Alex Lang

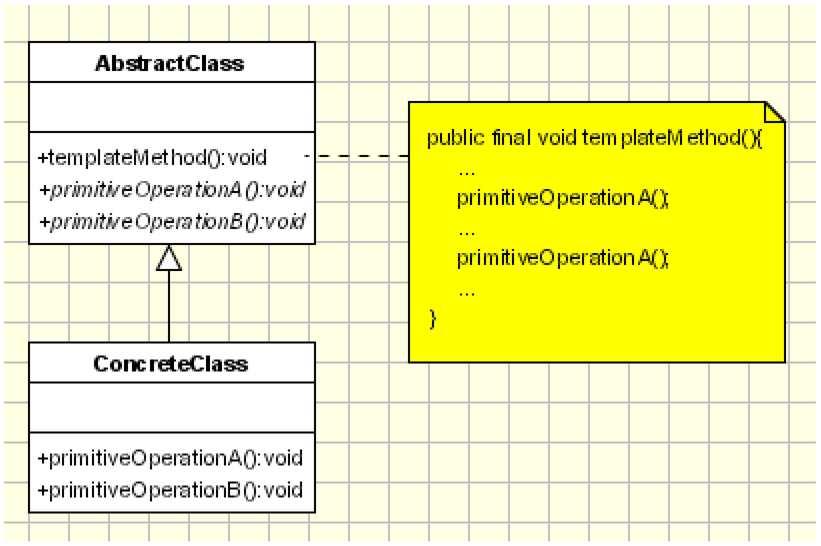
The Template Pattern

Design Patterns

12/1/16

For this assignment, we were told to do the Template Pattern. The template pattern defines the skeleton of an algorithm in an operation, deferring some steps to subclasses and lets subclasses redefine certain steps of an algorithm without letting them to change the algorithm's structure.

Here is the UML Diagram of the Template Pattern from OODesign.


AbstractClass
+templateMethod():void
+primitiveOperationA():void
+primitiveOperationB():void
ConcreteClass
+primitiveOperationA():void
+primitiveOperationB():void
public final void templateMethod(){
...
primitiveOperationA();
...
primitiveOperationA();
...
}

As OODesign states, the template pattern only requires two classes, the abstract class that contains various operations, and the concrete class which implements those methods from the abstract class.

For my application, I create a inch converter. The app converts inches to several other measurements of length.

First off, I created my abstract class, AbstractConvert.

```
public abstract class AbstractConvert
    {
        public abstract double toCM(double inches);
        public abstract double toM(double inches);
        public abstract double toFeet(double inches);
        public abstract double toYards(double inches);
    }
```

Since for this demo I will be converting inches to Cm, M, Feet, and yards, those are the necessary methods.

Next, I created my concrete Convert class.

```
public class Convert : AbstractConvert
    {
        public override double toCM(double inches)
        {
            double cm;
            cm = inches * 2.54;
            return cm;
        }

        public override double toFeet(double inches)
        {
            double feet;
            feet = inches / 12;
            return feet;
        }

        public override double toM(double inches)
        {
            double m;
            m = inches * 0.0254;
            return m;
        }

        public override double toYards(double inches)
        {
            double yards;
            yards = inches / 36;
            return yards;
        }


    }
```

This class defines all the various methods that can be done to the inches to convert them. Each of them takes in a inch double, and converts it to the relative number and returns the converted value.

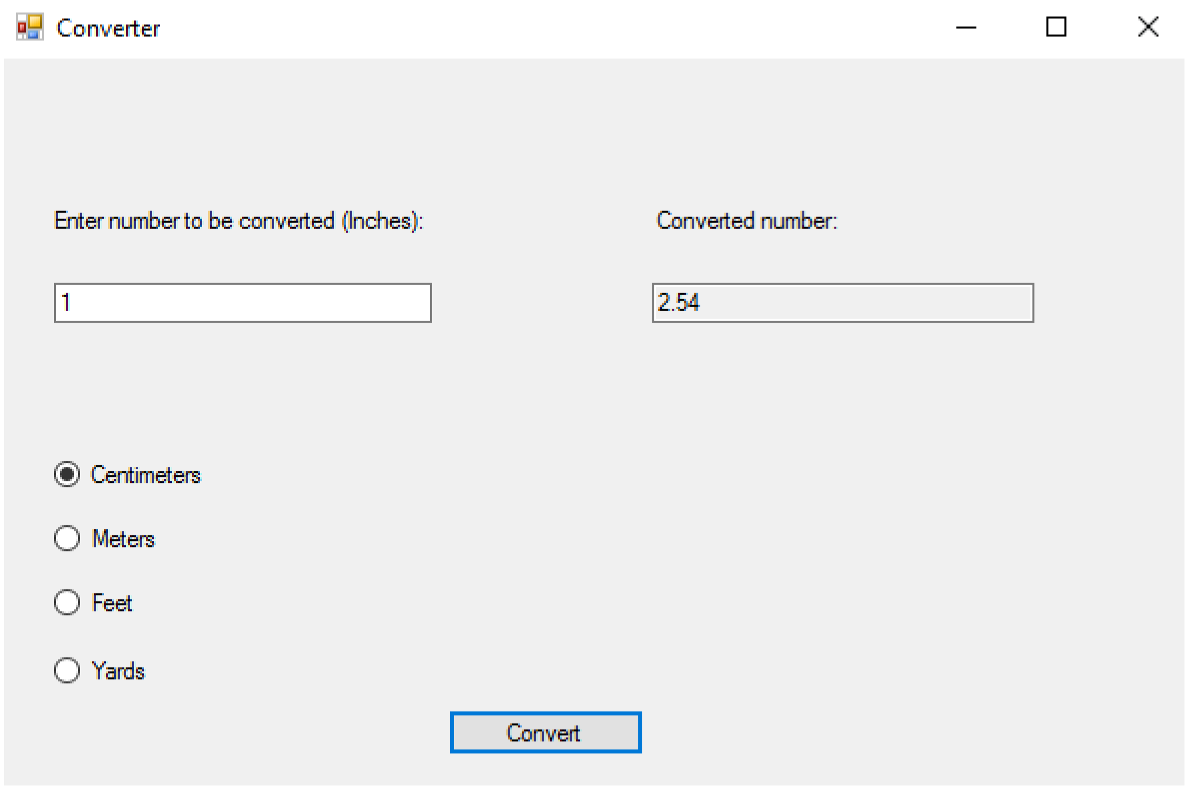
Finally, I put everything together in the main form.

```
public partial class Form1 : Form
    {
        AbstractConvert ac = new Convert();
        double inches;
        double result;

        public Form1()
        {
            InitializeComponent();
            cmBtn.Checked = true;
        }

        private void convertBtn_Click(object sender, EventArgs e)
        {
            inches = Double.Parse(inchesBox.Text);
            if (cmBtn.Checked)
            {
                result = ac.toCM(inches);
                convertBox.Text = result.ToString();
            }
            else if (meterBtn.Checked)
            {
                result = ac.toM(inches);
                convertBox.Text = result.ToString();
            }
            else if (feetBtn.Checked)
            {
                result = ac.toFeet(inches);
                convertBox.Text = result.ToString();
            }
            else if (yardBtn.Checked)
            {
                result = ac.toYards(inches);
                convertBox.Text = result.ToString();
            }
        }
    }
```

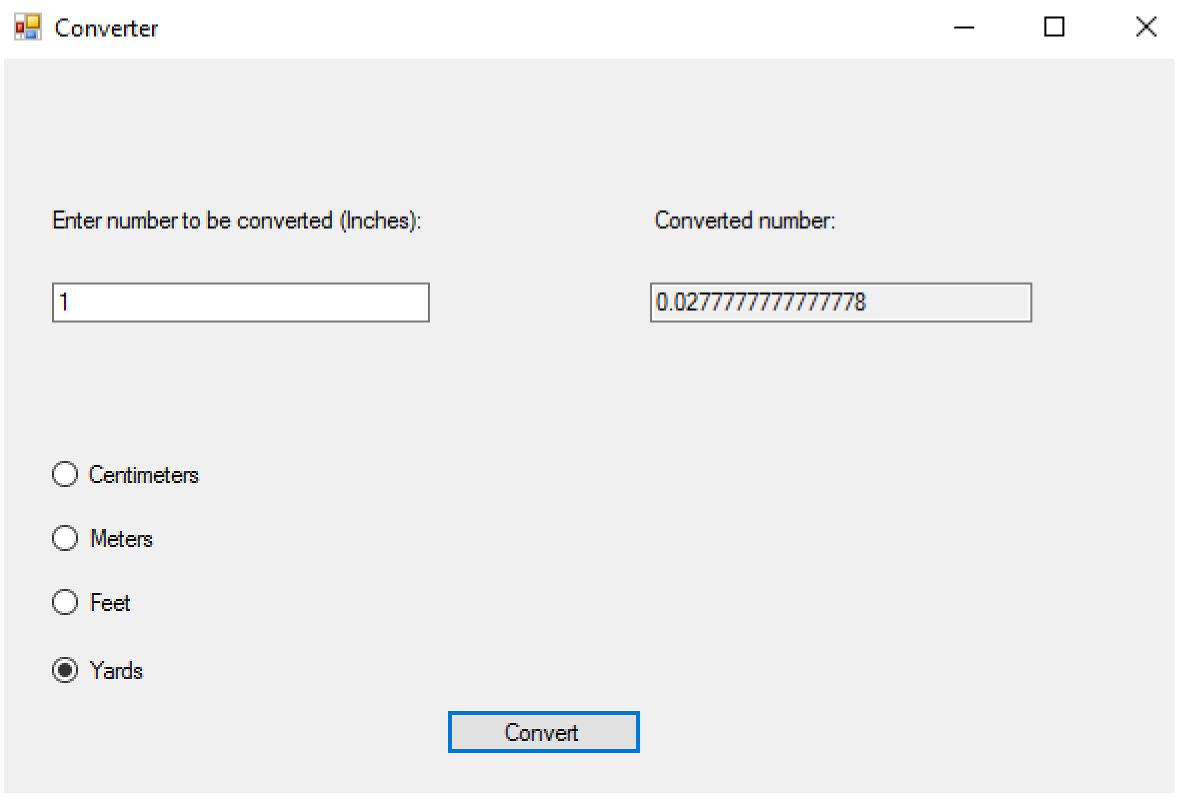
First, I defined inches and result, as well as created a convert object. I then made some radio button defaults for the program just so nothing could break. When the convert button is clicked, the inch string from the box is converted to a double. Next the app checks to see which radio button is selected, and makes the conversion accordingly. The result box is then update with the converted result as a string.


Converter

Enter number to be converted (Inches): 1

Converted number: 2.54

[● Centimeters] ○ Meters ○ Feet ○ Yards


Converter

Enter number to be converted (Inches): 1

Converted number: 0.0277777777777778

○ Centimeters ○ Meters ○ Feet [● Yards]

Conclusion: I can really see the many uses of this pattern. This pattern is so very easy, but makes so much sense to me, and I can see the many applications that it could be used for. I’ve always used google for number converters in many cases, so I thought it would be interesting to build my own.