COMP 3602

C# Application Development

Week Eight - Online

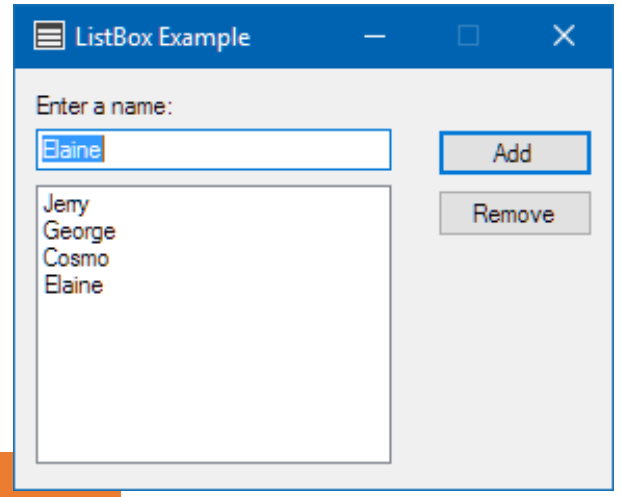
This Week's Learning Outcomes

Control Class

Control is the common base class for GUI Controls including the Form class. Several common Properties, Methods and Events are defined in and inherited from Control.

Properties	Methods	Events
Name	Select	Click
Text	GetHandle	DoubleClick
Enabled	Others…	MouseEnter
Size		MouseLeave
Location		MouseMove
Visible		Resize
BackColor		KeyUp
ForeColor		KeyDown
Others…		KeyPress
		Others…

ListBox Class



The ListBox is essentially an “ArrayList with a Viewport”. An internal collection is exposed through the Items property. It stores items as type object so casting is required when accessing ListBox elements.

Properties	Methods
Name	Items.Add
Items	Items.Remove
SelectedItem	Items.RemoveAt
SelectedIndex	Items.Clear
SelectionMode	BeginUpdate
DisplayMember	EndUpdate

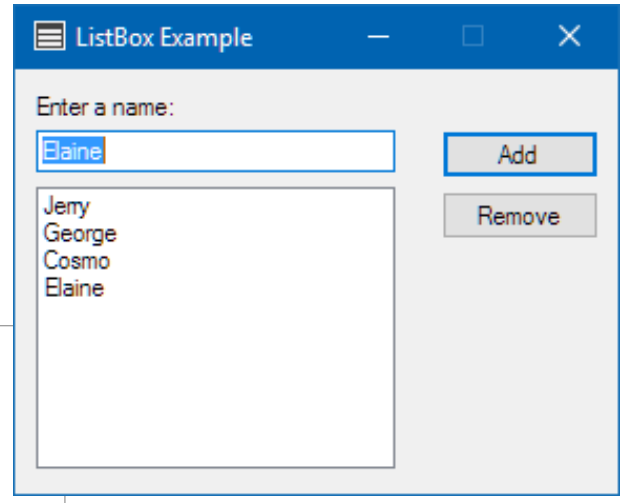
Events
SelectedIndexChanged*

→ This is the default event for the ListBox and should be used instead of the Click event to detect Item selection because it will work with the Mouse and the Keyboard

ListBox Class - Add

Add an object to a ListBox by calling Add on its Items collection. This collection stores objects as type object, like an ArrayList so Add with accept any datatype.

```
19          private void buttonAdd_Click(object sender, EventArgs e)
20          {
21              // First, some input validation.
22              if (textBoxName.Text == string.Empty)
23              {
24                  MessageBox.Show("Enter a name first"
25                                  , "Add Name"
26                                  , MessageBoxButtons.OK
27                                  , MessageBoxIcon.Information);
28              }
29              else
30              {
31                  // Add name from the TextBox into the ListBox. By default,
32                  // the new ListBox item will be added to the end of the list.
33                  listBoxNames.Items.Add(textBoxName.Text);
34              }
35
36              // Send the input focus to the TextBox control and select any text it has.
37              textBoxName.Select();
38              textBoxName.SelectAll();
39          }
```

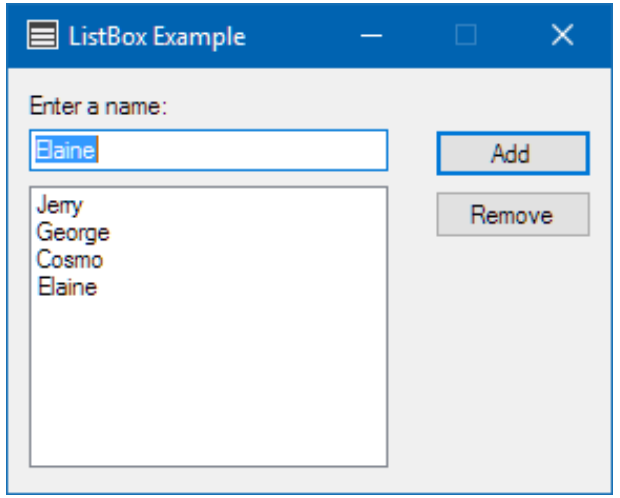


Items is an internal collection which is similar to an ArrayList. The Add method will accept any data type.

ListBox Class - RemoveAt

```
41          private void buttonRemove_Click(object sender, EventArgs e)
42          {
43              int selectedIndex = listBoxNames.SelectedIndex;
44
45              // First, some input validation. In this case, the
46              // input is the selected list index. An index
47              // of -1 indicates that nothing is selected.
48              if (listBoxNames.Items.Count == 0)
49              {
50                  MessageBox.Show("The list is empty"
51                                  , "Remove Name"
52                                  , MessageBoxButtons.OK
53                                  , MessageBoxIcon.Information);
54              }
55              else if (selectedIndex == -1)
56              {
                    MessageBox.Show("Select a name to remove first"
                                    , "Remove Name"
                                    , MessageBoxButtons.OK
                                    , MessageBoxIcon.Information);
61              }
62              else
63              {
64                  // Remove the selected item from the list.
65                  listBoxNames.Items.RemoveAt(selectedIndex);
66              }
67          }
```

SelectedIndex = -1 if nothing is selected

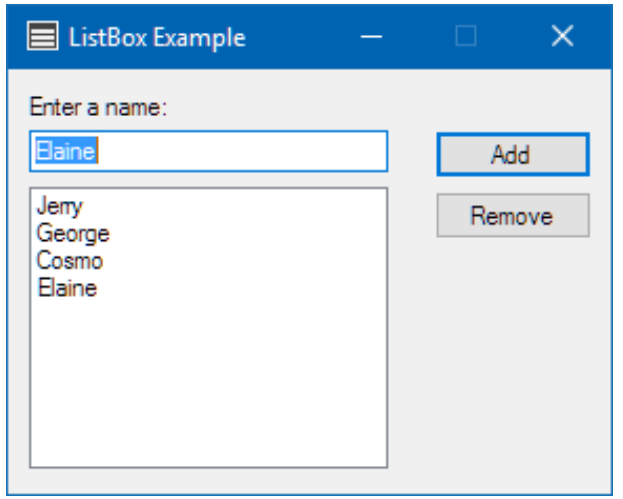


Remove an object from a ListBox by calling RemoveAt with the SelectedIndex as a parameter. This will not destroy the object. The object will live on if the ListBox was not the only reference to it.

ListBox Class - Remove

```
41          private void buttonRemove_Click(object sender, EventArgs e)
42          {
43              object selectedItem = listBoxNames.SelectedItem;
44
45              // First, some input validation. In this case, the
46              // input is the selected list index. An index
47              // of -1 indicates that nothing is selected.
48              if (listBoxNames.Items.Count == 0)
49              {
50                  MessageBox.Show("The list is empty"
51                                  , "Remove Name"
52                                  , MessageBoxButtons.OK
53                                  , MessageBoxIcon.Information);
54              }
55              else if (listBoxNames.SelectedIndex == -1)
56              {
                    MessageBox.Show("Select a name to remove first"
                                    , "Remove Name"
                                    , MessageBoxButtons.OK
                                    , MessageBoxIcon.Information);
61              }
62              else
63              {
64                  // Remove the selected item from the list.
65                  listBoxNames.Items.Remove(selectedItem);
66              }
67          }
```

SelectedIndex = -1 if nothing is selected



You can also remove an object from a ListBox by calling Remove with the SelectedItem as a parameter. This will not destroy the object. The object will live on if the ListBox was not the only reference to it.

ListBox Class – Object Display

Each item in a ListBox is always displayed as a string

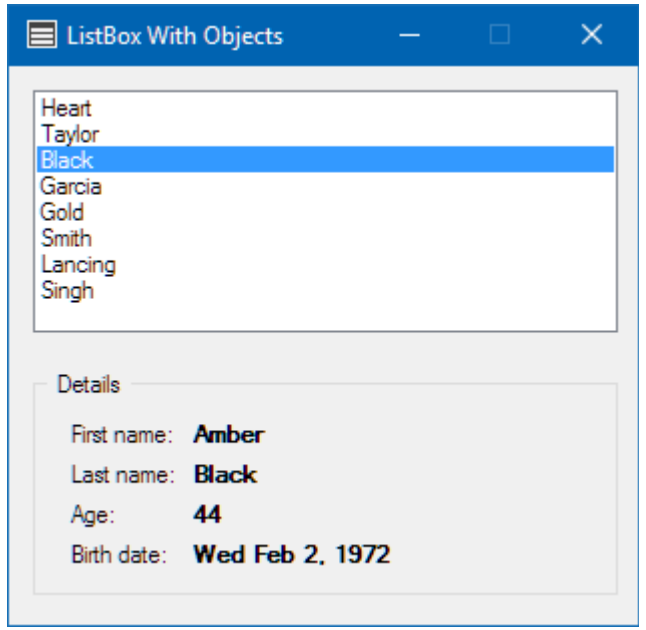
Often, each item needs to be associated with more than a simple string, such as an inventory item or a sales transaction or a person.

When the user selects one of the items from the ListBox, how can we access the corresponding business object?

It is possible to add any object to a ListBox instead of a string.

The ListBox has two ways to display a string associated with that object:
The ListBox's DisplayMember property
The object's ToString method (which can be overridden)

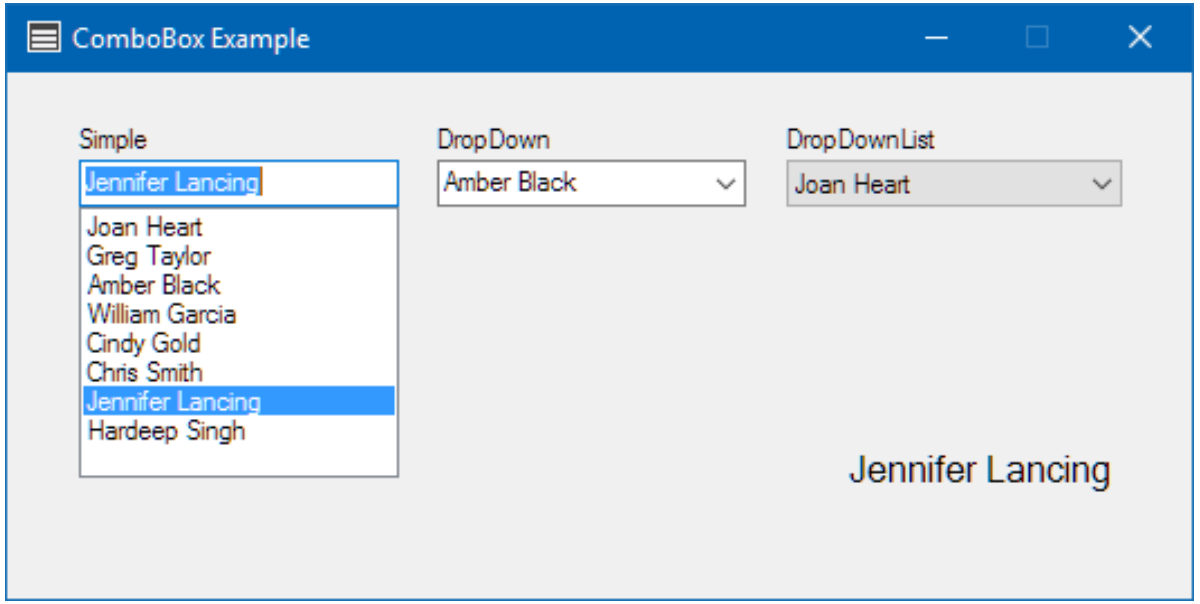
Assign a collection to the ListBox's DataSource property to display its contents

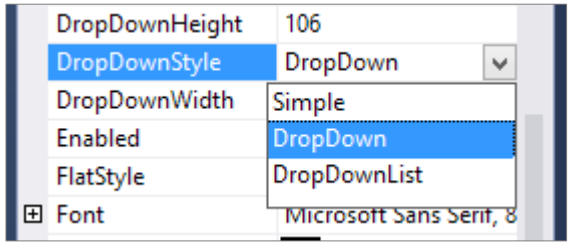



```
29              // Note: The DisplayMember is set to a non-empty string then it is used
30              //       instead of the object's ToString method.
31              listBoxNames.DisplayMember = "LastName";
```

ComboBox Class

The ComboBox shares most of the Properties of the ListBox with additional rendering Properties.





Simple

List is always visible, select one of the options or enter your own value.

DropDown

List closes up, select one of the options or enter your own value.

DropDownList

List closes up, selection is limited to the available options only.