

Lab 12 by Bryant Tunbutr

The screenshot shows a window titled "Your Pet" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains the following elements:

- Select Pet:** A group box containing two radio buttons: "Dog" (selected) and "Cat".
- Name:** A text input field.
- Color:** A text input field.
- Buttons:** Two buttons labeled "New Pet" and "Display Current Info".
- Current Info:** A large text area for displaying information.
- Exit:** A button at the bottom right.

All input fields are currently empty.

This screenshot shows the same "Your Pet" window after the "Display Current Info" button has been clicked. The state is as follows:

- Select Pet:** "Dog" remains selected.
- Name:** The text "barkie" has been entered.
- Color:** The text "red" has been entered.
- Buttons:** The "Display Current Info" button is now highlighted with a blue dashed border, indicating it was the last active control.
- Current Info:** The text area now displays "Your red dog barkie says Woof!".
- Exit:** The button remains at the bottom right.

Your Pet

Select Pet

☒ Dog

☐ Cat

Name

Color

New Pet

Display Current Info

Current Info

Exit

Your Pet

Select Pet

☒ Dog

☐ Cat

Name

Color

New Pet

Display Current Info

Current Info

Exit

Your Pet

Select Pet

☐ Dog

☒ Cat

Name

Color

New Pet

Display Current Info

Current Info

Exit

Your Pet

Select Pet

☐ Dog

☒ Cat

Name

Color

New Pet

Display Current Info

Current Info

Exit

Source code for BaseForm.cs

```
/*
 * Project:                BTunbutrLab12
 * Programmer:            Bradley/Millspaugh
 * Date:                  Nov 29
 * Description:           Use classes, methods, over riding to
 *                         display pet information
 * * I certify that the code below is my own work.
 */

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication22
{
    public partial class BaseForm : Form
    {
        public BaseForm()
        {
            InitializeComponent();
        }

        public virtual void okButton_Click(object sender, EventArgs e)
        {
            // Close the form.
            this.Close();
        }
    }
}
```

Source code for petsForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication22
{
    public partial class petsForm : Form
    {
        public petsForm()
        {
            InitializeComponent();

            // Display info in text box
            public void displayInfoButton_Click(object sender, EventArgs e)
            {
                // Use info about pet name and color
                string petName = nameTextBox.Text;
                string petColor = colorTextBox.Text;

                // Call dog speak method
                if (dogRadioButton.Checked == true)
                {
                    displayInfoTextBox.Text = "Your " + petColor + " dog " +
                        petName + " says " + dog.Speak;
                }

                // Call cat speak method
                if (catRadioButton.Checked == true)
                {
                    displayInfoTextBox.Text = "Your " + petColor + " cat " +
                        petName + " says " + cat.Speak;
                }
            }

            private void petsForm_Load(object sender, EventArgs e)
            {
                // Initialize the form.
            }

            private void newPetButton_Click(object sender, EventArgs e)
            {
                // clear textboxes, reset to default radio button
                displayInfoTextBox.Text = "";
                nameTextBox.Text = "";
                colorTextBox.Text = "";
                dogRadioButton.Checked = true;
            }
        }
    }
}
```

```
}  
  
private void button1_Click(object sender, EventArgs e)  
{  
    // Close the form.  
    this.Close();  
}  
}
```

Source code for pets.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace WindowsFormsApplication22
{
    // create a pets class.
    class pets
    {
        // create a pets class.
        protected string petName, petColor;

        // name property of the pets class.
        public string Name
        {
            get
            {
                return petName;
            }
            private set
            {
                petName = value;
            }
        }

        // Color property of the pets class.
        public string Color
        {
            get
            {
                return petColor;
            }
            private set
            {
                petColor = value;
            }
        }

        // create a pets speak method.
        // use static to allow it to pass to the other form.
        public static string Speak
        {
            get
            {
                return "Speaking!";
            }
            private set
            {
                Speak = value;
            }
        }
    }
}
```

```

class dog : pets
{
    public dog()
    {
    }

    // override the pets speak method.
    // use new to override
    new public static string Speak
    {
        get
        {
            return "Woof!";
        }
        private set
        {
            Speak = value;
        }
    }
}

// override the pets speak method.
// use new to override
class cat : pets
{
    public cat()
    {
    }

    new public static string Speak
    {
        get
        {
            return "Meow!";
        }
        private set
        {
            Speak = value;
        }
    }
}
}

```


Why wouldn't the compiler complain about the fact that a Pet variable is referring to a Dog or Cat object? What is the name of this feature?

The dog and cat class and their objects are based on the pet class

This is known as inheritance

The dog and cat classes are derived

The derived class has access to public and protected data members of the base class

The good part is the protected keyword allows elements to accessible within their own class, AND any class derived from the class.

Thus the pet class has access to the dog and cat elements