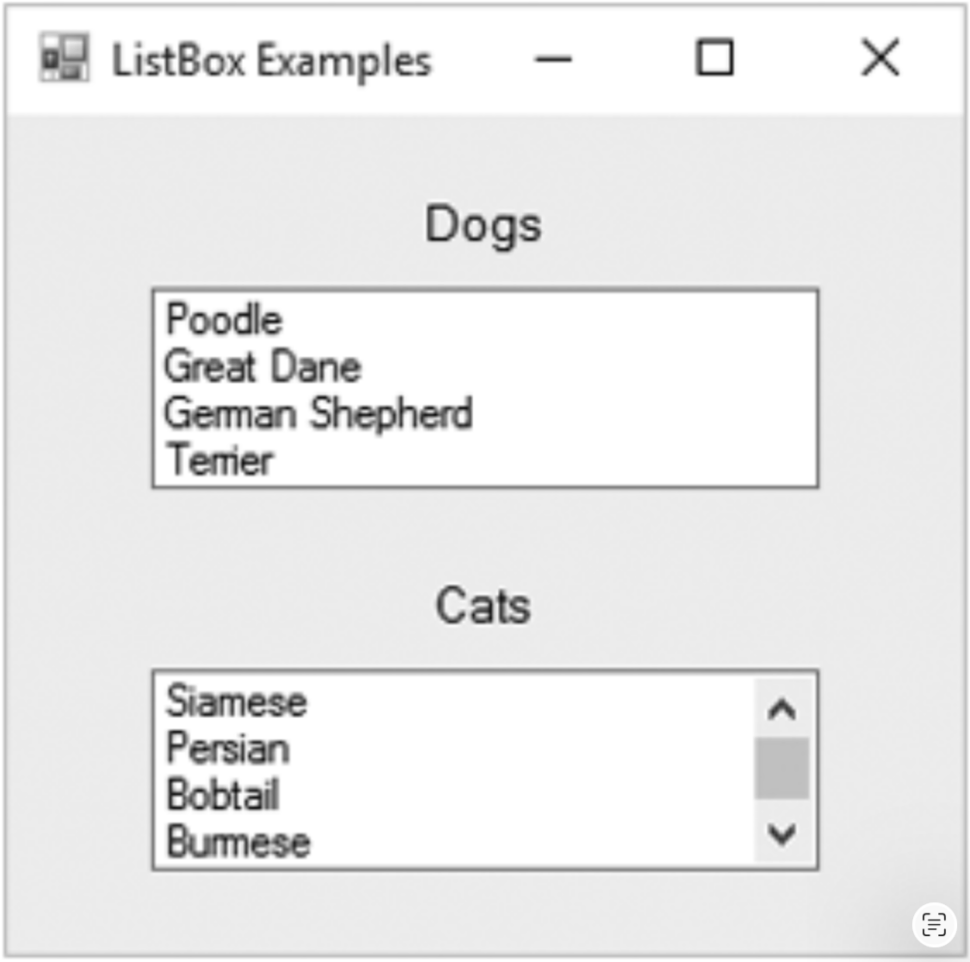
List Boxes

List boxes display a list of items and allow the user to select an item from the list.

Figure 5-1 shows a form, at run time, with two ListBox controls. At run time, the user may select one of the items, causing the item to appear selected.

Figure 5-1 ListBox examples



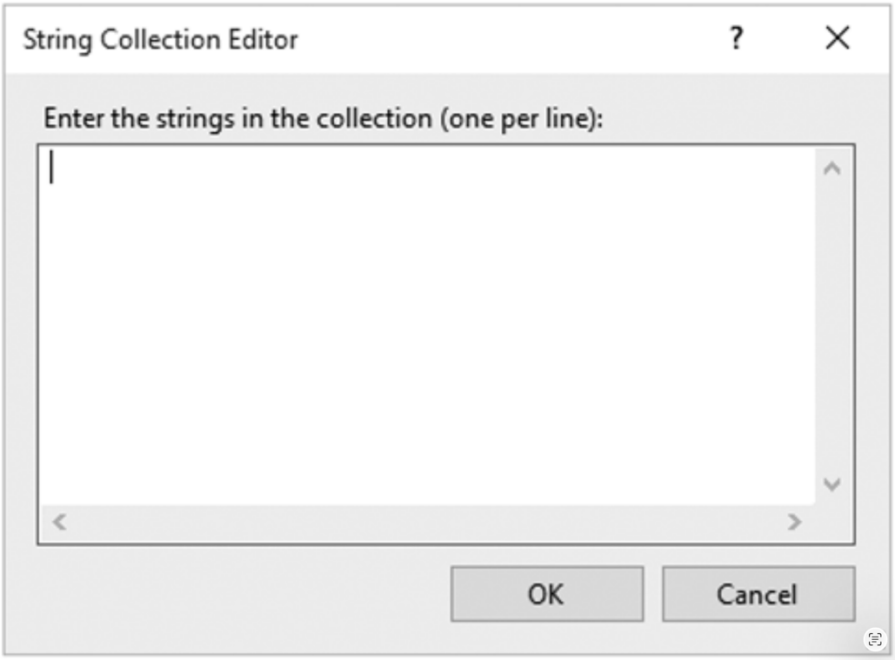
A scroll bar appears when a ListBox contains more items than can be displayed in the space provided.

Once you create a ListBox control, you add items to its **Items property**. The items that you add to a ListBox's Items property are displayed in the ListBox.

To store values in the Items property at design time, follow these steps:

1- Select the ListBox control in the Designer window.
2- In the Properties window, the setting for the **Items property** is displayed as (Collection). When you select the Items property, an ellipses button () appears.
3- Click the ellipses button. The String Collection Editor dialog box appears, as shown in Figure 5-2.
4- Type the values that are to appear in the ListBox into the String Collection Editor dialog box. Type each value on a separate line by pressing the Enter key after each entry.
5- When you have entered all the values, click the OK button.

Figure 5-2 The String Collection Editor dialog box.



The SelectedItem Property

When the user selects an item in a ListBox, the item is stored in the ListBox's SelectedItem property. For example, suppose an application has a ListBox control named fruitListBox and a string variable named selectedFruit. The fruitListBox control contains the items Apples, Pears, and Bananas. If the user has selected Pears, the following statement assigns the string "Pears" to the variable selectedFruit:

```
selectedFruit = fruitListBox.SelectedItem.ToString();
```

- Notice that you have to call the SelectedItem property's **ToString** method to retrieve the value as a string.
- An exception (Error) will occur if you try to get the value of a ListBox's SelectedItem property when no item is selected in the ListBox.
- The items that are stored in a ListBox each have an index. The index is simply a number that identifies the item's position in the ListBox. The first item has the index 0, the next has the index 1, and so forth. The last index value is n−1, where n is the number of items in the ListBox. When the user selects an item in a ListBox, the item's index is stored in the ListBox's **SelectedIndex property**. If no item is selected in the ListBox, the SelectedIndex property is set to −1.
- make sure the SelectedIndex property is not set to −1 before trying to read the SelectedItem property. Here is an example:

```
if (fruitListBox.SelectedIndex != -1)
{
    selectedFruit = fruitListBox.SelectedItem.ToString();
}
```

More about ListBoxes

ListBox controls have various methods and properties that you can use in code to manipulate the ListBox's contents.

In this chapter, we use ListBox controls to display output.

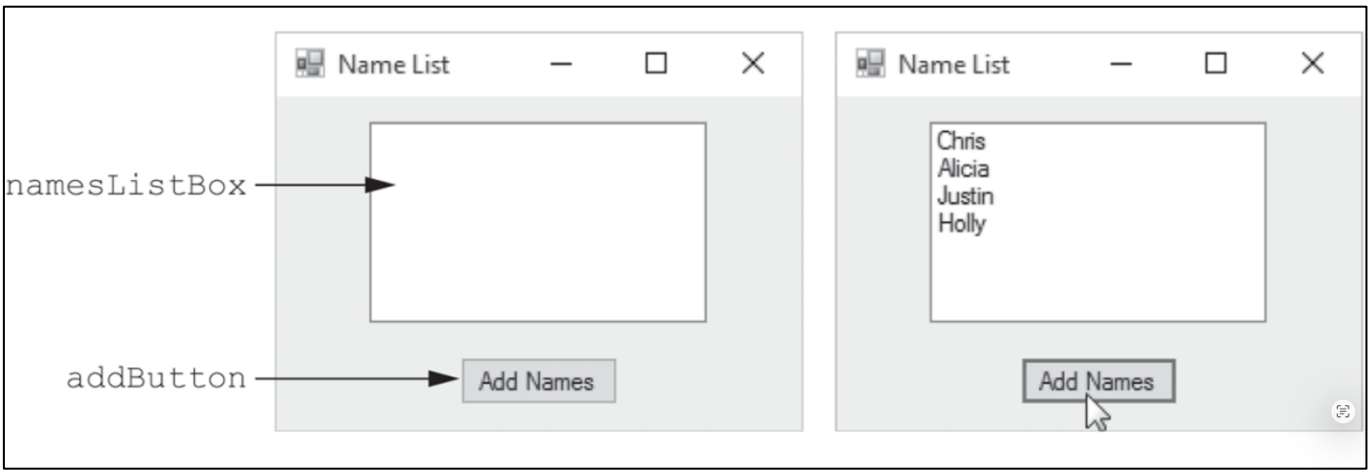
You can write code that adds items to a ListBox control at run time. To add an item to a ListBox control with code, you call the control's Items.Add method.

```
ListBoxName.Items.Add(Item);
```

ListBoxName is the name of the ListBox control. Item is the value to be added to the Items property.

Figure 5-3 shows the application's form at run time. As shown in the image on the left, the ListBox's name is nameListBox and the Button control's name is addButton. At run time, when you click the addButton control, the names shown in the image on the right are added to the nameListBox control.

Figure 5-3 The Name List application

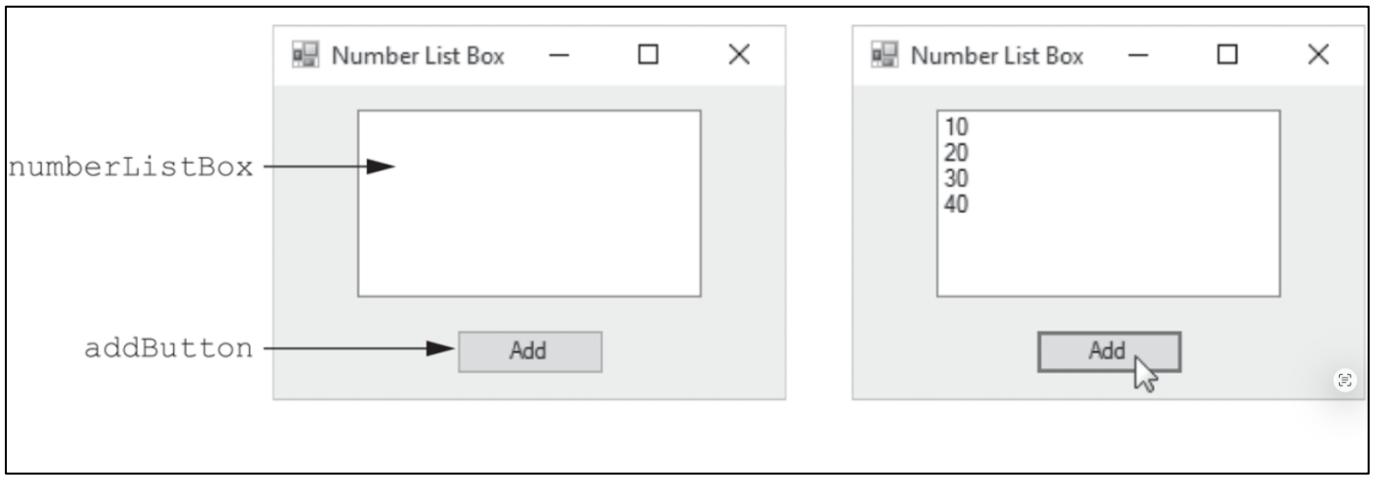


Here is the code for the addButton_Click event handler:

```
1        private void addButton_Click(object sender, EventArgs e)
2        {
3            namesListBox.Items.Add("Chris");
4            namesListBox.Items.Add("Alicia");
5            namesListBox.Items.Add("Justin");
6            namesListBox.Items.Add("Holly");
7        }
```

Figure 5-4 shows the application's form. As shown in the image on the left, the ListBox's name is numberListBox and the Button control's name is addButton. At run time, when you click the addButton control, the numbers shown in the image on the right are added to the numberListBox control.

Figure 5-4 The Number List application



Here is the code for the addButton_Click event handler:

```
1       private void addButton_Click(object sender, EventArgs e)
2       {
3           numberListBox.Items.Add(10);
4           numberListBox.Items.Add(20);
5           numberListBox.Items.Add(30);
6           numberListBox.Items.Add(40);
7       }
```

The Items.Count Property

ListBox controls have an **Items.Count property** that reports the number of items stored in the ListBox. If the ListBox is empty, the Items.Count property equals 0. For example, assume an application has a ListBox control named employeesListBox. The following if statement displays a message box if there are no items in the ListBox:

```
if (employeesListBox.Items.Count == 0)
{
    MessageBox.Show("There are no items in the list!");
}
```

The Items.Clear Method

ListBox controls have an Items.Clear method that erases all the items in the Items property. Here is the method's general format:

```
ListBoxName.Items.Clear();
```