Tori Lentz

Design Patterns

Dave Retterer

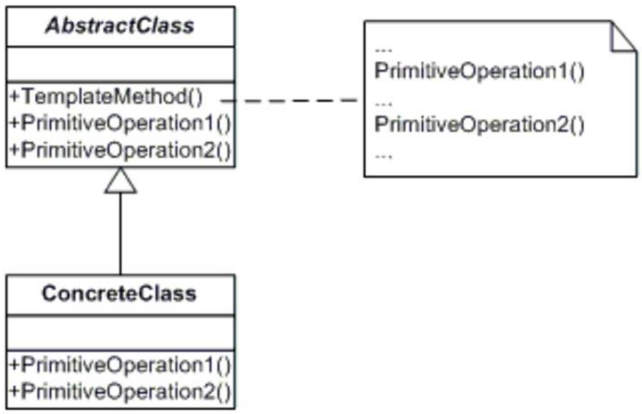
November 30, 2016

Template Pattern

Introduction

The purpose of this assignment is to design and implement the template pattern. As the name suggests, this pattern is to be used as a starting point and has a particular format that will be used. If, for example, you have a lot of programs that you want to look the same & have the same formatting, you can use this pattern to create the template. This pattern defines an algorithm in a base class using abstract operations that the concrete classes will override with concrete behavior.

UML Diagram



As shown by the UML diagram, the template pattern is made up of two different types of classes: the abstract class and the concrete classes. There will be one abstract class, which is where the template will be created. This class will also define the primitive operation that the concrete classes will redefine so they can implement the steps of an algorithm. The next part is the concrete class, there can be one concrete class or multiple concrete classes. These classes will have the definitions for the primitive operations within them and, depending on how many things one would want to do with their program, determines how many concrete classes they would have.

Code and Description

Abstract Class

```
public abstract class AbstractClass
{
    protected string ConnectionPath;
    protected DataSet dataset;

    public virtual void Connect()
    {
```

```
            ConnectionPath = "provider=Microsoft.JET.OLEDB.4.0; " + "data
source=C:\\Temp\\Northwind.mdb";
        }

        public abstract void Select();

        public abstract string Process();

        public virtual void Disconnect()
        {
            ConnectionPath = "";
        }

        public string Run()
        {
            string _colorOutput;

            Connect();
            Select();
            _colorOutput = Process();
            Disconnect();

            return _colorOutput;
        }
    }
```

This is my abstract class which sets up the template to be used by my other classes. My first method Connect sets up my connection to my database, Northwind. Within this database I made my own table called Colors. My Select and Process methods are the ones that will be overridden by my concrete classes. My method Run is the one I call to print my template pattern to the textbox.

Concrete Class A

```
    public class ColorNames : AbstractClass
    {
        public override void Select()
        {
            string sql = "select Color from Colors";
            OleDbDataAdapter dataAdapter = new OleDbDataAdapter(sql, ConnectionPath);
            dataset = new DataSet();
            dataAdapter.Fill(dataset, "Colors");
        }
        public override string Process()
        {
            string colorOutput = "";
            colorOutput = "Colors ----- \r\n";
            DataTable dataTable = dataset.Tables["Colors"];
            foreach (DataRow row in dataTable.Rows)
            {
                colorOutput = colorOutput + " - " + row["Color"] + "\r\n";
            }
            return colorOutput;
        }
    }
```

This is my first concrete class and within it, I override the Select and Process methods I had already defined in my abstract class. The Select method identifies and calls the correct column value I wanted from the table I created. My Process method then calls the rows from that column and returns them. The column identified and called in this class is Color.

Concrete Class B

```
    public class ARGBValues : AbstractClass
    {
        public override void Select()
        {
            string sql = "select ARGB from Colors";
            OleDbDataAdapter dataAdapter = new OleDbDataAdapter(sql, ConnectionPath);
```

This class is very similar to my other concrete class, yet it calls a different column and the rows that correspond to that column. The column that is called and returned in this class is ARGB.

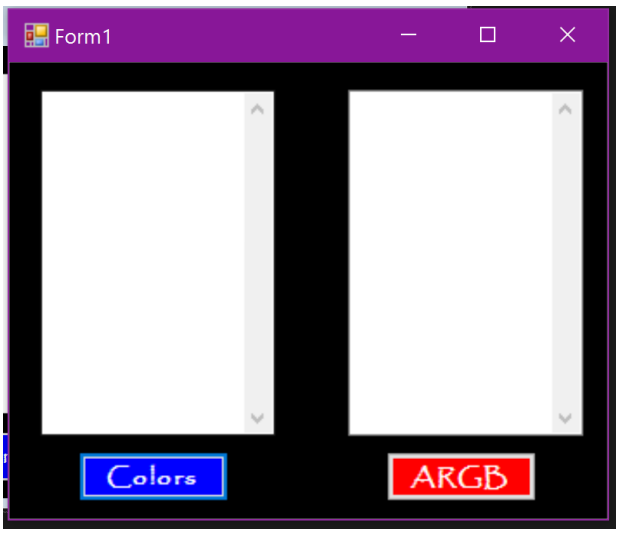
```
            dataset = new DataSet();
            dataAdapter.Fill(dataset, "Colors");
        }
        public override string Process()
        {
            string colorOutput = "";
            colorOutput = "ARGB ----- \r\n";
            DataTable dataTable = dataset.Tables["Colors"];
            foreach (DataRow row in dataTable.Rows)
            {
                colorOutput = colorOutput + " - " + row["ARGB"] + "\r\n";
            }
            return colorOutput;
        }
    }
```

Form

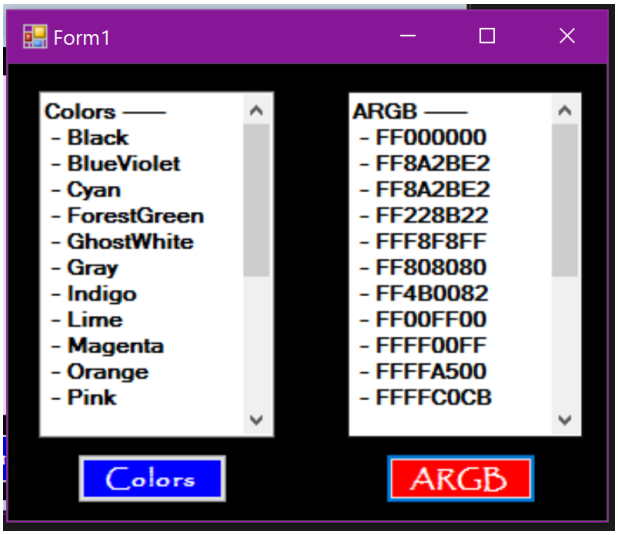
Within my form, I defined my abstract class as being each of my concrete classes and then I print the run method to textboxes through button clicks.

```
    public partial class Form1 : Form
    {
        AbstractClass ColorPallet = new ColorNames();
        AbstractClass argbvalues = new ARGBValues();

        public Form1()
        {
            InitializeComponent();
        }

        private void btn_DisplayColors_Click(object sender, EventArgs e)
        {
            tb_Colors.Text = ColorPallet.Run();
        }

        private void btn_ARGB_Click(object sender, EventArgs e)
        {
            tb_ARGB.Text = argbvalues.Run();
        }
    }
```

Screen Shots



Form on load



Form after buttons were clicked

Observations

This assignment went very well. I thought it was really cool how I was able to figure out how to connect to Microsoft Access and make my own table within a database. I really didn't have any issues with this pattern and look forward to using it more in the future because, I feel like it will be very useful.