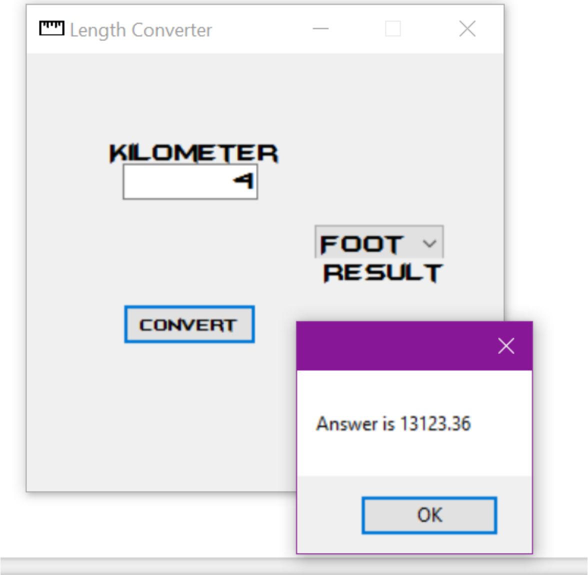
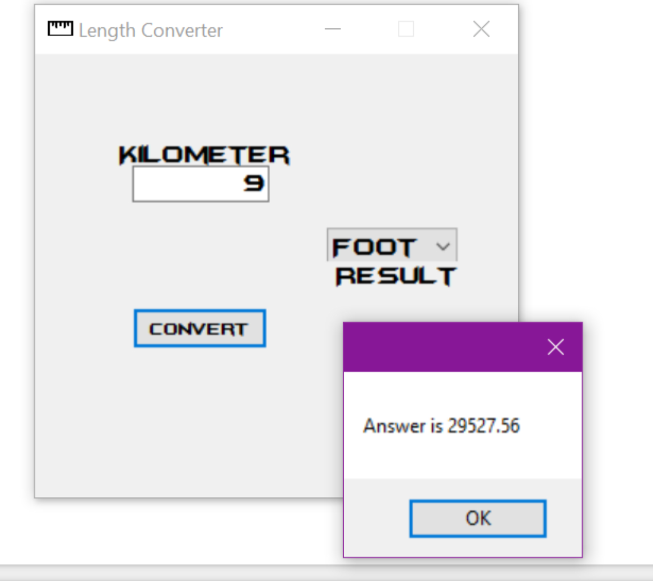
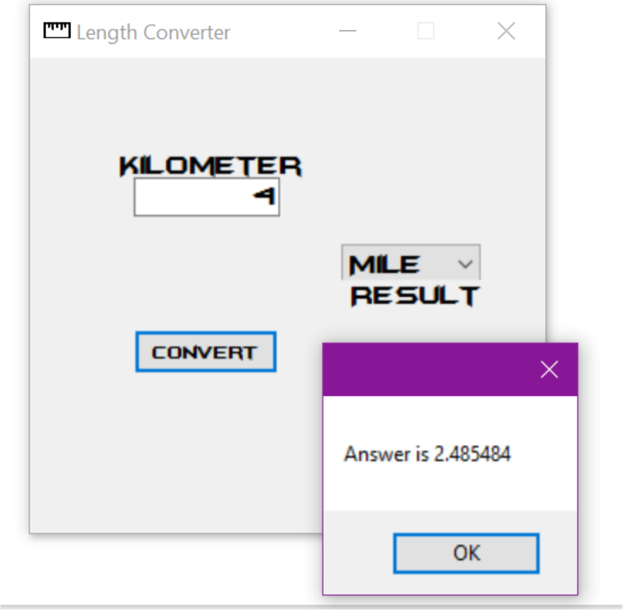
PART 1



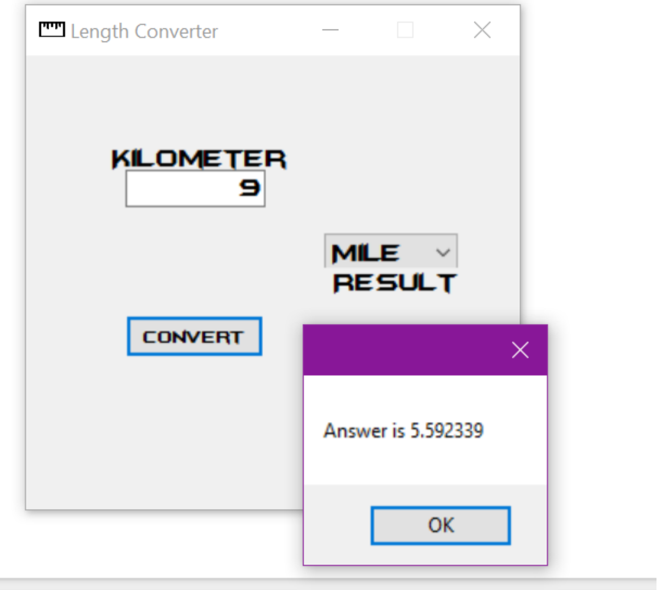
Answer is 13123.36



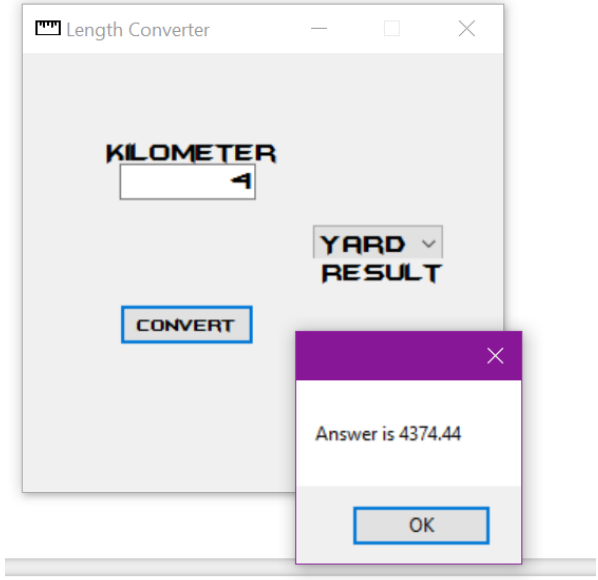
Answer is 29527.56



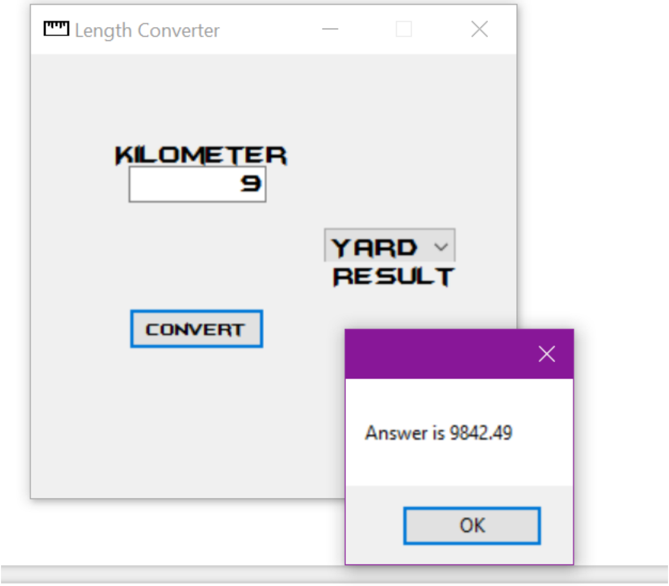
Answer is 2.485484



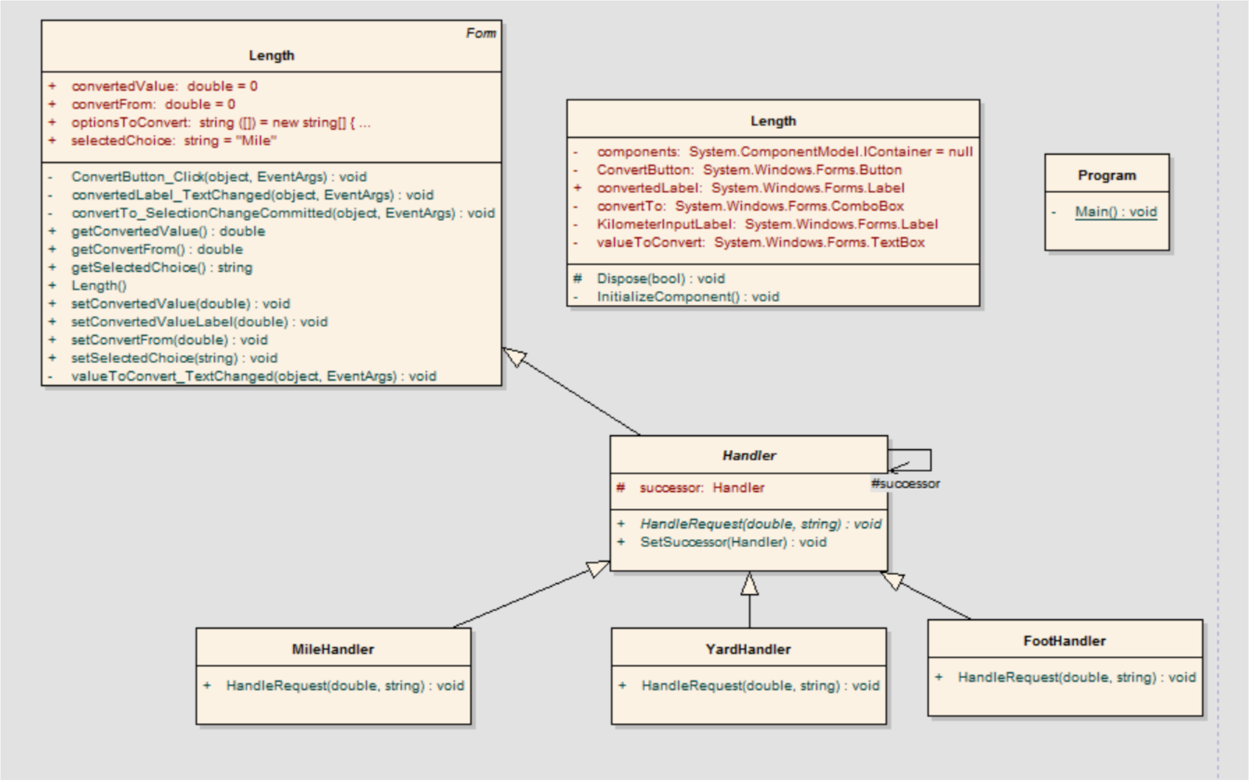
Answer is 5.592339



Answer is 4374.44

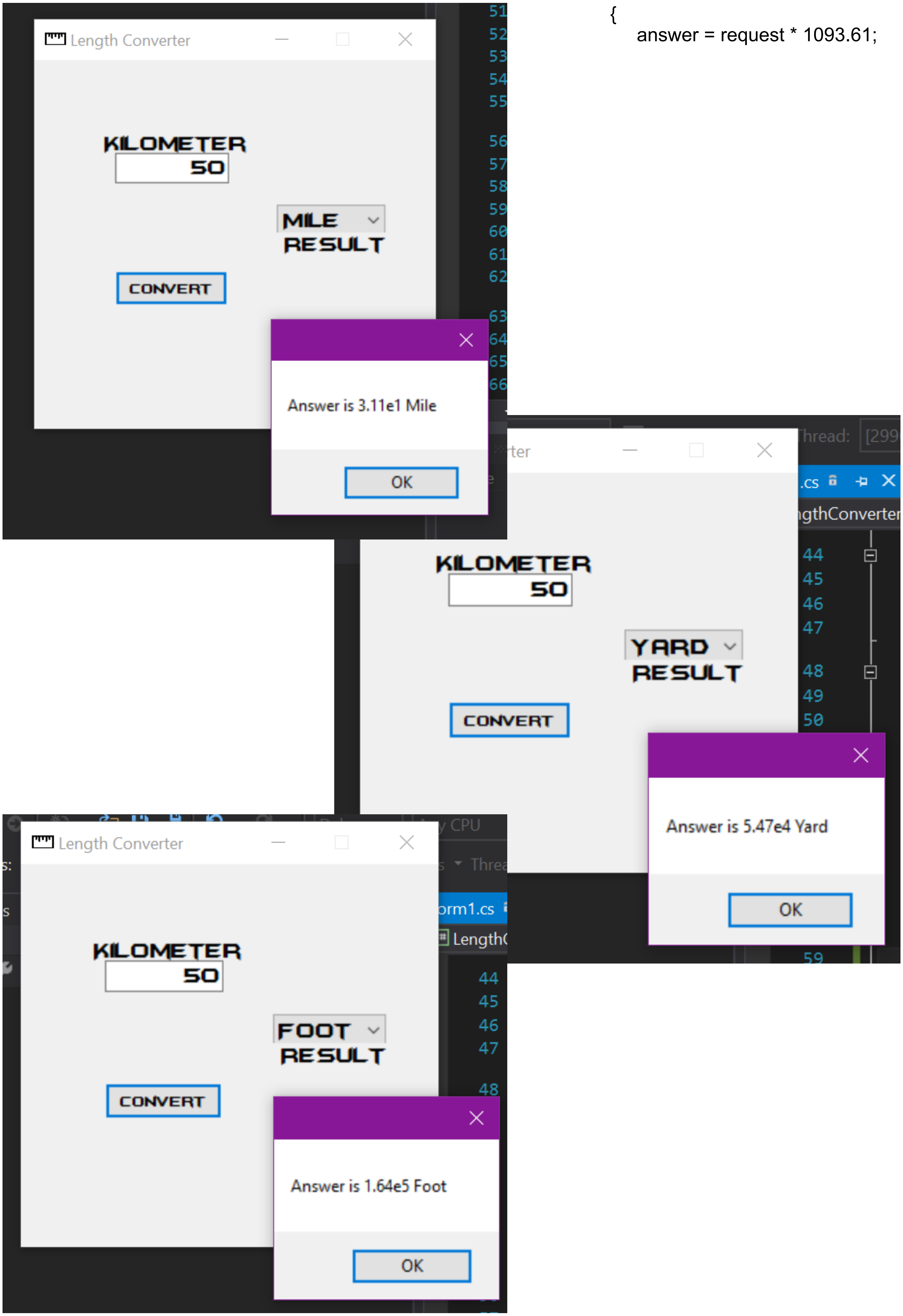


Answer is 9842.49


Form
Length
+ convertedValue: double = 0
+ convertFrom: double = 0
+ optionsToConvert: string ([]) = new string[] { ...
+ selectedChoice: string = "Mile"
- ConvertButton_Click(object, EventArgs) : void
- convertedLabel_TextChanged(object, EventArgs) : void
- convertTo_SelectionChangeCommitted(object, EventArgs) : void
+ getConvertedValue() : double
+ getConvertFrom() : double
+ getSelectedChoice() : string
+ Length()
+ setConvertedValue(double) : void
+ setConvertedValueLabel(double) : void
+ setConvertFrom(double) : void
+ setSelectedChoice(string) : void
- valueToConvert_TextChanged(object, EventArgs) : void
Length
- components: System.ComponentModel.IContainer = null
- ConvertButton: System.Windows.Forms.Button
+ convertedLabel: System.Windows.Forms.Label
- convertTo: System.Windows.Forms.ComboBox
- KilometerInputLabel: System.Windows.Forms.Label
- valueToConvert: System.Windows.Forms.TextBox
# Dispose(bool) : void
- InitializeComponent() : void
Program
- Main() : void
Handler
# successor: Handler
#successor
+ HandleRequest(double, string) : void
+ SetSuccessor(Handler) : void
MileHandler
+ HandleRequest(double, string) : void
YardHandler
+ HandleRequest(double, string) : void
FootHandler
+ HandleRequest(double, string) : void

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

namespace LengthConverter
{
    public partial class Length : Form
    {
        public string[] optionsToConvert = new string[] { "Mile", "Yard", "Foot" };
        public double convertFrom = 0;
        public double convertedValue = 0;
        public string selectedChoice = "Mile";

        public Length()
        {
            InitializeComponent();
            convertTo.Items.AddRange(optionsToConvert);
            convertTo.DataSource = optionsToConvert;
            convertTo.DropDownStyle = ComboBoxStyle.DropDownList;
            convertTo.SelectionStart = 1;
        }
        public void setConvertFrom(double value)
        {
            convertFrom = value;
        }
        public double getConvertFrom()
        {
            return convertFrom;
        }
        public void setConvertedValue(double value)
        {
            convertedValue = value;
        }
        public double getConvertedValue()
        {
            return convertedValue;
        }
        public void setSelectedChoice(string choice)
        {
            selectedChoice = choice;
        }
        public string getSelectedChoice()
        {
            if (convertTo.SelectedItem != null)
            {
                return convertTo.Text.ToString();
            }
            return selectedChoice;
        }
        public void setConvertedValueLabel(double answer)
        {
            MessageBox.Show("Answer is " + answer.ToString());

        }

        private void ConvertButton_Click(object sender, EventArgs e)
        {
            Handler Mile = new MileHandler();
            Handler Yard = new YardHandler();
            Handler Foot = new FootHandler();
            Mile.SetSuccessor(Yard);
            Yard.SetSuccessor(Foot);

            Mile.HandleRequest(convertFrom,selectedChoice);
        }

        private void valueToConvert_TextChanged(object sender, EventArgs e)
        {
            convertFrom = Convert.ToDouble(valueToConvert.Text);
        }

        private void convertTo_SelectionChangeCommitted(object sender, EventArgs e)
        {
            selectedChoice = convertTo.GetItemText(convertTo.SelectedItem);
        }

        private void convertedLabel_TextChanged(object sender, EventArgs e)
        {
            this.convertedLabel.Update();
        }
    }

    abstract class Handler : Length
    {
        protected Handler successor;
        public void SetSuccessor(Handler successor)
        {
            this.successor = successor;
        }
        public abstract void HandleRequest(double request, string choice);
    }

    class MileHandler : Handler
    {
        public override void HandleRequest(double request, string choice)
        {
            double answer = 0;
            if (string.Equals(choice,"Mile"))
            {
                answer = request * 0.621371;
                setConvertedValueLabel(answer);
            }
            else if(successor != null)
            {
                successor.HandleRequest(request, choice);
            }
        }
    }

    class YardHandler : Handler
    {
        public override void HandleRequest(double request, string choice)
        {
            double answer = 0;
            if (choice.Equals("Yard"))
```

```
{
    answer = request * 1093.61;
```

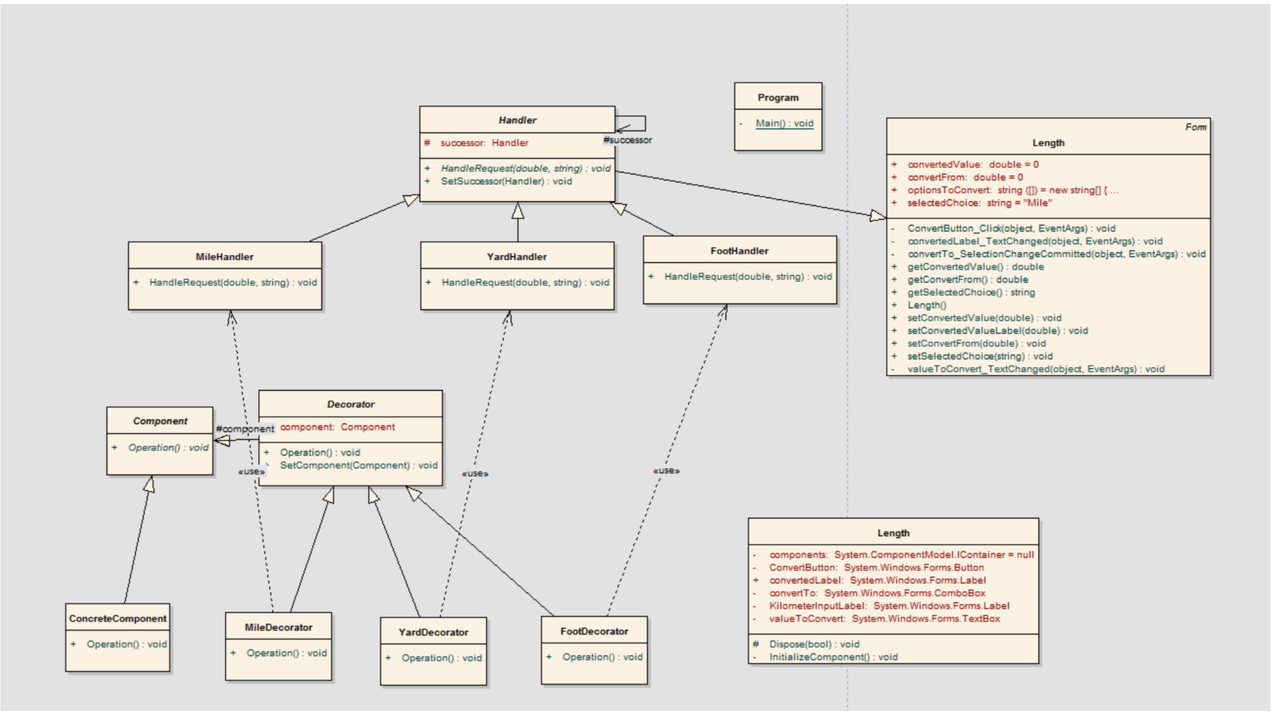


Answer is 3.11e1 Mile

Answer is 5.47e4 Yard

Answer is 1.64e5 Foot

```
                setConvertedValueLabel(answer);
            }
            else if (successor != null)
            {
                successor.HandleRequest(request, choice);
            }
        }
    }

    class FootHandler : Handler
    {
        public override void HandleRequest(double request, string choice)
        {
            double answer = 0;
            if (choice.Equals("Foot"))
            {
                answer = request * 3280.84;
                setConvertedValueLabel(answer);
            }
            else if (successor != null)
            {
                successor.HandleRequest(request, choice);
            }
        }
    }
}
```


Handler
# successor: Handler
+ HandleRequest(double, string) : void
+ SetSuccessor(Handler) : void
#successor
Program
- Main() : void
Form
Length
+ convertedValue: double = 0
+ convertFrom: double = 0
+ optionsToConvert: string ([]) = new string[] { ...
+ selectedChoice: string = "Mile"
- ConvertButton_Click(object, EventArgs) : void
- convertedLabel_TextChanged(object, EventArgs) : void
- convertTo_SelectionChangeCommitted(object, EventArgs) : void
+ getConvertedValue() : double
+ getConvertFrom() : double
+ getSelectedChoice() : string
+ Length()
+ setConvertedValue(double) : void
+ setConvertedValueLabel(double) : void
+ setConvertFrom(double) : void
+ setSelectedChoice(string) : void
- valueToConvert_TextChanged(object, EventArgs) : void
MileHandler
+ HandleRequest(double, string) : void
YardHandler
+ HandleRequest(double, string) : void
FootHandler
+ HandleRequest(double, string) : void
Component
+ Operation() : void
#component
Decorator
component: Component
+ Operation() : void
SetComponent(Component) : void
«use»
«use»
«use»
ConcreteComponent
+ Operation() : void
MileDecorator
+ Operation() : void
YardDecorator
+ Operation() : void
FootDecorator
+ Operation() : void
Length
- components: System.ComponentModel.IContainer = null
- ConvertButton: System.Windows.Forms.Button
+ convertedLabel: System.Windows.Forms.Label
- convertTo: System.Windows.Forms.ComboBox
- KilometerInputLabel: System.Windows.Forms.Label
- valueToConvert: System.Windows.Forms.TextBox
# Dispose(bool) : void
- InitializeComponent() : void

PART 2

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

namespace LengthConverter
{
    public partial class Length : Form
    {
        public string[] optionsToConvert = new string[] { "Mile", "Yard", "Foot" };
        public double convertFrom = 0;
        public double convertedValue = 0;
        public string selectedChoice = "Mile";

        public Length()
        {
            InitializeComponent();
            convertTo.Items.AddRange(optionsToConvert);
            convertTo.DataSource = optionsToConvert;
            convertTo.DropDownStyle = ComboBoxStyle.DropDownList;
            convertTo.SelectionStart = 1;
        }
        public void setConvertFrom(double value)
        {
            convertFrom = value;
        }
        public double getConvertFrom()
        {
            return convertFrom;
        }
        public void setConvertedValue(double value)
        {
            convertedValue = value;
        }
        public double getConvertedValue()
        {
            return convertedValue;
        }
        public void setSelectedChoice(string choice)
        {
            selectedChoice = choice;
        }
        public string getSelectedChoice()
        {
            if (convertTo.SelectedItem != null)
            {
                return convertTo.Text.ToString();
            }
            return selectedChoice;
        }
        public void setConvertedValueLabel(double answer)
        {
            Math.Round(answer, 2);
            MessageBox.Show("Answer is " + string.Format("{0:#.0#e-0}", answer) +" "+
convertTo.SelectedItem.ToString());

        }

        private void ConvertButton_Click(object sender, EventArgs e)
        {
            Handler Mile = new MileHandler();
            Handler Yard = new YardHandler();
            Handler Foot = new FootHandler();
            Mile.SetSuccessor(Yard);
            Yard.SetSuccessor(Foot);

            Mile.HandleRequest(convertFrom,selectedChoice);
        }

        private void valueToConvert_TextChanged(object sender, EventArgs e)
        {
            convertFrom = Convert.ToDouble(valueToConvert.Text);
        }

        private void convertTo_SelectionChangeCommitted(object sender, EventArgs e)
        {
            selectedChoice = convertTo.GetItemText(convertTo.SelectedItem);
        }

        private void convertedLabel_TextChanged(object sender, EventArgs e)
        {
            this.convertedLabel.Update();
        }
    }

    abstract class Handler : Length
    {
        protected Handler successor;
        public void SetSuccessor(Handler successor)
        {
            this.successor = successor;
        }
        public abstract void HandleRequest(double request, string choice);
    }

    class MileHandler : Handler
    {
        public override void HandleRequest(double request, string choice)
        {
            double answer = 0;
            if (string.Equals(choice,"Mile"))
            {
                answer = request * 0.621371;
                setConvertedValueLabel(answer);
            }
            else if(successor != null)
            {
                successor.HandleRequest(request, choice);
            }
        }
    }

    class YardHandler : Handler
    {
        public override void HandleRequest(double request, string choice)
        {
            double answer = 0;
            if (choice.Equals("Yard"))
            {
                answer = request * 1093.61;
                setConvertedValueLabel(answer);
            }
            else if (successor != null)
            {
                successor.HandleRequest(request, choice);
            }
        }
    }

    class FootHandler : Handler
    {
        public override void HandleRequest(double request, string choice)
        {
            double answer = 0;
            if (choice.Equals("Foot"))
            {
                answer = request * 3280.84;
                setConvertedValueLabel(answer);
            }
            else if (successor != null)
            {
                successor.HandleRequest(request, choice);
            }
        }
    }

    abstract class Component {
        public abstract void Operation();
    }

    class ConcreteComponent : Component
    {
        public override void Operation()
        {

            Console.WriteLine("ConcreteComponent.Operation()");
        }
    }

    abstract class Decorator : Component
```

```
    {
        protected Component component;

        public void SetComponent(Component component)
        {
            this.component = component;
        }

        public override void Operation()
        {
            if (component != null)
            {
                component.Operation();
            }
        }
    }

    class MileDecorator : Decorator
    {
        public override void Operation()
        {
            base.Operation();
        }
    }

    class YardDecorator : Decorator
    {
        public override void Operation()
        {
            base.Operation();
        }
    }

    class FootDecorator : Decorator
    {
        public override void Operation()
        {
            base.Operation();
        }
    }

}
```