

9/13/16

The façade pattern serves to hide all the complex operations that go on, and can do several operations at once. The example that was given to us in class was a universal remote. The façade pattern is extremely similar to a universal remote, since both do plenty of things with just a few clicks in most cases. I chose to use my façade pattern to obtain several different combinations that go on for weight lifting and exercising. The two methods I chose for exercising were defining muscle and bulking up muscle. The two methods of exercise are very different on how they are done, and the program I created specifies all the different things you need to do for each just by clicking a button for each category.

```

graph TD
    subgraph Facade
        F1[ ]
        F2[ ]
        F3[ ]
    end
    subgraph Subsystem
        S1[ ]
        S2[ ]
        S3[ ]
        S4[ ]
        S5[ ]
        S6[ ]
    end
    F1 --> S1
    F1 --> S2
    F1 --> S3
    F2 --> S4
    F2 --> S5
    F3 --> S6
  
```

The example in the video said that the many classes and actions were “make dough, add sauce, add cheese” but could all be summed up with one method in the façade pattern, “make pizza”.

The first thing I did when creating my façade pattern was to sort the different categories exercising could fall into. I found that dividing exercise into three different classes worked the best. These classes were Diet, Conditioning, and Lifting. All of these three items contribute to exercise and getting healthier.

```
public class Diet
{
    public enum Vegetables { low, medium, high}           //Declaring all the possible diet options
    public enum Protein { low, medium, high}
    public enum Fat { low, medium, high}

    public Vegetables vegAmt;
    public Protein proAmt;
    public Fat fatAmt;

    public void toneDiet()                                //Tone muscle diet specifications
    {
        vegAmt = Vegetables.high;
        proAmt = Protein.medium;
        fatAmt = Fat.low;
    }

    public void bulkDiet()                                // Bulking diet specifications
    {
        vegAmt = Vegetables.medium;
        proAmt = Protein.high;
        fatAmt = Fat.high;
    }
}
```

The diet class was the first class I wrote. The diet class is where I established the three basic parts of one's diet for exercising. Vegetables consumed, protein consumed, and fat consumed. I created an object for each of the following, and declared all of the possible options. Below I created a couple of methods, toneDiet served to specify what the amounts of each object were for a tone diet, and bulkDiet served to specify the amounts for a bulk diet.

The other two classes, Conditioning and Lifting were also done in a similar fashion to this one.

The code for the conditioning class:

```
public class Conditioning
{
    public enum Running { thirtyMins, tenMins }           //Declaring possible conditioning options
    public enum Sprints { eight, four}

    public Running runAmt;
    public Sprints sprintAmt;

    public void toneConiditioning()                       //Tone conditioning specifications
    {
        runAmt = Running.thirtyMins;
        sprintAmt = Sprints.eight;
    }

    public void bulkConditioning()                         //Bulk conditioning specifications
    {
        runAmt = Running.tenMins;
        sprintAmt = Sprints.four;
    }
}
```

The code for the lifting class:

```
public class Lifting
{
    public enum Weight { fortyLbs, ninetyLbs }           //Declaring possible lifting options
    public enum Sets { fourSets, threeSets }
    public enum Reps { twelveReps, fiveReps}

    public Weight weightAmt;
    public Sets setAmt;
    public Reps repAmt;

    public void toneLifting()                             //Tone lifting specifications
    {
        weightAmt = Weight.fortyLbs;
        setAmt = Sets.fourSets;
        repAmt = Reps.twelveReps;
    }

    public void bulkLifting()                             //Bulk lifting specifications
    {
        weightAmt = Weight.ninetyLbs;
        setAmt = Sets.threeSets;
        repAmt = Reps.fiveReps;
    }
}
```

The hardest part for this project however, was actually creating the façade pattern itself. The code for the façade class is as follows.

```
public class WorkoutFacade
{
    private Diet diet;
    private Conditioning condition;
    private Lifting lift;

    public WorkoutFacade(Diet diet1, Conditioning condition1, Lifting lift1)
//Constructor for the facade
    {
        diet = diet1;
        condition = condition1;
        lift = lift1;
    }

    public void toning()           //Sets the qualifications for toning muscle
    {
        diet.toneDiet();
        condition.toneConiditioning();
        lift.toneLifting();
    }

    public void bulking()         //Sets the qualifications for bulking muscle
    {
        diet.bulkDiet();
        condition.bulkConditioning();
        lift.bulkLifting();
    }
}
```

The WorkoutFacade() with the objects passed in is the constructor. Without this, the façade would have nothing to work with and would not be able to manage the information. The toning method sets all of the specifications for each object. For example, diet.toneDiet() sets all of the values in diet to the specifications for building tone muscle. It does the same here for conditioning and lift. The bulking does the same but with its respective values.

Finally, the form is what put everything together at the end.

```
public partial class Form1 : Form
{
    WorkoutFacade wf;
    Diet diet;
    Conditioning condition;
    Lifting lift;

    public Form1()
    {
        InitializeComponent();
        diet = new Diet();
        condition = new Conditioning();
        lift = new Lifting();
        wf = new WorkoutFacade(diet, condition, lift);
    }

    public void updateText()
    {
        weightTxt.Text = lift.weightAmt.ToString(); //Updates the textboxes based on button click
        setsTxt.Text = lift.setAmt.ToString();
        repsTxt.Text = lift.repAmt.ToString();
        runTxt.Text = condition.runAmt.ToString();
        sprintTxt.Text = condition.sprintAmt.ToString();
        vegTxt.Text = diet.vegAmt.ToString();
        proteinTxt.Text = diet.proAmt.ToString();
        fatTxt.Text = diet.fatAmt.ToString();
    }

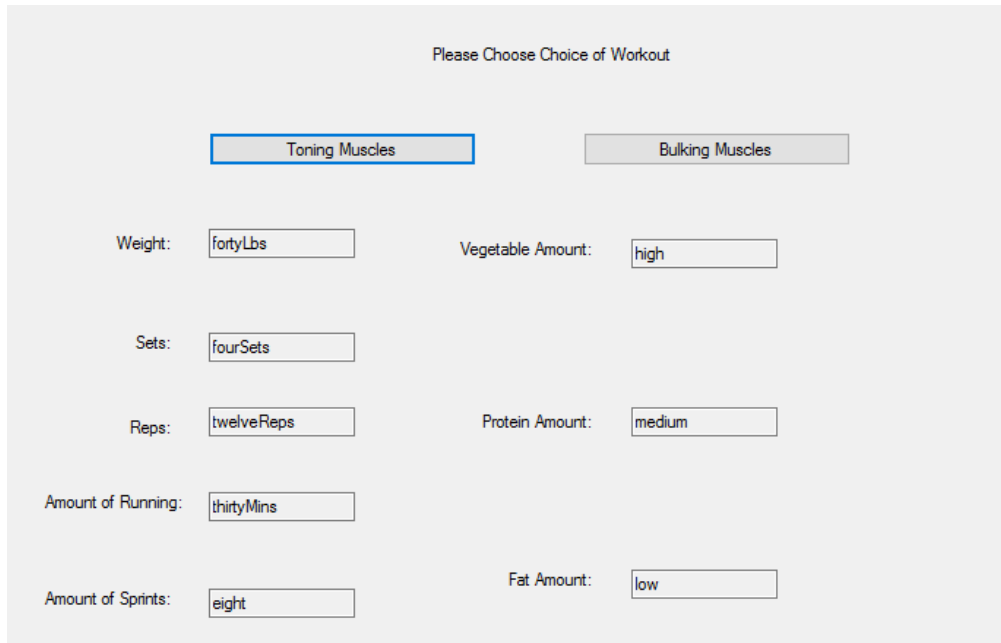
    private void toneBtn_Click(object sender, EventArgs e)
    {
        wf.toning(); //Facade pattern, changes all textboxes to fit toning options
        updateText();
    }

    private void bulkBtn_Click(object sender, EventArgs e)
    {
        wf.bulking(); //Facade pattern, changes all textboxes to fit bulking options
        updateText();
    }
}
```

The form starts off by initializing each of the classes, and passing in the objects into the Workout Façade. The toneBtn and the bulkBtn are the two available buttons that can be pressed to set the preferences for exercising. When the tone button is clicked, it calls the toning method in the workout façade class, and all the values are set, and then updated on the form because of the updateText method. When the bulk button is clicked, it calls the bulking method

in the workout façade class and all the values are set, then updated by the same updateText method from earlier.

Example of tone button being pressed:



Please Choose Choice of Workout

Toning Muscles Bulking Muscles

Weight: fortyLbs Vegetable Amount: high

Sets: fourSets

Reps: twelveReps Protein Amount: medium

Amount of Running: thirtyMins

Amount of Sprints: eight Fat Amount: low

Conclusion: This design pattern took a lot longer than the first pattern I did, but I really think I got a lot more out of this than the previous. While doing it I thought “Wow, I could just press this button and have it change the text and I could do this so much faster” but after completing it, I realize how much easier it can be to add more information to this now. The façade pattern brings everything together, and rather than making a button that just changes the text for each textbox by itself, I can just add more information to the classes which would be significantly easier. I really understand the value of this pattern much more after doing it.

Pizza example video watched:

<https://www.youtube.com/watch?v=WLjvNpP6yeQ&index=1&list=FL2PVVBxgBdeNYFZwdXD9S-A>