Design Patterns Jeremy Martin
Prof. Retterer 11/8/16

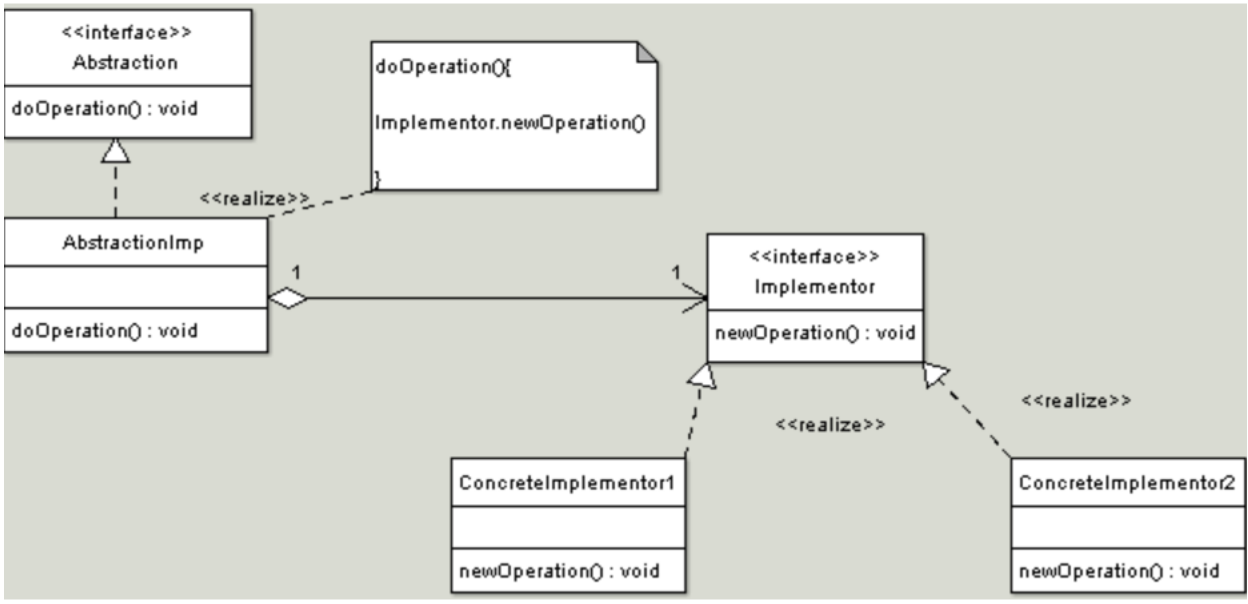
Bridge Pattern

Introduction

For this project, we were tasked with making an application that demonstrates the Bridge pattern. My application lets you choose between a regular and an admin user and gives you a message to show which user you are currently in.

Bridge UML

To the right, you will find the UML diagram for the Bridge pattern.



Now, we will move on to the code that makes my application work the way that it does.

The first line of code shows the abstract class Abstraction.

```
public abstract class Abstraction
    {
        public abstract void changeUser();
    }
```

The next piece of code shows the AbstractionImp class.

```
    public class AbstractionImp : Abstraction
    {
        public Implementor Imp;

        public AbstractionImp(Implementor imp)
        {
            Imp = imp;
        }

        public override void changeUser()
        {
          Imp.chooseUser();
        }
    }
```

The next pieces of code show the abstract class Implementor and the two concreteImplementors, AdminUser and RegularUser.

```
public abstract class Implementor
    {
       public abstract void chooseUser();
    }

public class AdminUser : Implementor
    {
        public override void chooseUser()
        {
            MessageBox.Show("You are now an Admin user, enjoy changing the
computer as you see fit.", "User Status");
        }
    }

    public class RegularUser : Implementor
    {
        public override void chooseUser()
        {
            MessageBox.Show("You are now a Regular user, enjoy your experience.",
"User Status");
        }
    }
```

The piece of code that I needed to add to both of the concreteImplementors in order for them to use the MessageBox command is this.

```
using System.Windows.Forms;
```

Finally, here is the Form1 code.

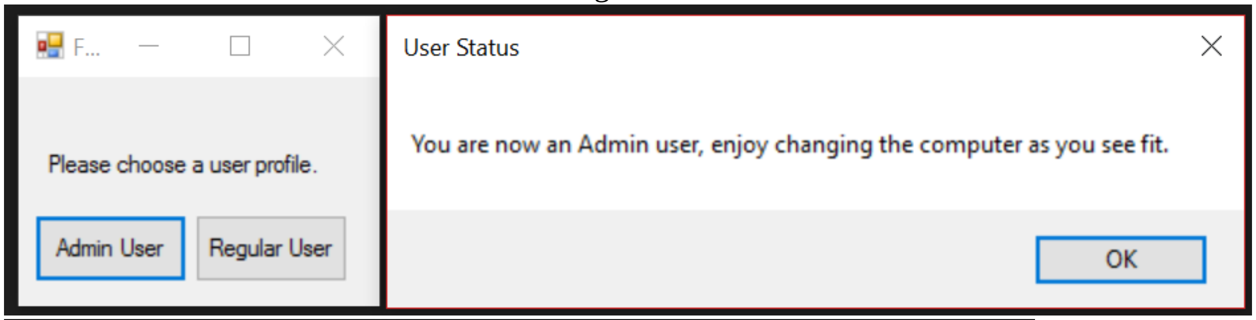
```
public partial class Form1 : Form
    {
        AbstractionImp AbsImp;

        public Form1()
        {
            InitializeComponent();
        }

        private void m_btnAdmin_Click(object sender, EventArgs e)
        {
            AbsImp = new AbstractionImp(new AdminUser());
            AbsImp.changeUser();
        }

        private void m_btnRegular_Click(object sender, EventArgs e)
        {
            AbsImp = new AbstractionImp(new RegularUser());
            AbsImp.changeUser();
        }
    }
```

Here are a few screenshots of the application working.

Design Patterns
Prof. Retterer

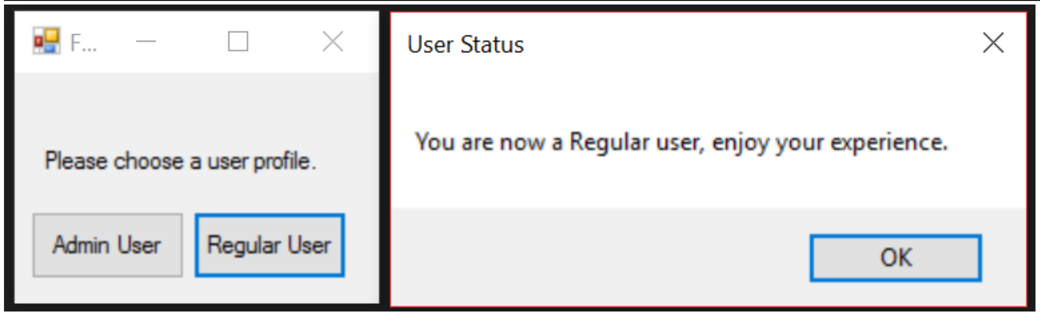
Jeremy Martin
11/8/16

Bridge Pattern



Please choose a user profile.

You are now an Admin user, enjoy changing the computer as you see fit.



Please choose a user profile.

You are now a Regular user, enjoy your experience.

Conclusion

Overall, I think that this was a pretty easy pattern to get the concept of. There is probably a lot more that I can do with this pattern in the future and I feel like I've only scratched the surface of what this pattern is truly capable of. The bridge allows us to use multiple classes without having to make a reference to all of them within the main form code.