The Facade Pattern

Introduction: This project required the implementation of the facade design pattern. The definition of what a facade pattern is, is as follows:

"Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher level interface that makes the subsystem easier to use." *-dofactory.com*

Analysis: My approach to the facade was to create a virtual fish tank with all the elements of a fish tank, like the heater, filter, feeding, and lights, and control them with one unified interface. My facade is the panel labeled "Tank Controls" in Illustration 1. I have shown only one combination of settings that you can do with the four simple controls I narrowed it down into. The "Tank Size" setting will affect only the filter's gallons per minute (g.p.m.) rate, increasing when the size increases and vise versa for decreasing.

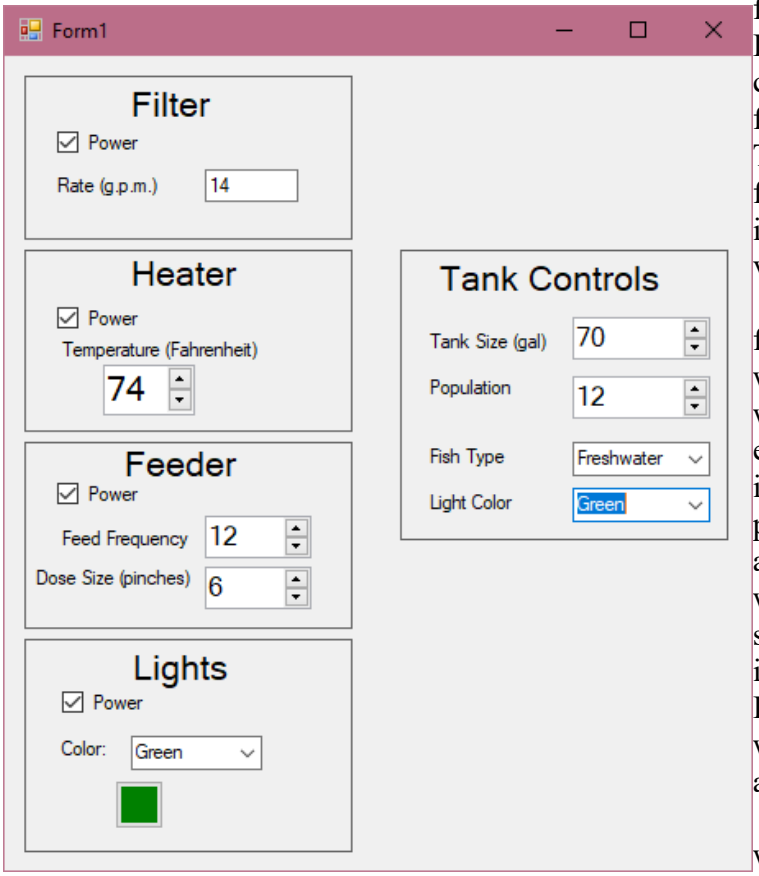

Illustration 1:

The population setting is where most of my facade's simplification takes place because whether or not there are inhabitants in the tank would determine if you want the tank to waste energy. For example, although you cannot see it in the screenshot I have presented, if the population is turned down to zero then automatically filter, heater, feeder and lights will all shutdown. Unless you turn them on separately. That is the special case however and if there are inhabitants in the tank, like in Illustration 1, then the feeder only will kick on with a healthy feed frequency of "12" hours and half a "pinch" of food per inhabitant.

Adjusting the "Fish Type" will affect the warmth of the tank and turn on the heater only. The defaults are "74" degrees for freshwater as shown and "78" degrees for saltwater fish. Then the final setting is just a drop down of six color options for the lights of the tank. You can change the color under the "Lights" panel and the color will be updated, as seen by the background of the currently green button, but won't affect the "Tank Controls" light color. However, changing the main light color under tank controls will override all colors and change the green button's color. Additionally, there is a `No Color` option which turns off all lights and makes the green button black.

Now that the basics of how the display works is covered I will talk about the coding behind the scenes. The best case of making things easy is the "Feeder" controls as seen in the text box to the right on the next page. It is only passed the set population number and from there determines the amount to feed and sets when to feed. Also, only if population is not zero does the feeder get turned on and because its value comes from a `NumericUpDown` there is no way for a negative.

The getters and setters are also shown in code in the lower text box on the right. I chose to only show the code for the feeder because the methods are basically copy and pasted with key words changed. `getState` is a `bool` because I wanted a straight return type that I could use in comparisons in the form. `setState` only changes the state of the feeder if the true or false result from if its checkbox is checked is true or false.

```
public TankFeeder(int pop)
{
  doseSize = pop / 2;
  frequency = 12;

  if (pop != 0)
  feederState = FeederState.ON;
  else
  feederState = FeederState.OFF;
}
```

```
public bool getState()
{
  if (feederState == FeederState.ON)
      return true;

      return false;
}
public void setState(bool OnOff)
{
  if (OnOff)
      feederState = FeederState.ON;
  else
      feederState = FeederState.OFF;
}
```

Reflection: This was a very interesting projects that even though it is one of the simplest patterns it made me think of how to control a single object in code two different ways. I also want to mention that I used the constructor of my Filter, Heater, Feeder, and Lights objects to changed all the settings when any of the main tank controls where changed. This may not have been a wise choice from a programmers view but I believe it was perfect for the demo. That code can be found below for how I managed to call constructors and take in strings from the controls to properly update the other fields which should be easy to comprehend.

```
        private void UpdateGPM()
        {
            tbox_gpm.Text = (numUpDown_tanksize.Value / 5).ToString();
            filter = new TankFilter( Convert.ToInt32(tbox_gpm.Text) );

            checkbox_filterPower.Checked = filter.getState();

        }

        private void UpdateHeat()
        {
            if(combobox_type.Text == "Freshwater")
                numUpDown_temp.Value   = Convert.ToDecimal(74);
            else
                numUpDown_temp.Value = Convert.ToDecimal(78);

            heater = new TankHeater( Convert.ToInt32(numUpDown_temp.Value), "F");

            checkbox_heaterPower.Checked = heater.getState();
        }

        private void UpdateLights()
        {
            lights = new TankLights(combobox_color.SelectedItem.ToString());

            comboColor.Text = lights.Color;//gets the current selected color to the light
dropdown

            checkbox_lightPower.Checked = lights.getState();

            switch (comboColor.SelectedItem.ToString())
            {
```

```
            case "Red":    btn_color.BackColor = Color.Red;
                break;
            case "Green":  btn_color.BackColor = Color.Green;
                break;
            case "Blue":   btn_color.BackColor = Color.Blue;
                break;
            case "Yellow": btn_color.BackColor = Color.Yellow;
                break;
            case "White":  btn_color.BackColor = Color.White;
                break;
            default:       btn_color.BackColor = Color.Black;
                break;

        }

    }

    private void UpdateFeeder()
    {
        population = Convert.ToInt32(numUpDown_population.Value);

        feeder = new TankFeeder(population);

        if ( feeder.getState() )
        {
            checkbox_feederPower.Checked = feeder.getState();

            numUpDown_dose.Value = feeder.getDose();
            numUpDown_frequency.Value = feeder.getFrequency();
        }

        else
        {//here if there are no fish so we can turn filter, heater, and feeder off
            checkbox_filterPower.Checked = false;
            filter.setState(false);

            checkbox_heaterPower.Checked = false;
            heater.setState(false);

            checkbox_feederPower.Checked = false;
            feeder.setState(false);
        }
    }
```