The Template Method Pattern

Design Patterns

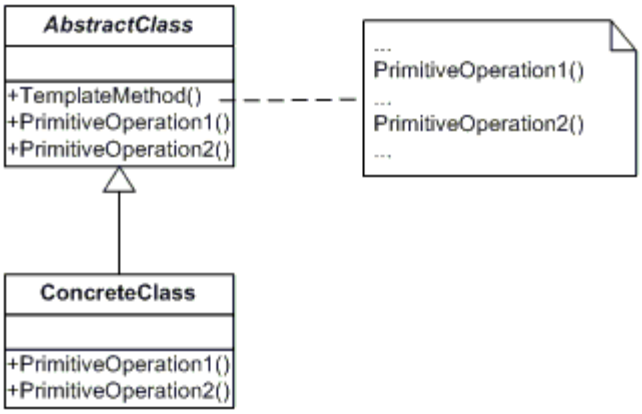
12/1/16

Cory Heitkamp

Introduction:

For this assignment, we were tasked with using the template method pattern in an application. The template method pattern is used to have a template for an operation, while some of the interior of the operation is controlled by subclasses. My particular program is simulates getting dressed, and the get dressed method works differently depending on if you want to be formal or casual.

UML Diagram: The diagram shows the most necessary components and functions used for the template method pattern. It describes how the classes are connected, and some of the functions contained inside.



This table below summarizes the classes that I've created and used to make the template method pattern.

Dressing	An abstract class with function whose interior is defined by concrete classes.
Casual	Defines the shirt and pants functions for a person dressing casual.
Formal	Defines the shirt and pants functions for a formal person.
Form	My windows form application

Narrative: I began by making the Dressing class. This is my abstract class. It contains the getDressed method, which contains abstract functions to be defined by the concrete classes.

```
public abstract class Dressing
    {
        public abstract string putOnShirt();
        public abstract string putOnPants();
        public string shirt;
        public string pants;
```

```
        public string getDressed()
        {
            shirt = putOnShirt();
            pants = putOnPants();
            return pants + " and " + shirt;
        }
    }
```

I then defined my first concrete class, the casual class. In this class, the putOnShirt and putOnPants methods return strings describing the nice comfortable clothes you are now wearing.

```
  class Casual : Dressing
    {
        public override string putOnPants()
        {
            return "You are wearing shorts";
        }

        public override string putOnShirt()
        {
            return "you are wearing a T-shirt.";
        }
    }
```

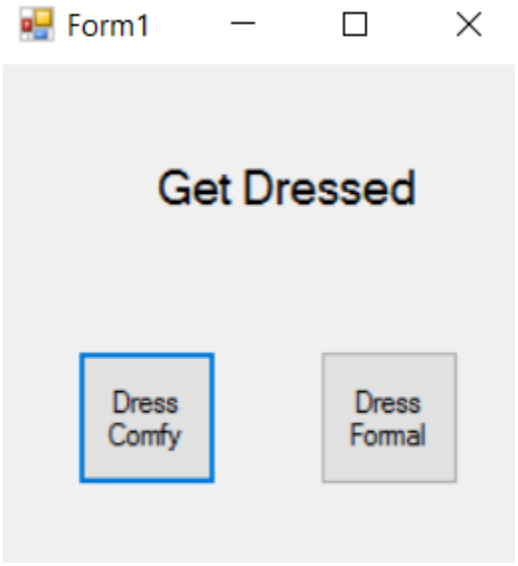
I then defined the formal class, whose functions describe much fancier clothes.

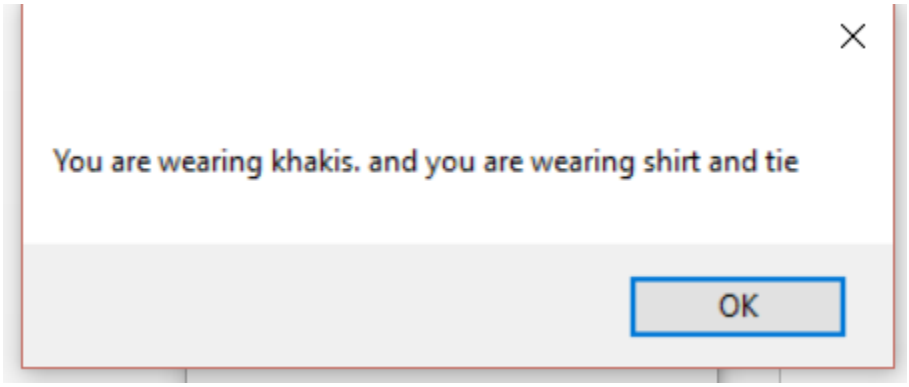
```
class Formal : Dressing
    {
        public override string putOnPants()
        {
            return "You are wearing khakis.";
        }

        public override string putOnShirt()
        {
            return "you are wearing shirt and tie";
        }
    }
```

Finally, I made the form itself. I started by making an object of the Dressing class. Then, depending on if the user selected formal or casual, it becomes an object of the appropriate concrete class, and the getDressed function is ran. Then a message box displays what you are wearing.

```
        private void formalbutton_Click(object sender, EventArgs e)
        {
            clothes = new Formal();
            MessageBox.Show(clothes.getDressed());
        }

        private void button1_Click(object sender, EventArgs e)
        {
            clothes = new Casual();
            MessageBox.Show( clothes.getDressed());
        }
```

And here is what the final product looks like:





Reflection:

For the Template Method pattern, the hardest part was figuring out a demo that displays the pattern. Once that was achieved the method itself was very straightforward, as it is possibly the simplest we have had.