

```
AddressForm.cs Prog2Form.cs X LetterForm.cs
UPVApp.Prog2Form parcelViewer
1 //Program 2
2 //CIS 200-50
3 //10/28/2021
4 //5368806
5 //name: Prog2Form.cs
6 //this file creates the base gui windows form page that will display address and parcels
7 //using a menu to access all its features.
8 using System;
9 using System.Collections.Generic;
10 using System.ComponentModel;
11 using System.Data;
12 using System.Drawing;
13 using System.Linq;
14 using System.Text;
15 using System.Threading.Tasks;
16 using System.Windows.Forms;
17
18 namespace UPVApp
19 {
20     3 references
21     public partial class Prog2Form : Form
22     {
23         private UserParcelView parcelViewer; //class that instructor made to simplify the class and gui integration
24         //precondition: none
25         //postcondition: test parcels and address are added to lists and program is initialized
26         1 reference
27         public Prog2Form()
28         {
29             InitializeComponent();
30             parcelViewer = new UserParcelView();
31             //test addresses
32             parcelViewer.AddAddress("jeff bro", "60 Fairway Ave.",
33 "Westerville", "OH", 43081);
34             parcelViewer.AddAddress("bruh man", "670 Helen St.",
35 "Cottage Grove", "MN", 55016);
36             parcelViewer.AddAddress("lisa lesson", "690 way Rd.",
37 "Vilville", "CO", 43945);
38             parcelViewer.AddAddress("test person", "123 random St.",
39 "potential", "CA", 12345);
40
41             parcelViewer.AddAddress("guy number", "1200 person St.",
42 "fakeland", "OH", 98765);
43             //test parcels
44             parcelViewer.AddLetter(parcelViewer.AddressAt(0), parcelViewer.AddressAt(1), 10.00M);
45             parcelViewer.AddLetter(parcelViewer.AddressAt(1), parcelViewer.AddressAt(2), 16.00M);
46             parcelViewer.AddLetter(parcelViewer.AddressAt(3), parcelViewer.AddressAt(1), 100.00M);
47             parcelViewer.AddLetter(parcelViewer.AddressAt(4), parcelViewer.AddressAt(3), 13.00M);
48             parcelViewer.AddLetter(parcelViewer.AddressAt(0), parcelViewer.AddressAt(2), 109.00M);
49             parcelViewer.AddNextDayAirPackage(parcelViewer.AddressAt(2), parcelViewer.AddressAt(4), 5.6, 9.0, 2.4,
50 5.25, 5.25M);
51             parcelViewer.AddNextDayAirPackage(parcelViewer.AddressAt(2), parcelViewer.AddressAt(3), 8, 12, 3.4,
52 5.51, 15.10M);
53             parcelViewer.AddNextDayAirPackage(parcelViewer.AddressAt(3), parcelViewer.AddressAt(4), 10.5, 6.5, 9.5,
54 15.5, 51.00M);
55             parcelViewer.AddTwoDayAirPackage(parcelViewer.AddressAt(4), parcelViewer.AddressAt(4), 23.11, 23.89, 21.0,
56 81, TwoDayAirPackage.Delivery.Saver);
57             parcelViewer.AddTwoDayAirPackage(parcelViewer.AddressAt(2), parcelViewer.AddressAt(0), 12.0, 6.9, 6.9,
58 75.5, TwoDayAirPackage.Delivery.Early);
59             parcelViewer.AddTwoDayAirPackage(parcelViewer.AddressAt(1), parcelViewer.AddressAt(4), 17.0, 11.0, 6.2,
60 5.53, TwoDayAirPackage.Delivery.Saver);
61         }
62         //precondition the menu is activated
63         //post condition information about file is shown
64         1 reference
65         private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
66         {
67             MessageBox.Show("This a program that uses a front end \n gui to manage our UPS package objects\n" +
68 "$5368806 CIS 200-50", "About Me",
69 MessageBoxButtons.OK, MessageBoxIcon.Information);
70         }
71         //precondition the menu is activated
72         //postcondition the file is closed
73         1 reference
74         private void exitToolStripMenuItem_Click(object sender, EventArgs e)
75         {
76             Close();
77         }
78         //precondition the menu is activated
79         //postcondition the address form is displayed and if data is valid then address is added to current list of addresses
80     }
81 }
```

```

75 private void addressToolStripMenuItem_Click(object sender, EventArgs e)
76 {
77     AddressForm addressForm = new AddressForm(); //the address form
78     DialogResult result = addressForm.ShowDialog(); //show's address form in dialog box
79
80     if(result == DialogResult.OK) // if address is valid then add to list
81     {
82         parcelViewer.AddAddress(addressForm.AddressName, addressForm.Address1, addressForm.Address2, addressForm.AddressState, Convert.ToInt32(addressForm.AddressZip));
83     }
84 }
85
86 //precondition menu is activated, address list is made with at least 2 addresses
87 //post condition all addresses are displayed in main text box
1 reference
88 private void listAddressesToolStripMenuItem_Click(object sender, EventArgs e)
89 {
90     Outputbox.Clear();
91     string newLine = Environment.NewLine; //used for new lines
92     foreach(Address adds in parcelViewer.AddressList) //cycles through address list and displays them
93     {
94         Outputbox.AppendText(adds.ToString());
95         Outputbox.AppendText(newLine);
96         Outputbox.AppendText("_____");
97         Outputbox.AppendText(newLine);
98     }
99 }
100
101 //precondition menu is activated
102 //postcondition letter form box is opened and if data is valid new letter is added to parcel list
1 reference
103 private void letterToolStripMenuItem_Click(object sender, EventArgs e)
104 {
105     LetterForm letterform = new LetterForm(parcelViewer.AddressList); //the letter form
106     DialogResult result = letterform.ShowDialog(); //result from letter dialog box
107
108     if (result == DialogResult.OK) // if data is valid then
109     {
110         parcelViewer.AddLetter(parcelViewer.AddressAt(letterform.OriginalAddress), parcelViewer.AddressAt(letterform.Address), Convert.ToDecimal(letterform.LetterCost));
111     }
112 }
113
114 //precondition menu is activated and there is a list of parcels
115 //postcondition parcels are displayed to main text box
116
117 1 reference
118 private void listParcelsToolStripMenuItem_Click(object sender, EventArgs e)
119 {
120     decimal totalCost = 0; //the cost of all current parcels in the list
121     string newLine = Environment.NewLine; //a shortcut for newlines
122     Outputbox.Clear();
123     foreach (Parcel x in parcelViewer.ParcelList) //cycles through parcel list
124     {
125         Outputbox.AppendText(x.ToString());
126         Outputbox.AppendText(newLine);
127         Outputbox.AppendText("_____");
128         Outputbox.AppendText(newLine);
129         totalCost = x.CalcCost() + totalCost;
130     }
131     Outputbox.AppendText(newLine);
132     Outputbox.AppendText($"Total Cost: {totalCost:c}");
133 }
134 }
135 }
136

```

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

```

1  //Program 2
2  //CIS 200-50
3  //10/28/2021
4  //5368806
5  //name: LetterForm.cs
6  //this is the form to add new letters to the parcel list if the data is valid
7  using System;
8  using System.Collections.Generic;
9  using System.ComponentModel;
10 using System.Data;
11 using System.Drawing;
12 using System.Linq;
13 using System.Text;
14 using System.Threading.Tasks;
15 using System.Windows.Forms;
16
17 namespace UPVApp
18 {
19     4 references
20     public partial class LetterForm : Form
21     {
22         private List<Address> addresslist; //the list of addresses to be used in the combo boxes
23         //preconditon must be sent a list of addresses
24         //postcondtion the forms gui is intialized
25         1 reference
26         public LetterForm(List<Address> addresses)
27         {
28             InitializeComponent();
29             addressList = addresses;
30         }
31         //precondition must be two addresses in list
32         //postcondition list of addresses fills combo boxes
33         1 reference
34         private void LetterForm_Load(object sender, EventArgs e)
35         {
36             foreach (Address x in addressList)
37             {
38                 address1ComboBox.Items.Add(x.Name);
39                 address2ComboBox.Items.Add(x.Name);
40             }
41         }
42     }
43 }

```

```

40 internal string LetterCost
41 {
42     //precondtion none
43     //postcondtion the text from cost text box is returned
44     get { return costTextBox.Text; }
45     //precondtion none
46     //postcondtion the text from cost text box is set to the value
47     set { costTextBox.Text = value; }
48 }
49
50 2 references
51 internal int OriginalAddress
52 {
53     //precondition user has selected an original address from address1combobox
54     //postcondtion the index is returned
55     get { return address1ComboBox.SelectedIndex; }
56     //precondition the user has selected an original address from address1combobox
57     //postcondtion the index has been set to the value
58     set { address1ComboBox.SelectedIndex = value; }
59 }
60
61 2 references
62 internal int DestAddress
63 {
64     //precondition user has selected an destination address from address2combobox
65     //postcondtion the index is returned
66     get { return address2ComboBox.SelectedIndex; }
67     //precondition the user has selected an destiantion address from address2combobox
68     //postcondtion the index has been set to the value
69     set { address2ComboBox.SelectedIndex = value; }
70 }
71
72 //precondition focus change from textbox
73 //postcondtion validates if textbox value is valid if it isn't it maintains focus
74 1 reference
75 private void Letter_Validating(object sender, CancelEventArgs e)
76 {
77     int lettercost; //how much letter costs
78     if (int.TryParse(LetterCost, out lettercost) || (lettercost < 0) || (string.IsNullOrWhiteSpace(LetterCost))) //is letter able to be int and negative?
79     {
80         e.Cancel = true;
81         errorProvider1.SetError(costTextBox, "Must be a positive number!");
82     }
83 }
84
85 //precondition focus change from combobox
86 //postcondtion validates if combobox value is valid if it isn't it maintains focus
87 2 references
88 private void address2ComboBox_Validating(object sender, CancelEventArgs e)
89 {
90     ComboBox comBx = sender as ComboBox; // casted sender as a combobox
91     if (comBx.SelectedIndex == -1) // -1 is when nothing is selected yet
92     {
93         e.Cancel = true;
94         errorProvider1.SetError(address2ComboBox, "Must select an address");
95     }
96 }

```

```

84         {
85             e.Cancel = true;
86             errorProvider1.SetError(comBx, "Address must be chosen");
87         }
88         else if (comBx.SelectedIndex == -1 || (DestAddress == OriginalAddress))//if both addresses are the same
89         {
90             e.Cancel = true;
91             errorProvider1.SetError(comBx, "Addresses cannot be the same");
92         }
93     }
94     //precondition validating passed so data is valid, sender is a control
95     //postcondition errors have been cleared and focus is now allowed to change
96     1 reference
97     private void address2ComboBox_Validated(object sender, EventArgs e)
98     {
99         Control control = sender as Control;//downcast
100         errorProvider1.SetError(control, ""); //clears errors
101     }
102     //precondition users presses cancel button
103     //postcondition letterform closes
104     1 reference
105     private void cancelBtn_MouseDown(object sender, MouseEventArgs e)
106     {
107         this.DialogResult = DialogResult.Cancel;
108     }
109     //precondition user pressed ok button
110     //postcondition lets user return to main form if all data is valid, if not focus remains on file until data corrected or
111     //cancel is pressed
112     1 reference
113     private void okBtn_Click(object sender, EventArgs e)
114     {
115         if (ValidateChildren())//is everything valid?
116             this.DialogResult = DialogResult.OK;
117     }

```

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

```

1  //Program 2
2  //CIS 200-50
3  //10/28/2021
4  //5368806
5  //name: AddressForm.cs
6  //this is the form to add new addresses to the address list if the data is valid
7  using System;
8  using System.Collections.Generic;
9  using System.ComponentModel;
10 using System.Data;
11 using System.Drawing;
12 using System.Linq;
13 using System.Text;
14 using System.Threading.Tasks;
15 using System.Windows.Forms;
16
17 namespace UPVApp
18 {
19
20
21
22     4 references
23     public partial class AddressForm : Form
24     {
25         //precondition none
26         //postcondition the form is initialized
27         1 reference
28         public AddressForm()
29         {
30             InitializeComponent();
31             //list of state options
32             List<string> states = new List<string> { "KY", "OH", "TX", "NY", "CA", "TN", "FL", "AK", "IN", "MO", "OR", "NV" };
33             //sends state list into combo box
34             foreach (string state in states)
35             {
36                 stateComboBox.Items.Add(state);
37             }
38
39         1 reference
40         internal string AddressName
41         {
42             //precondition none
43             //postcondition the text of address name is returned
44             get { return nameTextBox.Text; }
45             //precondition none
46             //postcondition the text of address name is set to the value
47             set { nameTextBox.Text = value; }
48         }
49     }
50 }

```

```

45         set { nameTextBox.Text = value; }
46     }
47     1 reference
48     internal string Address1
49     {
50         //preconditon none
51         //postconditon the text of address1 is returned
52         get { return address1TextBox.Text; }
53         //preconditon none
54         //postconditon the text of address1 is set to the value
55         set { address1TextBox.Text = value; }
56     }
57     1 reference
58     internal string Address2
59     {
60         //preconditon none
61         //postconditon the text of address2 is returned
62         get { return address2TextBox.Text; }
63         //preconditon none
64         //postconditon the text of address2 is set to the value
65         set { address2TextBox.Text = value; }
66     }
67     1 reference
68     internal string AddressCity
69     {
70         //preconditon none
71         //postconditon the text of address city is returned
72         get { return cityTextBox.Text; }
73         //preconditon none
74         //postconditon the text of address city is set to the value
75         set { cityTextBox.Text = value; }
76     }
77     1 reference
78     internal string AddressZip
79     {
80         //preconditon none
81         //postconditon the text of address zip is returned
82         get { return zipTextBox.Text; }
83         //preconditon none
84         //postconditon the text of address zip is set to the value
85         set { zipTextBox.Text = value; }
86     }

```

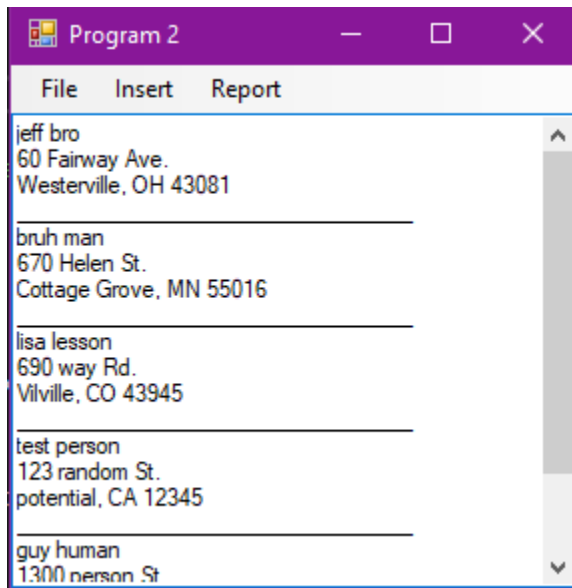
```

83         set { zipTextBox.Text = value; }
84     }
85
86     1 reference
87     internal string AddressState
88     //precondition none
89     //postcondition the text of address state is returned
90     {
91         get { return stateComboBox.SelectedItem.ToString(); }
92         //precondition none
93         //postcondition the index value of state is set to value
94         set { stateComboBox.SelectedItem = value; }
95     }
96
97     //precondition user pressed ok button
98     //postcondition lets user return to main form if all data is valid, if not focus remains on file un
99     //cancel is pressed
100
101     1 reference
102     private void okBtn_Click(object sender, EventArgs e)
103     {
104         if (ValidateChildren())
105         { this.DialogResult = DialogResult.OK; }
106     }
107
108     //precondition focus shifted from textbox
109     //postcondition if text is invalid focus is locked and error displays
110     3 references
111     private void AllText_Validating(object sender, CancelEventArgs e)
112     {
113         TextBox textbox = sender as TextBox; //downcasts to textbox
114         if (string.IsNullOrEmpty(textbox.Text)) //if text box empty
115         {
116             e.Cancel = true;
117             errorProvider1.SetError(textbox, "must enter text values for name, address, city!");
118         }
119     }
120
121     //precondition form is not locked due to invalid data so data is valid
122     // sender is a control
123     //postcondition error's are cleared and focus is freed
124     5 references
125     private void required_Validated(object sender, EventArgs e)
126     {
127         Control control = sender as Control;
128
129         errorProvider1.SetError(control, ""); //clears errors
130     }
131
132     //precondition focus shifted from zip textbox
133     //postcondition if zip code is invalid then focus is locked and error displayed
134     private void Zip_Validating(object sender, CancelEventArgs e)
135     {
136         TextBox textbox = sender as TextBox;
137         int num; //the zipcode number as an int
138         if (!int.TryParse(textbox.Text, out num) || (num > Address.MAX_ZIP) || (num < Address.MIN_ZIP))
139         {
140             e.Cancel = true;
141             errorProvider1.SetError(textbox, "enter a proper zip code!!!!");
142         }
143     }
144
145     //precondition focus shifted from state combo box
146     //postcondition if state is not selected error is displayed and focus locked
147     1 reference
148     private void stateComboBox_Validating(object sender, CancelEventArgs e)
149     {
150         if (stateComboBox.SelectedIndex == -1)
151         {
152             e.Cancel = true;
153             errorProvider1.SetError(stateComboBox, "please select a state!");
154         }
155     }
156
157     //precondition validating passed so data is valid, sender is a control
158     //postcondition errors have been cleared and focus is now allowed to change
159     1 reference
160     private void cancelBtn_MouseDown(object sender, MouseEventArgs e)
161     {
162         this.DialogResult = DialogResult.Cancel;
163     }
164
165 }

```

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

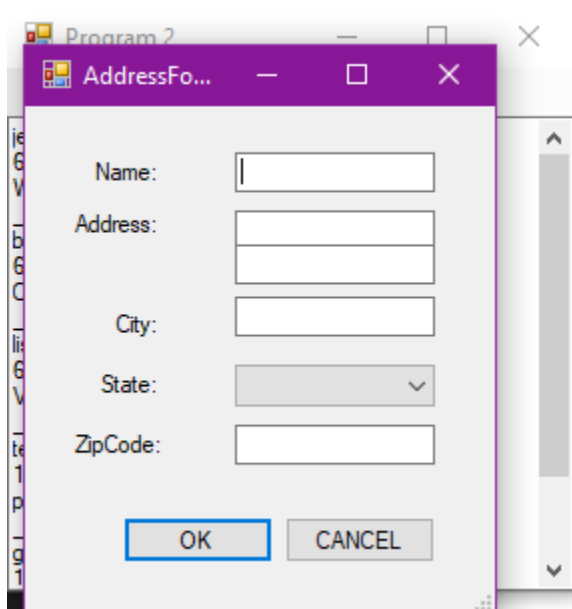
Here is the program reporting the current addresses:



The screenshot shows a window titled "Program 2" with a menu bar containing "File", "Insert", and "Report". The main area is a list box containing five entries, each separated by a horizontal line. The entries are:

- 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 nerson St

Here is the program with the insert address form:

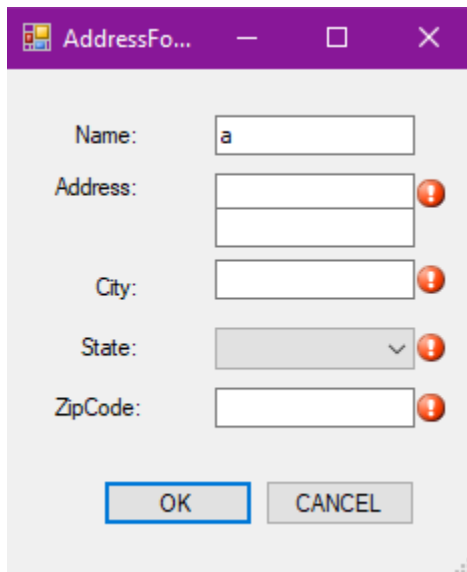


The screenshot shows the "Program 2" window with an "AddressForm" dialog box open in front of it. The dialog box has a title bar "AddressFo..." and contains the following fields:

- Name:
- Address:
- City:
- State:
- ZipCode:

At the bottom of the dialog box are two buttons: "OK" and "CANCEL".

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

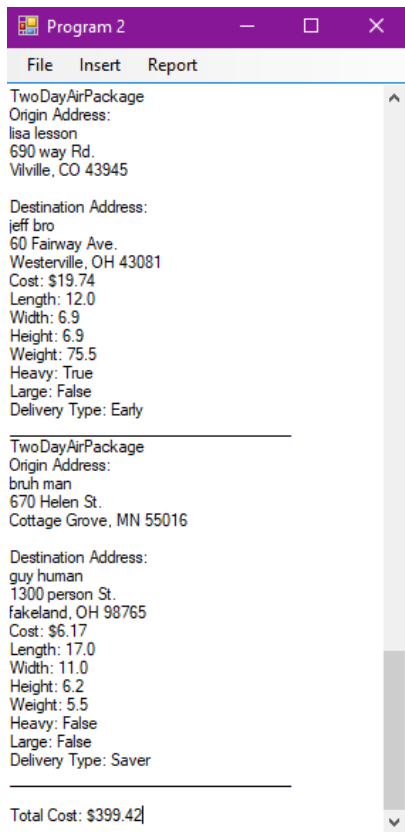


The screenshot shows a window titled "AddressFo...". It contains a form with the following fields and validation status:

- Name: (Valid)
- Address: (Invalid, indicated by a red exclamation mark icon)
- City: (Invalid, indicated by a red exclamation mark icon)
- State: (Invalid, indicated by a red exclamation mark icon)
- ZipCode: (Invalid, indicated by a red exclamation mark icon)

At the bottom of the form are two buttons: "OK" and "CANCEL".

Here are the parcels being displayed:



The screenshot shows a window titled "Program 2" with a menu bar containing "File", "Insert", and "Report". The main content area displays the following data:

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