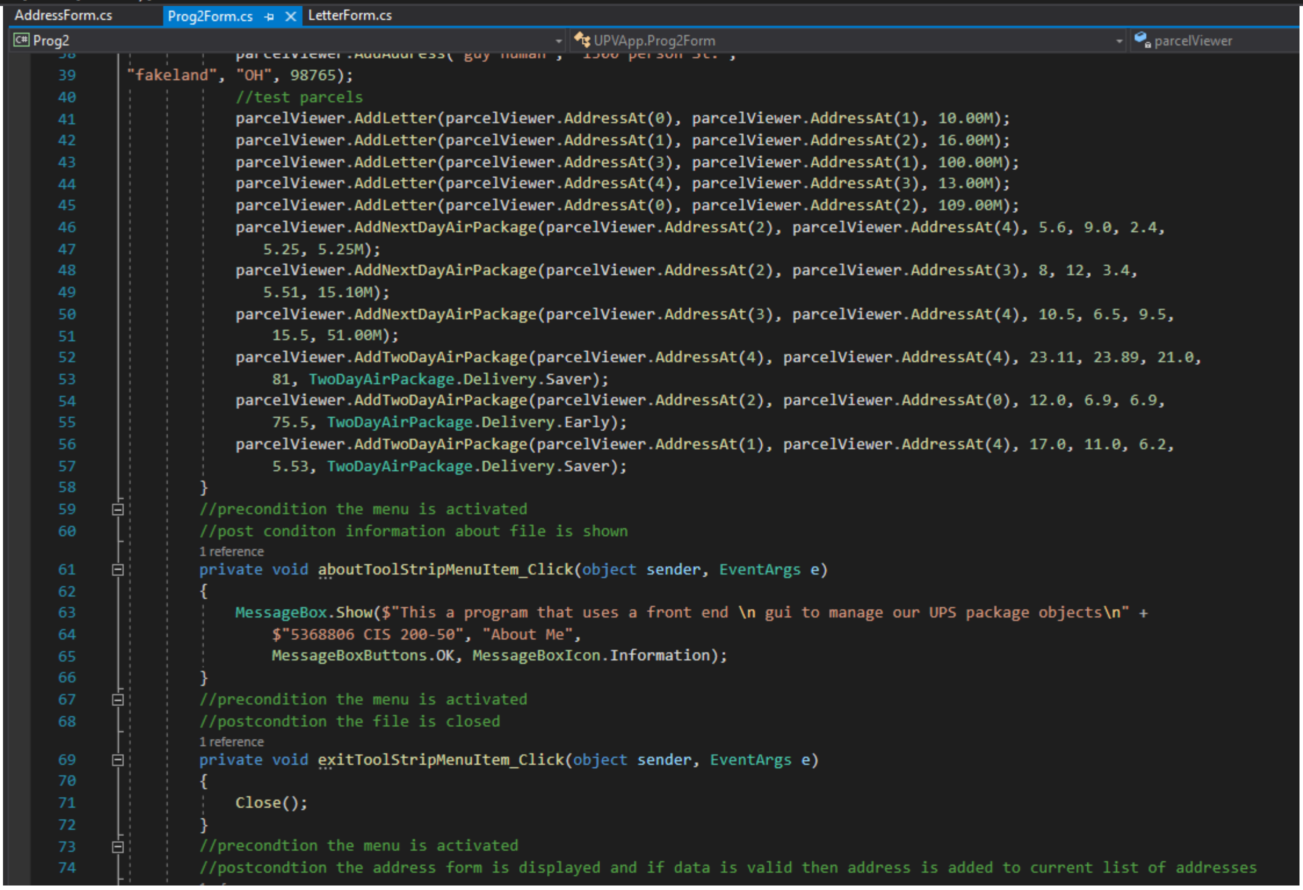
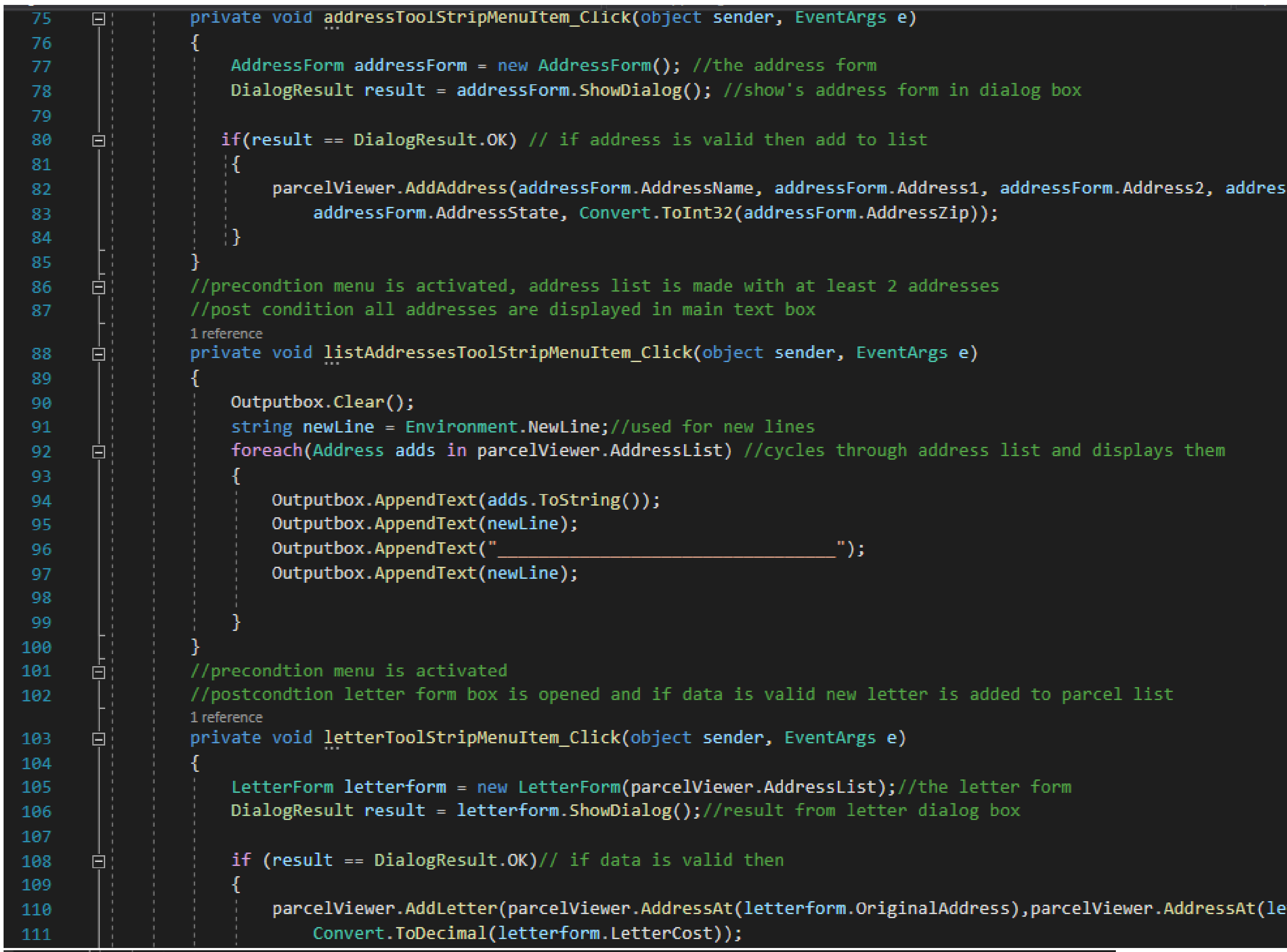


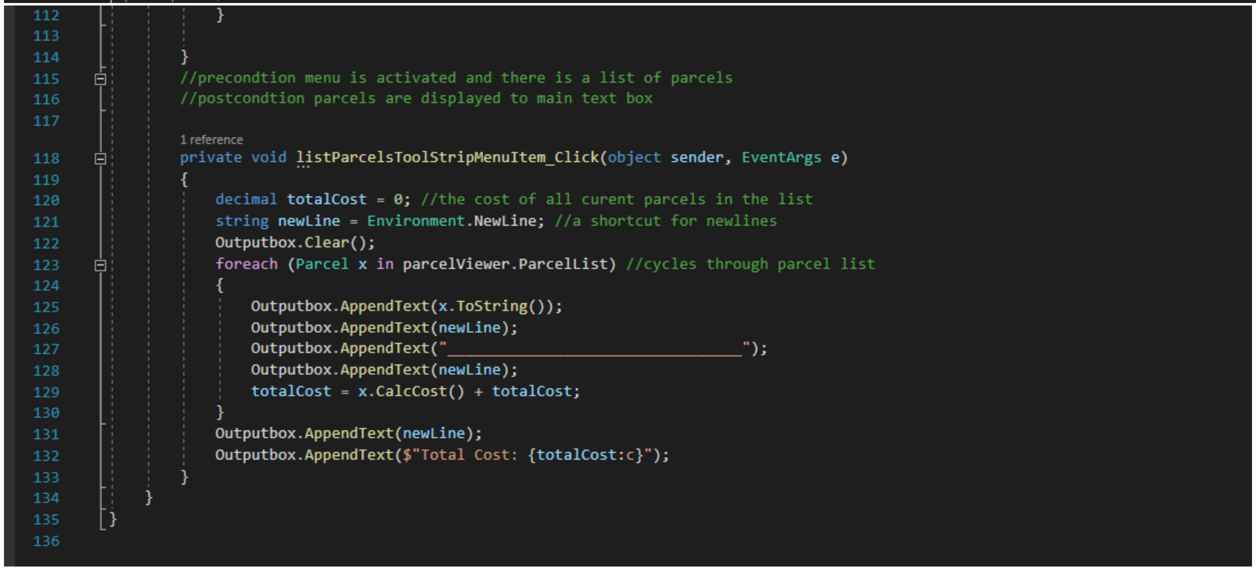
```
//Program 2
//CIS 200-50
//10/28/2021
//5368806
//name: Prog2Form.cs
//this file creates the base gui windows form page that will display address and parcels
//using a menu to access all its features.
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace UPVApp
{
    3 references
    public partial class Prog2Form : Form
    {
        private UserParcelView parcelViewer; //class that instructor made to simplify the class and gui integration
        //precondition: none
        //postcondition: test parcels and address are added to lists and program is initialized
        1 reference
        public Prog2Form()
        {
            InitializeComponent();
            parcelViewer = new UserParcelView();
            //test addresses
            parcelViewer.AddAddress("jeff bro", "60 Fairway Ave.",
"Westerville", "OH", 43081);
            parcelViewer.AddAddress("bruh man", "670 Helen St.",
"Cottage Grove", "MN", 55016);
            parcelViewer.AddAddress("lisa lesson", "690 way Rd.",
"Vilville", "CO", 43945);
            parcelViewer.AddAddress("test person", "123 random St.",
"potential", "CA", 12345);
```



```
            parcelViewer.AddAddress("guy human", "1500 person St.",
"fakeland", "OH", 98765);
            //test parcels
            parcelViewer.AddLetter(parcelViewer.AddressAt(0), parcelViewer.AddressAt(1), 10.00M);
            parcelViewer.AddLetter(parcelViewer.AddressAt(1), parcelViewer.AddressAt(2), 16.00M);
            parcelViewer.AddLetter(parcelViewer.AddressAt(3), parcelViewer.AddressAt(1), 100.00M);
            parcelViewer.AddLetter(parcelViewer.AddressAt(4), parcelViewer.AddressAt(3), 13.00M);
            parcelViewer.AddLetter(parcelViewer.AddressAt(0), parcelViewer.AddressAt(2), 109.00M);
            parcelViewer.AddNextDayAirPackage(parcelViewer.AddressAt(2), parcelViewer.AddressAt(4), 5.6, 9.0, 2.4,
               5.25, 5.25M);
            parcelViewer.AddNextDayAirPackage(parcelViewer.AddressAt(2), parcelViewer.AddressAt(3), 8, 12, 3.4,
               5.51, 15.10M);
            parcelViewer.AddNextDayAirPackage(parcelViewer.AddressAt(3), parcelViewer.AddressAt(4), 10.5, 6.5, 9.5,
                15.5, 51.00M);
            parcelViewer.AddTwoDayAirPackage(parcelViewer.AddressAt(4), parcelViewer.AddressAt(4), 23.11, 23.89, 21.0,
                81, TwoDayAirPackage.Delivery.Saver);
            parcelViewer.AddTwoDayAirPackage(parcelViewer.AddressAt(2), parcelViewer.AddressAt(0), 12.0, 6.9, 6.9,
                75.5, TwoDayAirPackage.Delivery.Early);
            parcelViewer.AddTwoDayAirPackage(parcelViewer.AddressAt(1), parcelViewer.AddressAt(4), 17.0, 11.0, 6.2,
                5.53, TwoDayAirPackage.Delivery.Saver);
        }
        //precondition the menu is activated
        //post conditon information about file is shown
        1 reference
        private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show($"This a program that uses a front end \n gui to manage our UPS package objects\n" +
                $"5368806 CIS 200-50", "About Me",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        //precondition the menu is activated
        //postcondtion the file is closed
        1 reference
        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }
        //precondtion the menu is activated
        //postcondtion the address form is displayed and if data is valid then address is added to current list of addresses
```

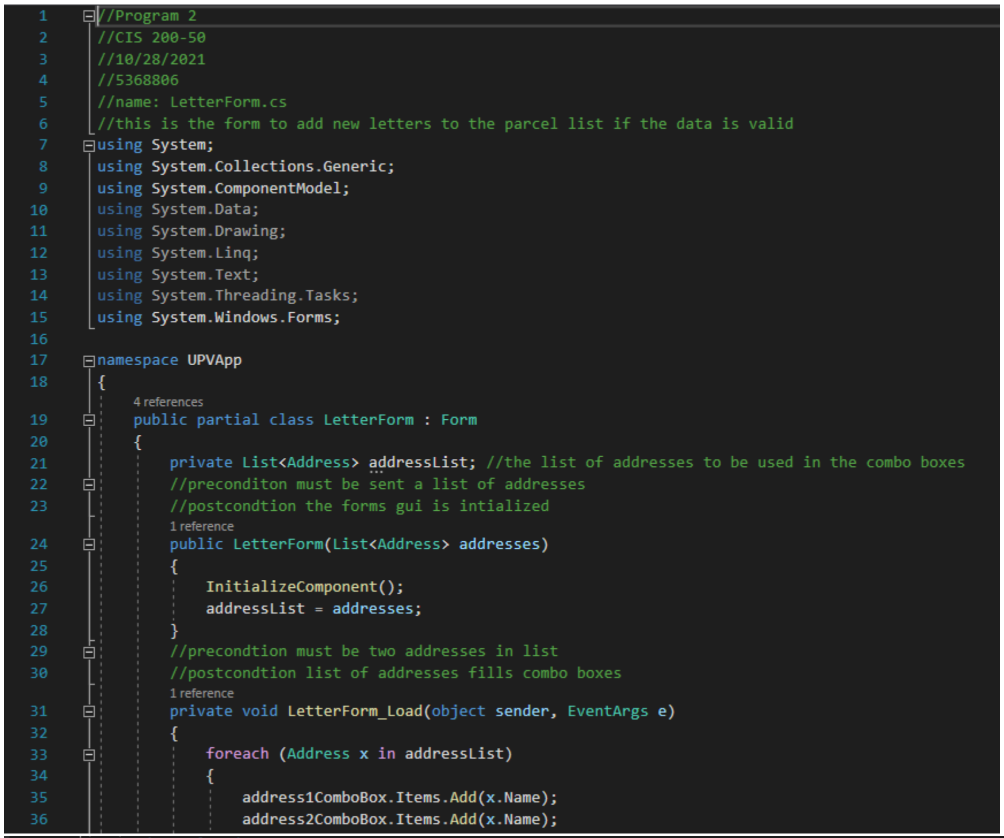


```
        private void addressToolStripMenuItem_Click(object sender, EventArgs e)
        {
            AddressForm addressForm = new AddressForm(); //the address form
            DialogResult result = addressForm.ShowDialog(); //show's address form in dialog box

           if(result == DialogResult.OK) // if address is valid then add to list
            {
                parcelViewer.AddAddress(addressForm.AddressName, addressForm.Address1, addressForm.Address2, addres
                    addressForm.AddressState, Convert.ToInt32(addressForm.AddressZip));
            }
        }
        //precondtion menu is activated, address list is made with at least 2 addresses
        //post condition all addresses are displayed in main text box
        1 reference
        private void listAddressesToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Outputbox.Clear();
            string newLine = Environment.NewLine;//used for new lines
            foreach(Address adds in parcelViewer.AddressList) //cycles through address list and displays them
            {
                Outputbox.AppendText(adds.ToString());
                Outputbox.AppendText(newLine);
                Outputbox.AppendText("_________________________________");
                Outputbox.AppendText(newLine);

            }
        }
        //precondtion menu is activated
        //postcondtion letter form box is opened and if data is valid new letter is added to parcel list
        1 reference
        private void letterToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LetterForm letterform = new LetterForm(parcelViewer.AddressList);//the letter form
            DialogResult result = letterform.ShowDialog();//result from letter dialog box

            if (result == DialogResult.OK)// if data is valid then
            {
                parcelViewer.AddLetter(parcelViewer.AddressAt(letterform.OriginalAddress),parcelViewer.AddressAt(le
                    Convert.ToDecimal(letterform.LetterCost));
```

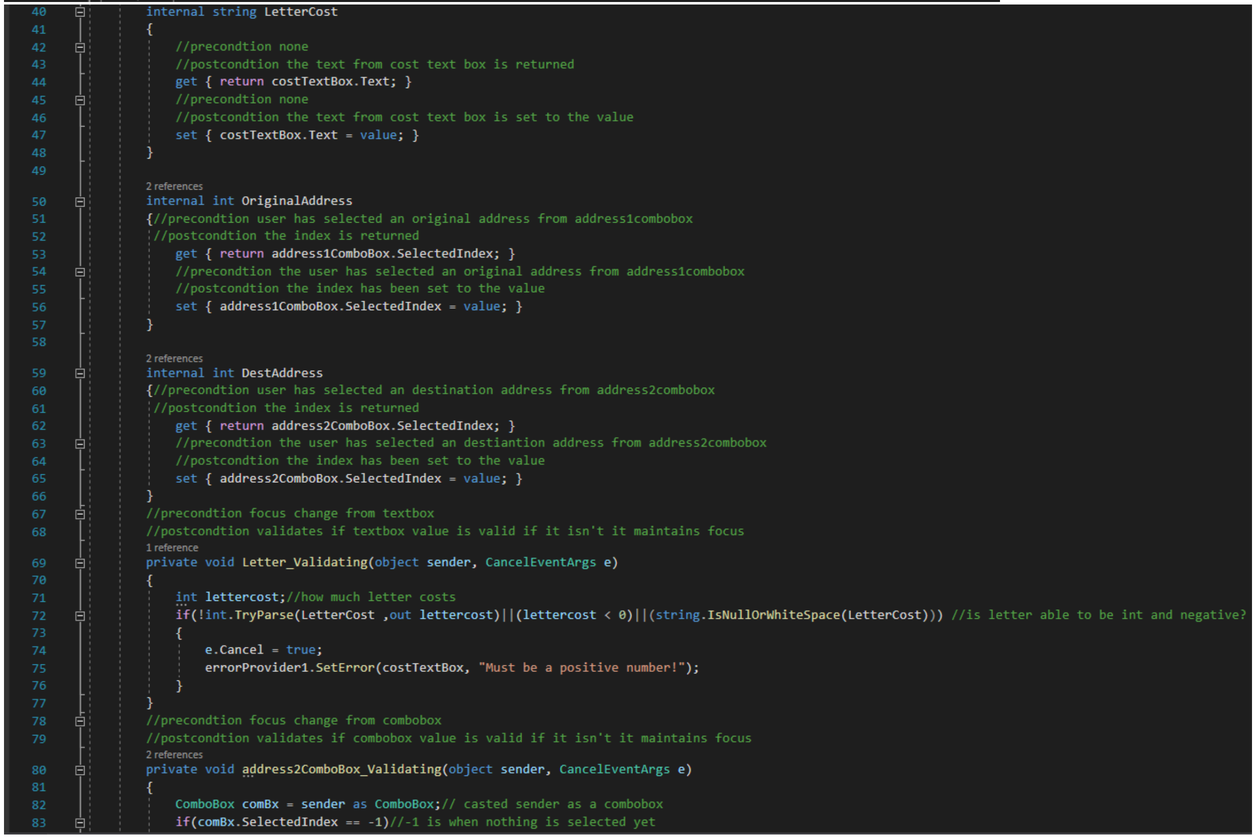


```
            }

        }
        //precondtion menu is activated and there is a list of parcels
        //postcondtion parcels are displayed to main text box

        1 reference
        private void listParcelsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            decimal totalCost = 0; //the cost of all curent parcels in the list
            string newLine = Environment.NewLine; //a shortcut for newlines
            Outputbox.Clear();
            foreach (Parcel x in parcelViewer.ParcelList) //cycles through parcel list
            {
                Outputbox.AppendText(x.ToString());
                Outputbox.AppendText(newLine);
                Outputbox.AppendText("_________________________________");
                Outputbox.AppendText(newLine);
                totalCost = x.CalcCost() + totalCost;
            }
            Outputbox.AppendText(newLine);
            Outputbox.AppendText($"Total Cost: {totalCost:c}");
        }
    }
}
```

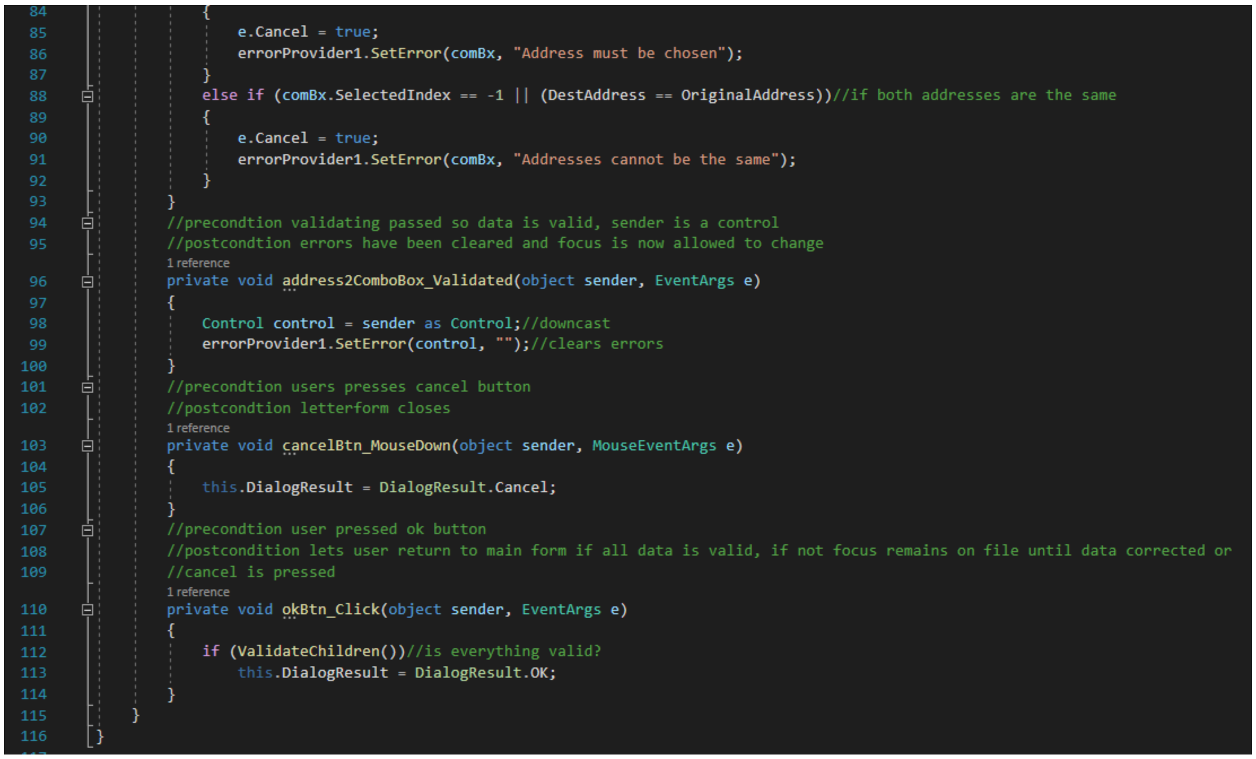
The previous screen shots are the code behind the main gui windows form



```
//Program 2
//CIS 200-50
//10/28/2021
//5368806
//name: LetterForm.cs
//this is the form to add new letters to the parcel list if the data is valid
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 UPVApp
{
    4 references
    public partial class LetterForm : Form
    {
        private List<Address> addressList; //the list of addresses to be used in the combo boxes
        //preconditon must be sent a list of addresses
        //postcondtion the forms gui is intialized
        1 reference
        public LetterForm(List<Address> addresses)
        {
            InitializeComponent();
            addressList = addresses;
        }
        //precondtion must be two addresses in list
        //postcondtion list of addresses fills combo boxes
        1 reference
        private void LetterForm_Load(object sender, EventArgs e)
        {
            foreach (Address x in addressList)
            {
                address1ComboBox.Items.Add(x.Name);
                address2ComboBox.Items.Add(x.Name);
```

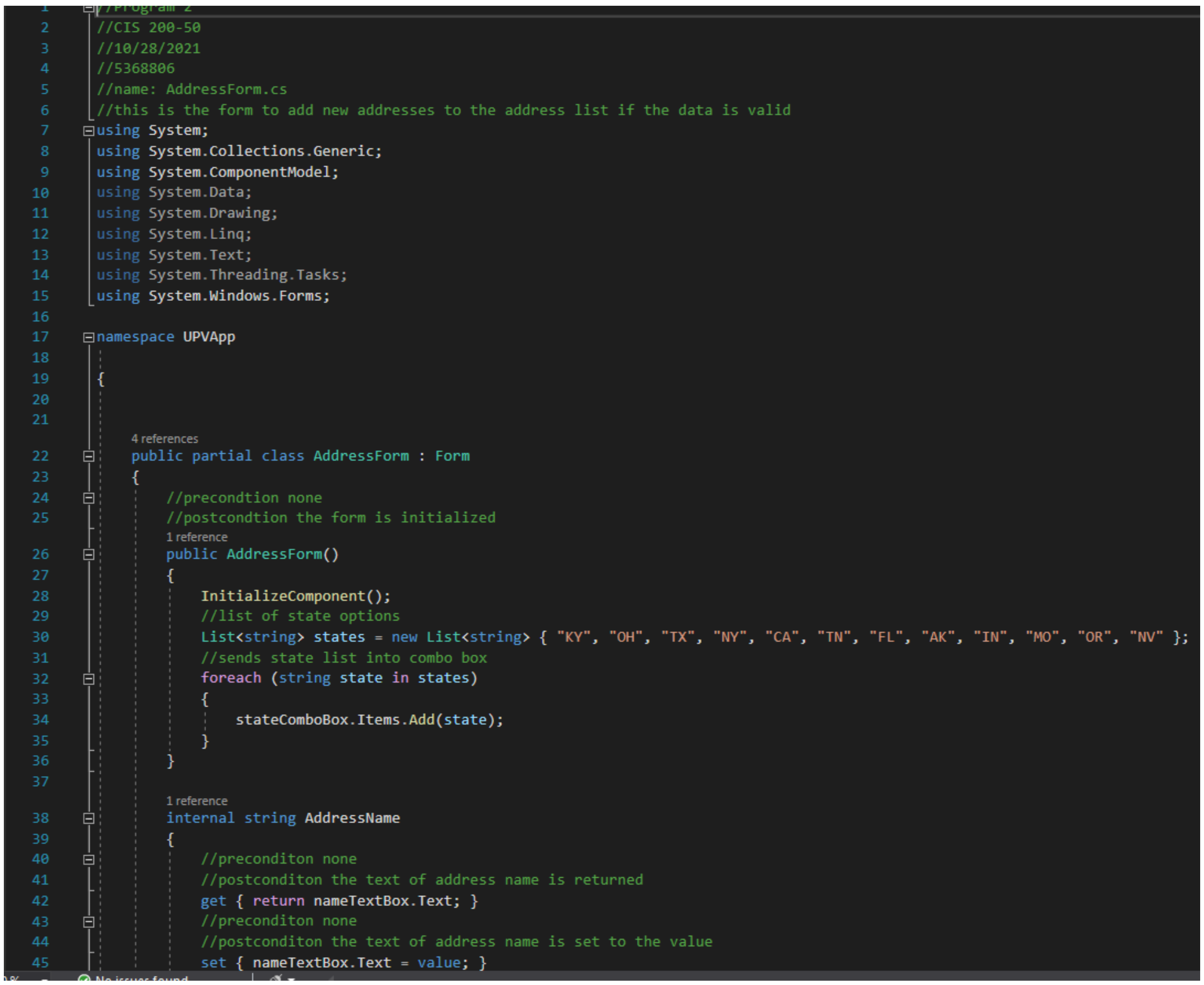


```
        internal string LetterCost
        {
            //precondtion none
            //postcondtion the text from cost text box is returned
            get { return costTextBox.Text; }
            //precondtion none
            //postcondtion the text from cost text box is set to the value
            set { costTextBox.Text = value; }
        }

        2 references
        internal int OriginalAddress
        {//precondtion user has selected an original address from address1combobox
         //postcondtion the index is returned
            get { return address1ComboBox.SelectedIndex; }
            //precondtion the user has selected an original address from address1combobox
            //postcondtion the index has been set to the value
            set { address1ComboBox.SelectedIndex = value; }
        }

        2 references
        internal int DestAddress
        {//precondtion user has selected an destination address from address2combobox
         //postcondtion the index is returned
            get { return address2ComboBox.SelectedIndex; }
            //precondtion the user has selected an destiantion address from address2combobox
            //postcondtion the index has been set to the value
            set { address2ComboBox.SelectedIndex = value; }
        }
        //precondtion focus change from textbox
        //postcondtion validates if textbox value is valid if it isn't it maintains focus
        1 reference
        private void Letter_Validating(object sender, CancelEventArgs e)
        {
            int lettercost;//how much letter costs
            if(!int.TryParse(LetterCost ,out lettercost)||(lettercost < 0)||(string.IsNullOrWhiteSpace(LetterCost))) //is letter able to be int and negative?
            {
                e.Cancel = true;
                errorProvider1.SetError(costTextBox, "Must be a positive number!");
            }
        }
        //precondtion focus change from combobox
        //postcondtion validates if combobox value is valid if it isn't it maintains focus
        2 references
        private void address2ComboBox_Validating(object sender, CancelEventArgs e)
        {
            ComboBox comBx = sender as ComboBox;// casted sender as a combobox
            if(comBx.SelectedIndex == -1)//-1 is when nothing is selected yet
```

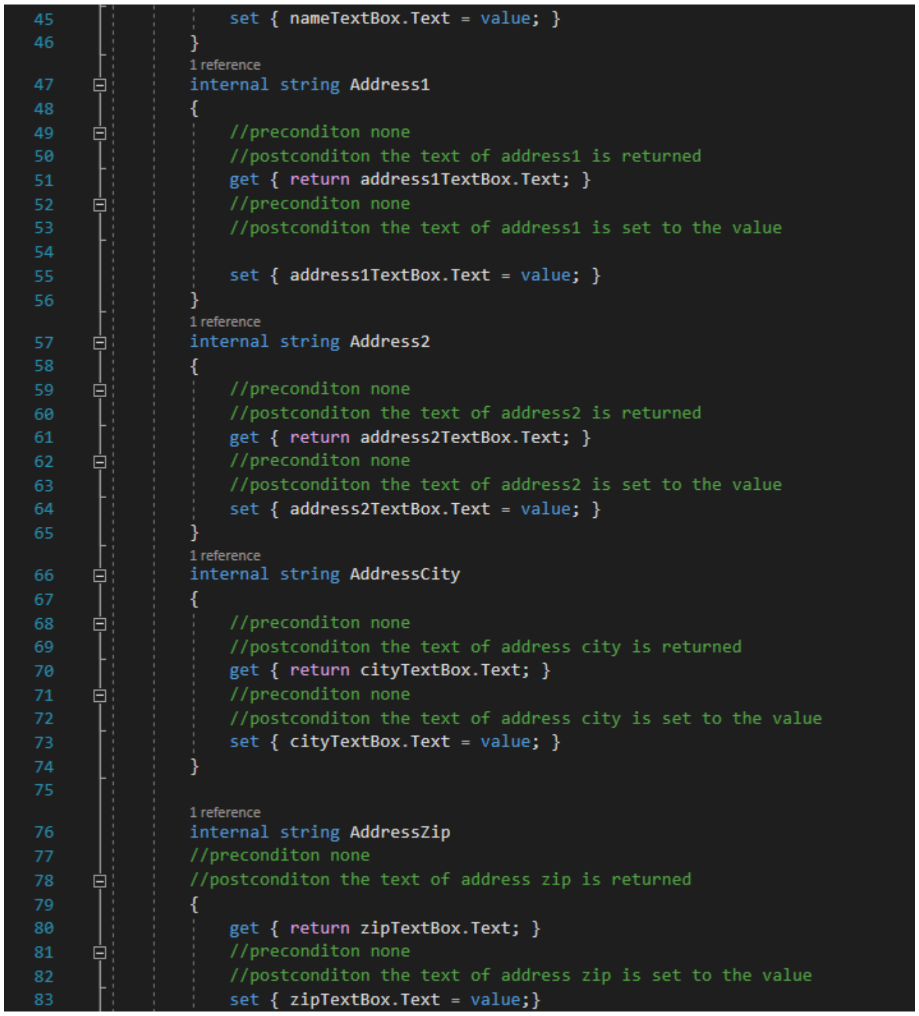


```
            {
                e.Cancel = true;
                errorProvider1.SetError(comBx, "Address must be chosen");
            }
            else if (comBx.SelectedIndex == -1 || (DestAddress == OriginalAddress))//if both addresses are the same
            {
                e.Cancel = true;
                errorProvider1.SetError(comBx, "Addresses cannot be the same");
            }
        }
        //precondtion validating passed so data is valid, sender is a control
        //postcondtion errors have been cleared and focus is now allowed to change
        1 reference
        private void address2ComboBox_Validated(object sender, EventArgs e)
        {
            Control control = sender as Control;//downcast
            errorProvider1.SetError(control, "");//clears errors
        }
        //precondtion users presses cancel button
        //postcondtion letterform closes
        1 reference
        private void cancelBtn_MouseDown(object sender, MouseEventArgs e)
        {
            this.DialogResult = DialogResult.Cancel;
        }
        //precondtion user pressed ok button
        //postcondition lets user return to main form if all data is valid, if not focus remains on file until data corrected or
        //cancel is pressed
        1 reference
        private void okBtn_Click(object sender, EventArgs e)
        {
            if (ValidateChildren())//is everything valid?
                this.DialogResult = DialogResult.OK;
        }
    }
}
```

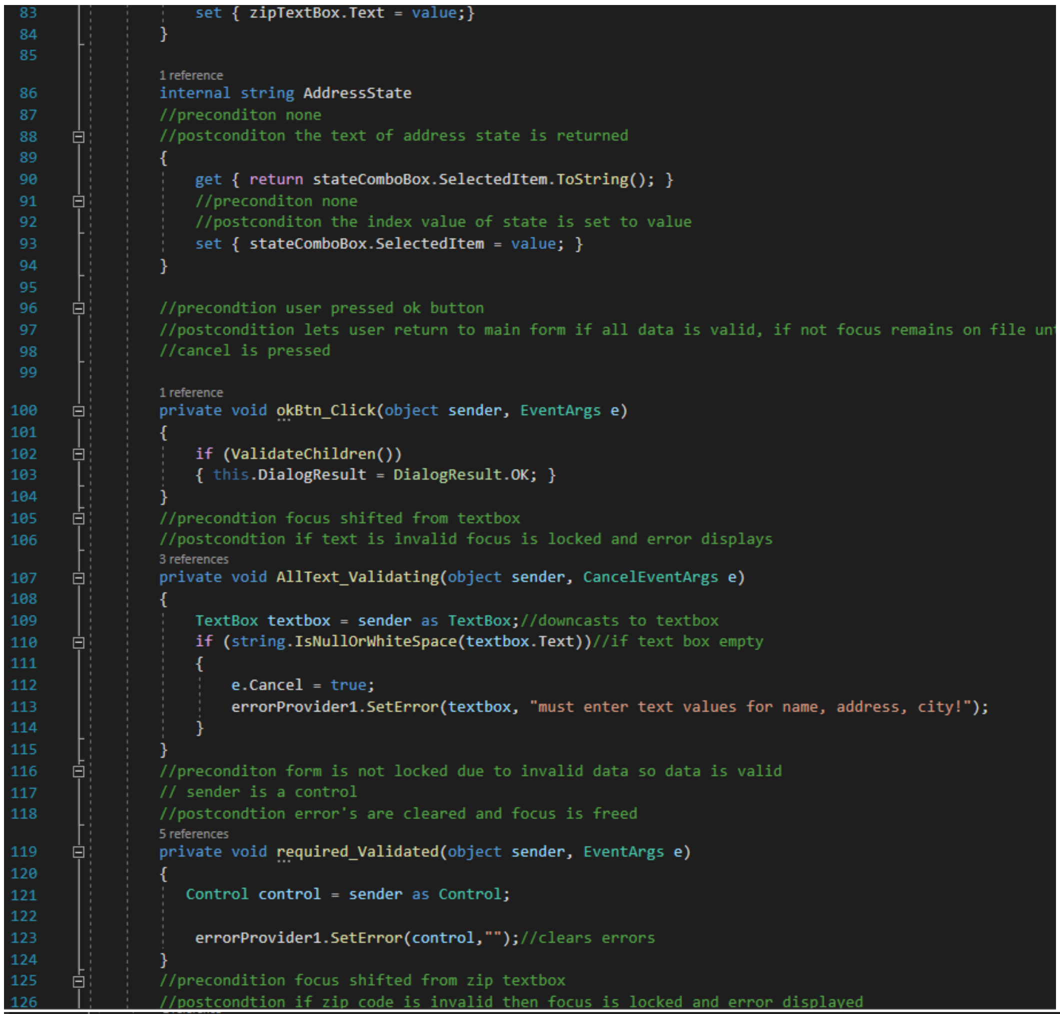
The previous screen shots are from the letter form gui's code



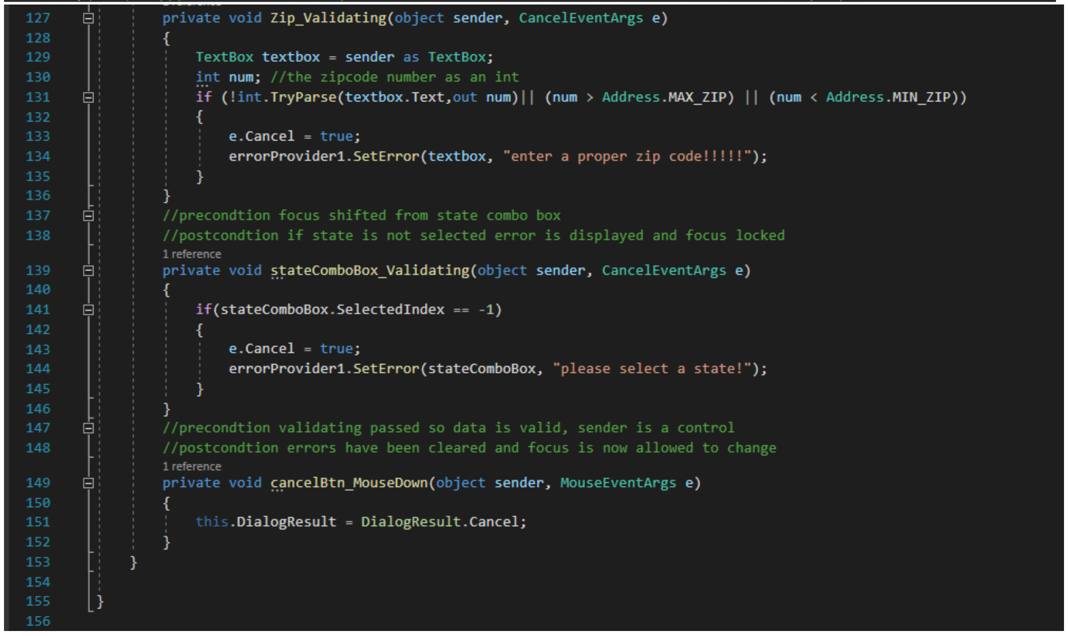
```csharp
//Program 2
//CIS 200-50
//10/28/2021
//5368806
//name: AddressForm.cs
//this is the form to add new addresses to the address list if the data is valid
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace UPVApp

{


    public partial class AddressForm : Form
    {
        //precondtion none
        //postcondtion the form is initialized
        public AddressForm()
        {
            InitializeComponent();
            //list of state options
            List<string> states = new List<string> { "KY", "OH", "TX", "NY", "CA", "TN", "FL", "AK", "IN", "MO", "OR", "NV" };
            //sends state list into combo box
            foreach (string state in states)
            {
                stateComboBox.Items.Add(state);
            }
        }

        internal string AddressName
        {
            //preconditon none
            //postconditon the text of address name is returned
            get { return nameTextBox.Text; }
            //preconditon none
            //postconditon the text of address name is set to the value
            set { nameTextBox.Text = value; }
```



```
            set { nameTextBox.Text = value; }
        }
        1 reference
        internal string Address1
        {
            //preconditon none
            //postconditon the text of address1 is returned
            get { return address1TextBox.Text; }
            //preconditon none
            //postconditon the text of address1 is set to the value

            set { address1TextBox.Text = value; }
        }
        1 reference
        internal string Address2
        {
            //preconditon none
            //postconditon the text of address2 is returned
            get { return address2TextBox.Text; }
            //preconditon none
            //postconditon the text of address2 is set to the value
            set { address2TextBox.Text = value; }
        }
        1 reference
        internal string AddressCity
        {
            //preconditon none
            //postconditon the text of address city is returned
            get { return cityTextBox.Text; }
            //preconditon none
            //postconditon the text of address city is set to the value
            set { cityTextBox.Text = value; }
        }

        1 reference
        internal string AddressZip
        //preconditon none
        //postconditon the text of address zip is returned
        {
            get { return zipTextBox.Text; }
            //preconditon none
            //postconditon the text of address zip is set to the value
            set { zipTextBox.Text = value;}
```



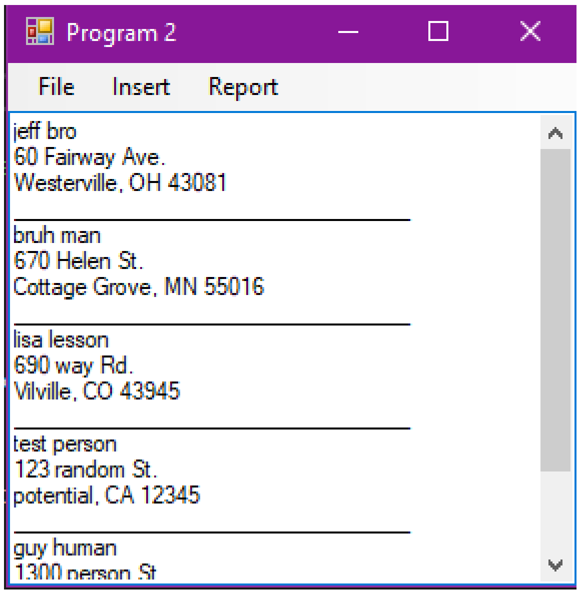
```
            set { zipTextBox.Text = value;}
        }

        1 reference
        internal string AddressState
        //preconditon none
        //postconditon the text of address state is returned
        {
            get { return stateComboBox.SelectedItem.ToString(); }
            //preconditon none
            //postconditon the index value of state is set to value
            set { stateComboBox.SelectedItem = value; }
        }

        //precondtion user pressed ok button
        //postcondition lets user return to main form if all data is valid, if not focus remains on file un
        //cancel is pressed

        1 reference
        private void okBtn_Click(object sender, EventArgs e)
        {
            if (ValidateChildren())
            { this.DialogResult = DialogResult.OK; }
        }
        //precondtion focus shifted from textbox
        //postcondtion if text is invalid focus is locked and error displays
        3 references
        private void AllText_Validating(object sender, CancelEventArgs e)
        {
            TextBox textbox = sender as TextBox;//downcasts to textbox
            if (string.IsNullOrWhiteSpace(textbox.Text))//if text box empty
            {
                e.Cancel = true;
                errorProvider1.SetError(textbox, "must enter text values for name, address, city!");
            }
        }
        //preconditon form is not locked due to invalid data so data is valid
        // sender is a control
        //postcondtion error's are cleared and focus is freed
        5 references
        private void required_Validated(object sender, EventArgs e)
        {
           Control control = sender as Control;

            errorProvider1.SetError(control,"");//clears errors
        }
        //precondition focus shifted from zip textbox
        //postcondtion if zip code is invalid then focus is locked and error displayed
```



```
        private void Zip_Validating(object sender, CancelEventArgs e)
        {
            TextBox textbox = sender as TextBox;
            int num; //the zipcode number as an int
            if (!int.TryParse(textbox.Text,out num)|| (num > Address.MAX_ZIP) || (num < Address.MIN_ZIP))
            {
                e.Cancel = true;
                errorProvider1.SetError(textbox, "enter a proper zip code!!!!!");
            }
        }
        //precondtion focus shifted from state combo box
        //postcondtion if state is not selected error is displayed and focus locked
        1 reference
        private void stateComboBox_Validating(object sender, CancelEventArgs e)
        {
            if(stateComboBox.SelectedIndex == -1)
            {
                e.Cancel = true;
                errorProvider1.SetError(stateComboBox, "please select a state!");
            }
        }
        //precondtion validating passed so data is valid, sender is a control
        //postcondtion errors have been cleared and focus is now allowed to change
        1 reference
        private void cancelBtn_MouseDown(object sender, MouseEventArgs e)
        {
            this.DialogResult = DialogResult.Cancel;
        }
    }

}
```

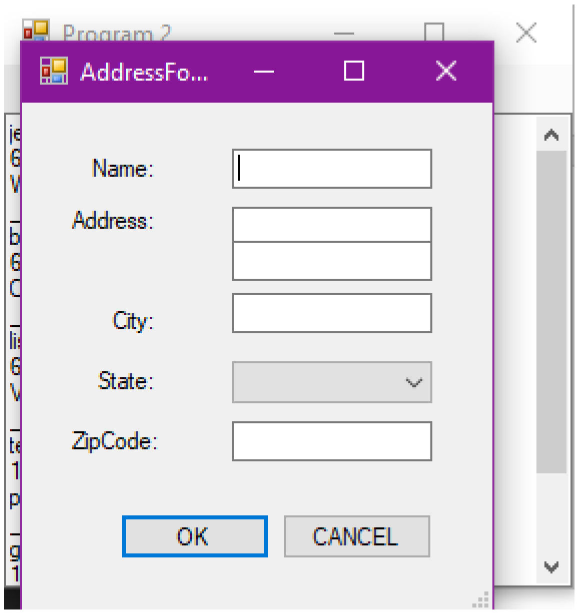
The previous screen shots are from the address form's code

Here is the program reporting the current addresses:

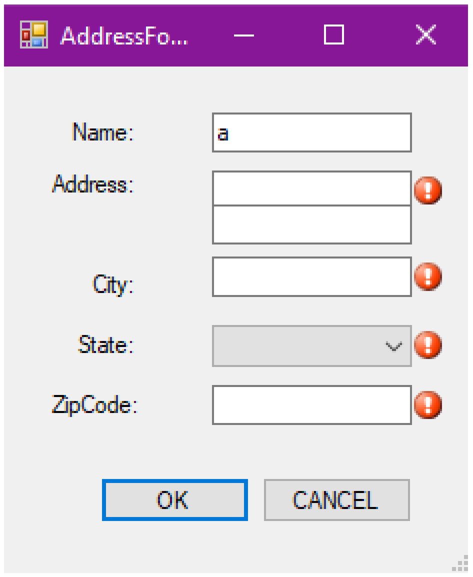


```
jeff bro
60 Fairway Ave.
Westerville, OH 43081

bruh man
670 Helen St.
Cottage Grove, MN 55016

lisa lesson
690 way Rd.
Vilville, CO 43945

test person
123 random St.
potential, CA 12345

guy human
1300 person St
```

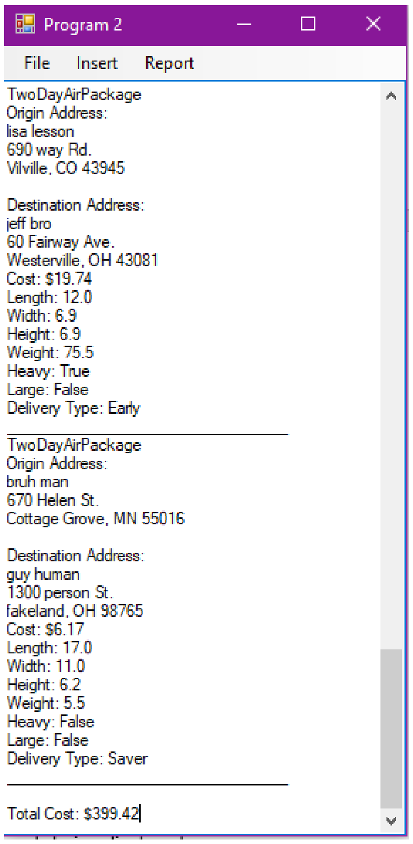
Here is the program with the insert address form:



The form has validation to make sure it can't be crashed:



Here are the parcels being displayed:



```
TwoDayAirPackage
Origin Address:
lisa lesson
690 way Rd.
Vilville, CO 43945

Destination Address:
jeff bro
60 Fairway Ave.
Westerville, OH 43081
Cost: $19.74
Length: 12.0
Width: 6.9
Height: 6.9
Weight: 75.5
Heavy: True
Large: False
Delivery Type: Early
________________________________
TwoDayAirPackage
Origin Address:
bruh man
670 Helen St.
Cottage Grove, MN 55016

Destination Address:
guy human
1300 person St.
fakeland, OH 98765
Cost: $6.17
Length: 17.0
Width: 11.0
Height: 6.2
Weight: 5.5
Heavy: False
Large: False
Delivery Type: Saver
________________________________

Total Cost: $399.42
```