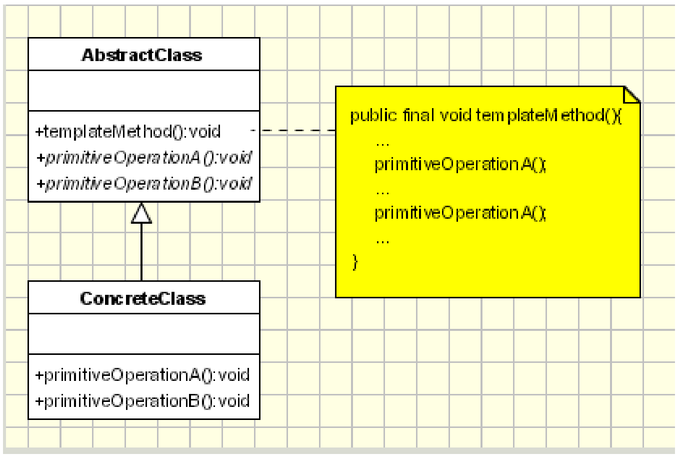
## Introduction

The template method pattern is used to build the outline of an algorithm or method that then calls on subclasses to fill in the rest of the algorithm or method. I did this with a program that shows a list of superheroes with each button calling a different list. To the right is the, really simple, UML that I followed.


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

## The Code

```
    public abstract class AbstractClass
    {


        public abstract string listMarvel();

        public abstract string listDC();

        public void Template_Method()
        {
            listMarvel();
            listDC();
        }
    }
```

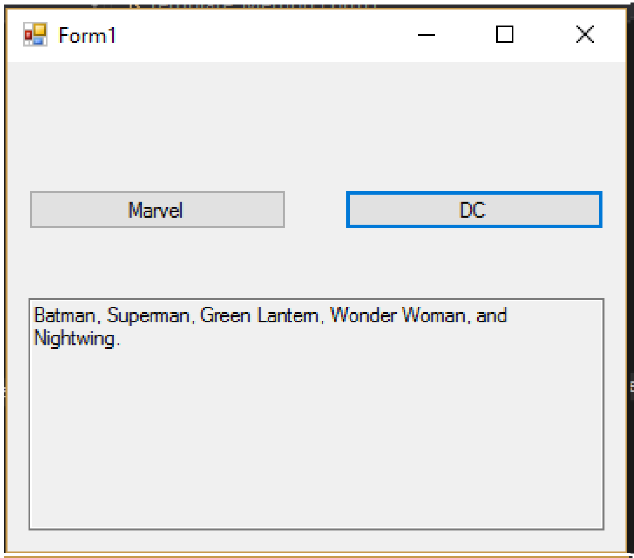
This is the abstract class. listMarvel and listDC are the two primitive operations.

```
  public class ConcreteClass : AbstractClass
    {
        public override string listDC()
        {

            return "Batman, Superman, Green Lantern, Wonder Woman, and Nightwing.";
        }

        public override string listMarvel()
        {
            return "Spider-Man, Captain America, Iron Man, Hulk, and Deadpool";
        }
    }
```

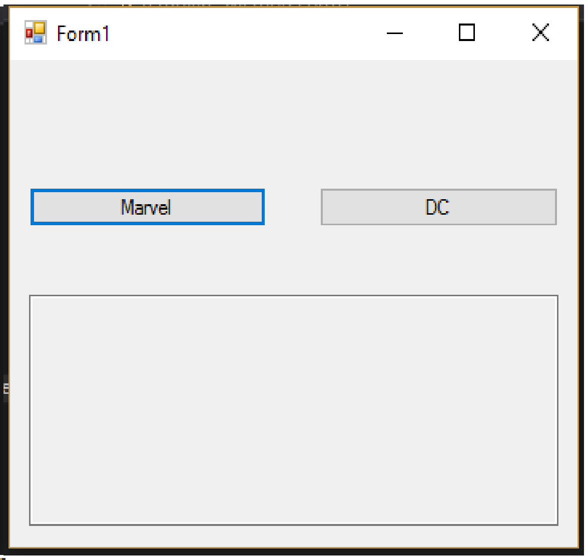
The concrete class derives from the abstract class. Here the primitive operations are defined. In this case they are given the strings that they will return.

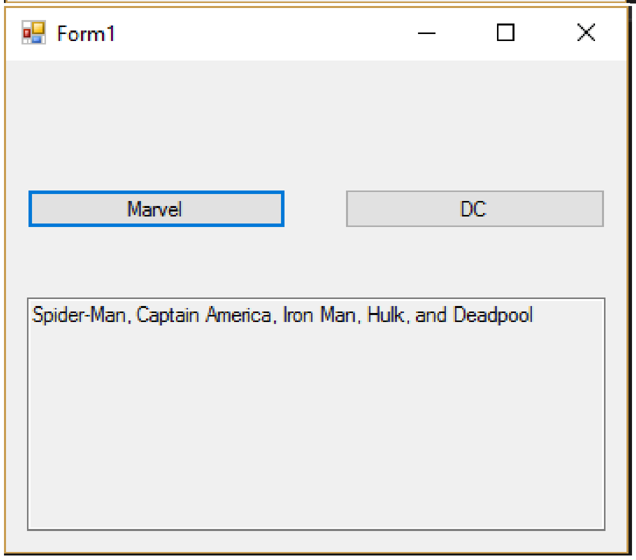
This is the form. Here the buttons call on the operations to populate the text box in my form.

```
public partial class Form1 : Form
{
    AbstractClass ac;
    public Form1()
    {
        InitializeComponent();

    }

    private void btnMarvel_Click(object sender, EventArgs e)
    {
        ac = new ConcreteClass();
        txtList.Text = ac.listMarvel();

    }

    private void btnDC_Click(object sender, EventArgs e)
    {
        ac = new ConcreteClass();
        txtList.Text = ac.listDC();
    }
       }
```



Batman, Superman, Green Lantern, Wonder Woman, and Nightwing.





Spider-Man, Captain America, Iron Man, Hulk, and Deadpool

**<u>Conclusion</u>**

This pattern was really easy, almost too easy. I over thought almost everything. I kept thinking I was skipping steps and as a result, I kept over complicating things. But now that I finished it, I can see how this would be applicable.