Class Hierarchy with two children class

```
/*
Learning C#
by Jesse Liberty

Publisher: O'Reilly
ISBN: 0596003765
*/
using System;

class Window
{
// constructor takes two integers to
// fix location on the console
public Window(int top, int left)
{
this.top = top;
this.left = left;
}

// simulates drawing the window
public virtual void DrawWindow()
{
Console.WriteLine(“Window: drawing Window at {0}, {1}”,
top, left);
}

// these members are protected and thus visible
// to derived class methods. We'll examine this
// later in the chapter
protected int top;
protected int left;

}
```

```
// ListBox derives from Window
class ListBox : Window
{
// constructor adds a parameter
public ListBox(
int top,
int left,
string contents):
base(top, left) // call base constructor
{

listBoxContents = contents;
}

// an overridden version (note keyword) because in the
// derived method we change the behavior
public override void DrawWindow()
{
base.DrawWindow(); // invoke the base method
Console.WriteLine (“Writing string to the listbox: {0}”,
listBoxContents);
}

private string listBoxContents; // new member variable
}

class Button : Window
{
public Button(
int top,
int left):
base(top, left)
{
}

// an overridden version (note keyword) because in the
// derived method we change the behavior
public override void DrawWindow()
{
Console.WriteLine(“Drawing a button at {0}, {1}
“,
top, left);
}
}

public class TesterClassArray1
{
static void Main()
{
Window win = new Window(1,2);
ListBox lb = new ListBox(3,4,”Stand alone list box”);
Button b = new Button(5,6);
win.DrawWindow();
lb.DrawWindow();
b.DrawWindow();

Window[] winArray = new Window[3];
winArray[0] = new Window(1,2);
winArray[1] = new ListBox(3,4,”List box in array”);
winArray[2] = new Button(5,6);

for (int i = 0;i < 3; i++) { winArray[i].DrawWindow(); } } }
[/csharp]
```