Façade Pattern

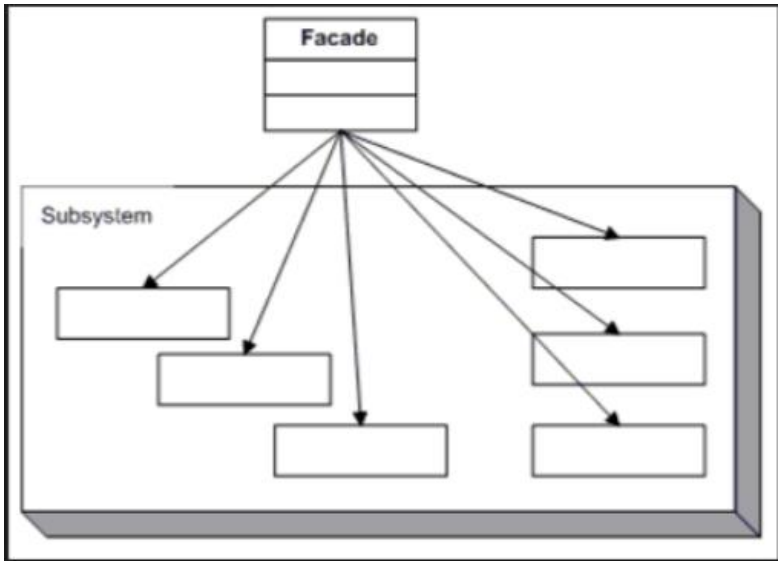
Design Patterns

Nathan Bemus

Façade Pattern

Introduction

The purpose of this pattern is to provide a united interface for a set of subsystems to make it easier to use. This particular program demonstrates the façade pattern by creating a universal remote for my laptop. The program follows the following UML diagram.



UML Diagram Analysis

The diagram shows that the façade pattern is divided into four classes. In it lies the Façade Class, and multiple subsystem classes. The client accesses the Façade class and the façade reaches out the subsystems as it needs information.

Program Walkthrough

Laptop Class (Façade Class)

My specific class called Laptop acts as the Façade Class. The class instantiates some public objects that are specific to their class. Those objects all contain methods that can be accessed by the Laptop Class. The Laptop class then creates its own public methods. These public methods call upon the objects methods to manipulate the variables in the other subsystem classes.

```
public class Laptop
    {

        public Power power = new Power();
        public Movies movies = new Movies();
        public Gaming gaming = new Gaming();
        public Homework homework = new Homework();
        public Internet internet = new Internet();

        public void Steam()
        {
            power.PowerOn();
            gaming.SteamOn();
            internet.InternetUseHigh();
            movies.NetflixOff();
            homework.IDEOff();
        }

        public void IDE()
        {
            power.PowerOn();
            gaming.SteamOff();
            internet.InternetUseLow();
            movies.NetflixOff();
            homework.IDEOn();
        }

        public void Netflix()
        {
            power.PowerOn();
            gaming.SteamOff();
            internet.InternetUseHigh();
            movies.NetflixOn();
            homework.IDEOff();
        }

        public void Power()
        {
            power.PowerDown();
            gaming.SteamOff();
            internet.InternetUseLow();
            movies.NetflixOff();
            homework.IDEOff();
        }

    }
```

Power Class (example of a Subsystem Class)

This is one of the subsystem classes that is accessed through the Laptop (Façade) Class. The subsystem classes are all set up the same as this Power Class. The Power class starts off by initializing a private Boolean and sets it false as a default stance. It then creates some public methods that returns either true or false, or just sends back the current position of the variable (whether the Boolean is true or false).

```
public class Power
    {
        private bool powerOn = false;

        public Power(){}

        public bool isPowerOn()
        {
            return powerOn;
        }

        public void PowerOn()
        {
            powerOn = true;
        }

        public void PowerDown()
        {
            powerOn = false;
        }
    }
```

Form1 Code (part 1)

This is where the main part of the code where everything comes together and all the functions come start manipulating the objects. This is the first half of the source code for Form1. In this section, a Laptop object is created to communicate to the other subsystem classes. This section also creates the functionality for the power button. The button functionality is very simple. It calls to the Laptop Class and does the specific method in that class, and then updates the screen.

```
public partial class Form1 : Form
    {
        Laptop remote;

        public Form1()
        {
            InitializeComponent();
            remote = new Laptop();
        }

        //Useless section of code that I cant get rid of
        private void Facade_Load(object sender, EventArgs e)
        {

        }
        //

        private void btnPower_Click(object sender, EventArgs e)
        {
            remote.Power();
            updateForm();
        }
```

Form1 Code (part 2)

This is the last section of the Form1 code. In this section the udateForm class is created. This class goes through and checks all the Booleans imbedded in the subsystem classes and returns their current status. Based off of the results returned, the method will change the labels on the form to display the correct information.

```
private void updateForm()
        {
            if (remote.power.isPowerOn())
            {
                lblPowerStatus.Text = "On";
            }
            else
            {
                lblPowerStatus.Text = "Off";
            }

            if (remote.gaming.isSteamActive())
            {
                lblSteamStatus.Text = "Active";
            }
            else
            {
                lblSteamStatus.Text = "Inactive";
            }

            if (remote.internet.isInternetUseHigh())
            {
                lblInternetUseStatus.Text = "High";
            }
            else
            {
                lblInternetUseStatus.Text = "Low";
            }

            if (remote.movies.isNetflixActive())
            {
                lblNetflixStatus.Text = "Active";
            }
            else
            {
                lblNetflixStatus.Text = "Inactive";
            }

            if (remote.homework.isIDEActive())
            {
                lblIDEStatus.Text = "Active";
            }
            else
            {
                lblIDEStatus.Text = "Inactive";
            }
        }
```

Reflection

This pattern uses a façade class to interact with a collection of subsystem classes at once to simplify input. This is shown in

the form of a remote that “controls” my laptop through the use of buttons.

Conclusion

This particular assignment taught me a general understanding on how to use a façade pattern, and gave me more experience with C# OOP. I think the assignment was enjoyable to do and was very satisfying to complete. I am hoping the next project is just as enjoyable to work on.