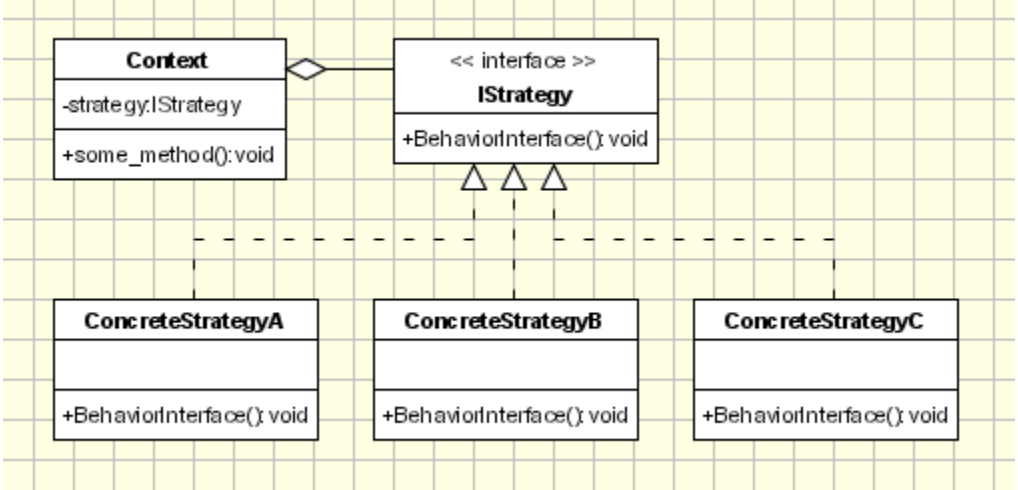
Jack Raney

Design Patterns

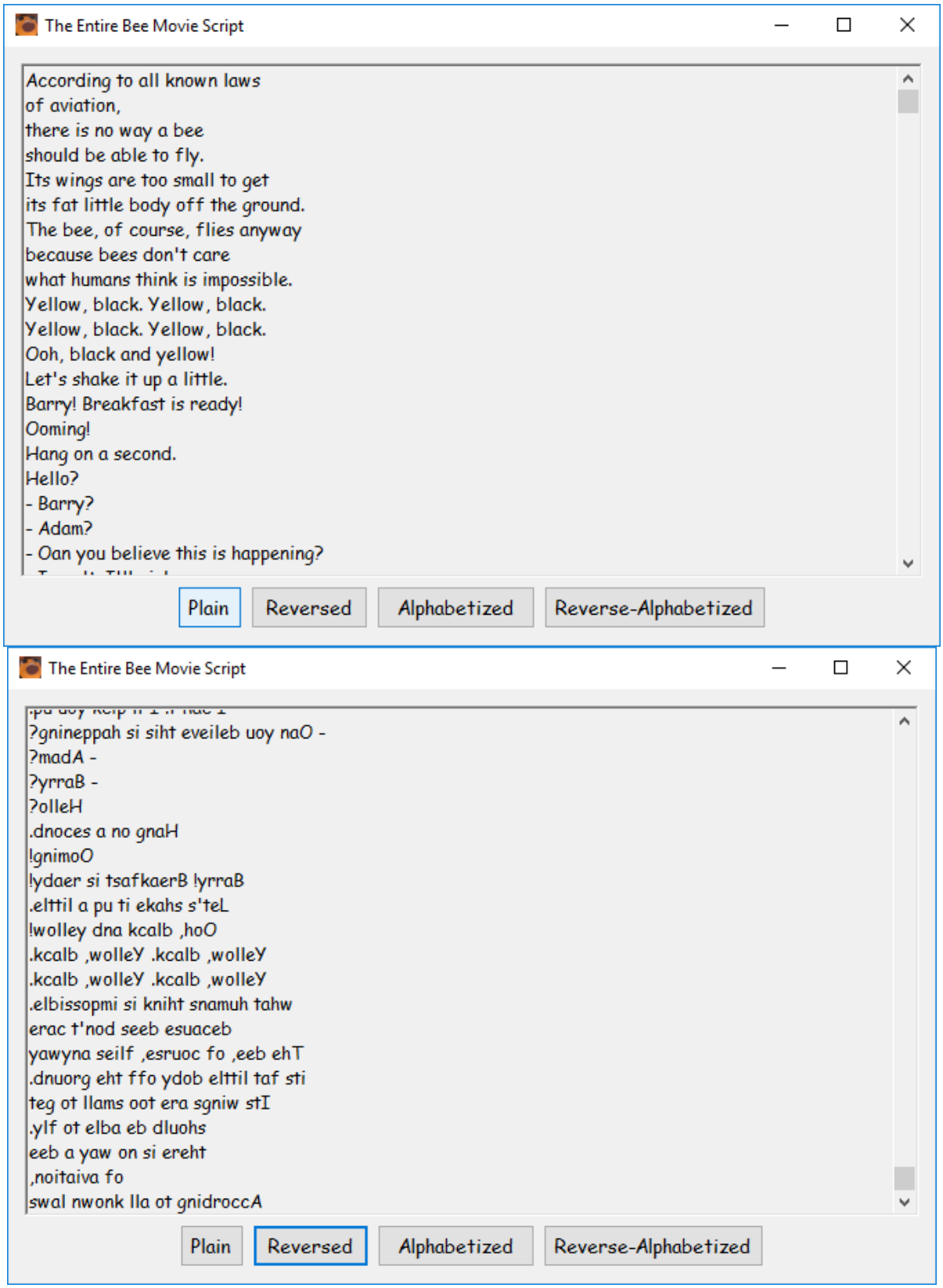
10/20/2016

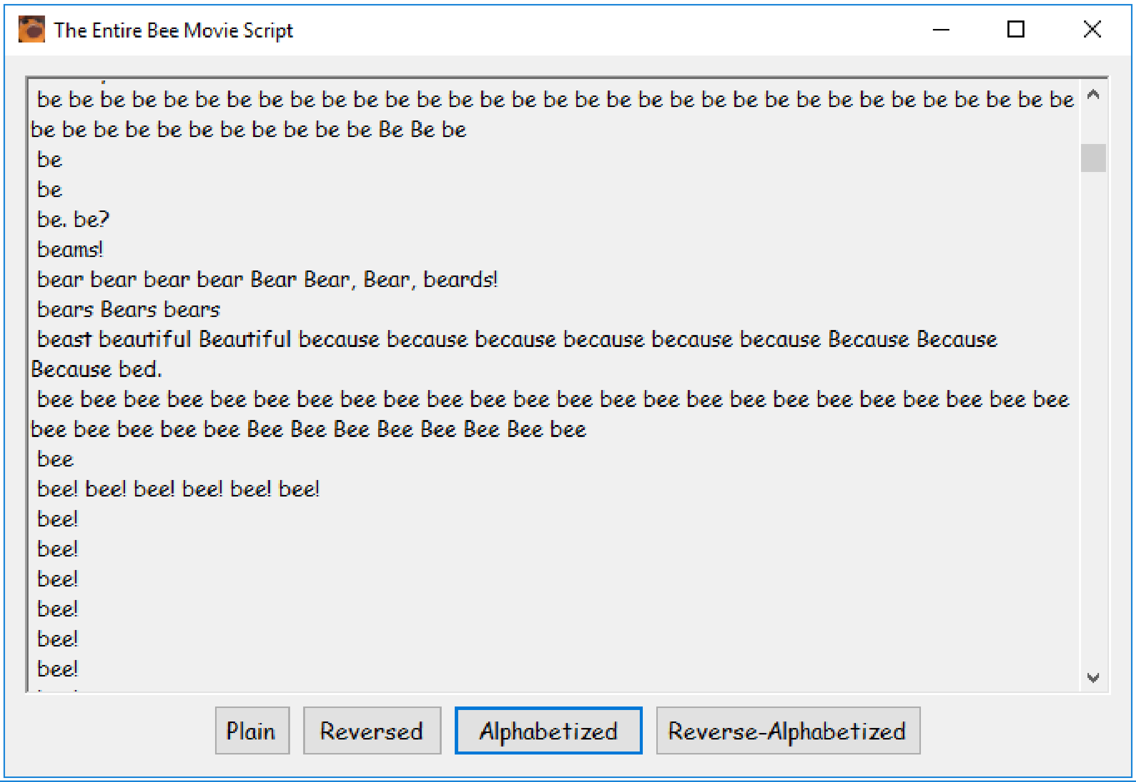
The Strategy Pattern

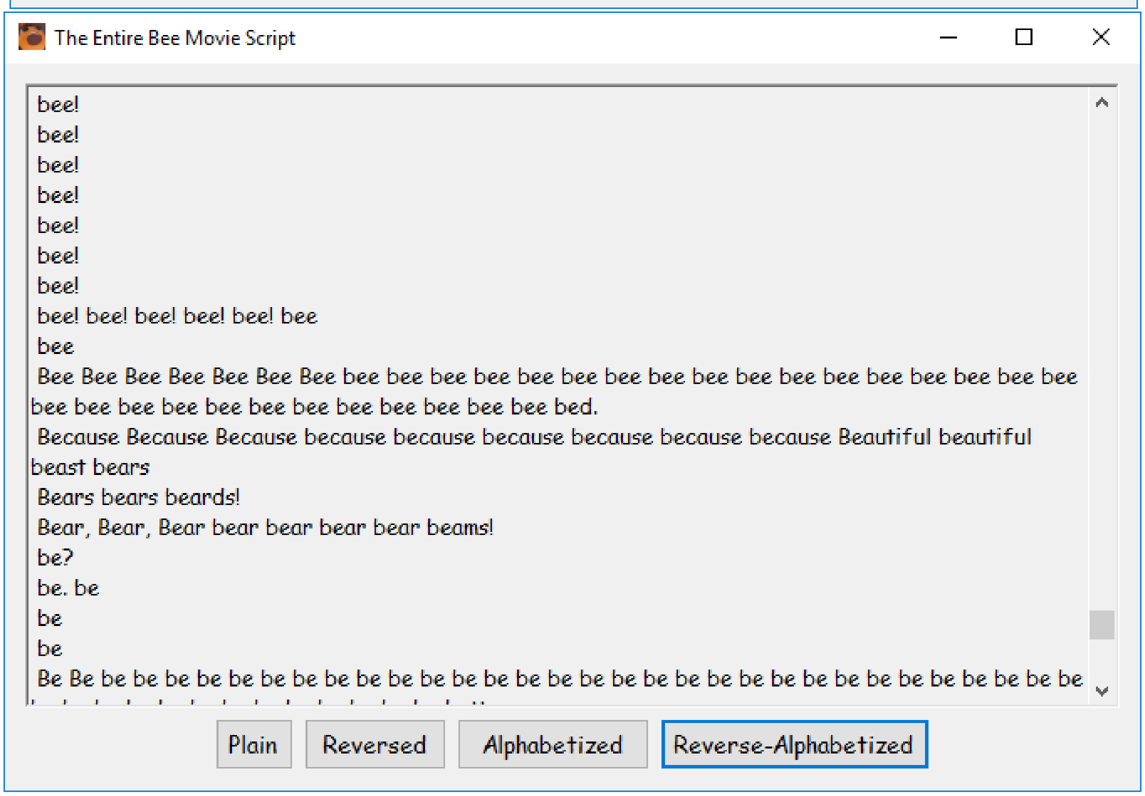
This paper will be explaining and showing an example of the Strategy Pattern, a design pattern that's intent is to, according to oodesign, "define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it". This means that the pattern allows an object to call a set of objects whose only differences are what the functions they contain do. The UML diagram for this pattern is shown below:



The example program I created for this pattern is BeeMovieScript:




The Entire Bee Movie Script
be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be
be be be be be be be be be be be be Be Be be
be
be
be. be?
beams!
bear bear bear bear Bear Bear, Bear, beards!
bears Bears bears
beast beautiful Beautiful because because because because because because Because Because
Because bed.
bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee
bee bee bee bee bee Bee Bee Bee Bee Bee Bee Bee bee
bee
bee! bee! bee! bee! bee! bee!
bee!
bee!
bee!
bee!
bee!
bee!
Plain
Reversed
Alphabetized
Reverse-Alphabetized


The Entire Bee Movie Script
bee!
bee!
bee!
bee!
bee!
bee!
bee!
bee! bee! bee! bee! bee! bee
bee
Bee Bee Bee Bee Bee Bee Bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee bee
bee bee bee bee bee bee bee bee bee bee bee bee bee bed.
Because Because Because because because because because because because Beautiful beautiful
beast bears
Bears bears beards!
Bear, Bear, Bear bear bear bear bear beams!
be?
be. be
be
be
Be Be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be
Plain
Reversed
Alphabetized
Reverse-Alphabetized

Code

ReadMethod.cs

```
public abstract class ReadMethod    //IStrategy class
    {
        public abstract string read(System.IO.StreamReader sr);
    }
```

PlainMethod.cs

```
public class PlainMethod : ReadMethod   //ConcreteStrategy
    {
        public override string read(System.IO.StreamReader sr)
        {
            return sr.ReadToEnd();
        }
    }
```

ReverseMethod.cs

```
public class ReverseMethod : ReadMethod //ConcreteStrategy
    {
        public override string read(System.IO.StreamReader sr)
        {
            char[] orderedScript = sr.ReadToEnd().ToCharArray();
            Array.Reverse(orderedScript);

            for (int i = 0; i < orderedScript.Length - 1; i++)
            {
                if (orderedScript[i + 1].Equals('\r'))
                {
                    orderedScript[i] = '\r';
                    orderedScript[i + 1] = '\n';
                }
            }

            return new string(orderedScript);
        }
    }
```

ReadMethod serves as a parent class for the methods that determine how the script's text is ordered.

PlainMethod puts it in its original order.

ReverseMethod reverses the order (which requires some correcting of the NewLine characters).

AlphabetizeMethod.cs

```
public class AlphabetizeMethod : ReadMethod //ConcreteStrategy
    {
        public override string read(System.IO.StreamReader sr)
        {
            string script = sr.ReadToEnd();

            char[] splitChars = {' ', '\n'};

            string[] sortArray = script.Split(splitChars);

            List<string> sortList = new List<string>();

            foreach (string s in sortArray)
            {
                sortList.Add(s);
            }

            sortList.Sort();

            string alphaScript = "";

            foreach (string s in sortList)
            {
                alphaScript += s + " ";
            }

            return alphaScript;
        }
    }
```

AlphabetizeMethod orders each word in alphabetical order (which required transferring the data into several different aggregates for different reordering methods.

ReverseAlphabetizeMethod does the same task, but with the words in reverse order.

ReverseAlphabetizeMethod.cs

```
public class ReverseAlphabetizeMethod : ReadMethod  //ConcreteStrategy
    {
        public override string read(System.IO.StreamReader sr)
        {
            string script = sr.ReadToEnd();

            char[] splitChars = { ' ', '\n' };

            string[] sortArray = script.Split(splitChars);

            List<string> sortList = new List<string>();

            foreach (string s in sortArray)
            {
                sortList.Add(s);
            }

            sortList.Sort();

            sortList.Reverse();

            string alphaScript = "";

            foreach (string s in sortList)
            {
                alphaScript += s + " ";
            }

            return alphaScript;
        }
    }
```

FileReader.cs

FileReader reads text from a text file and uses the read function to return the text depending on which ReadMethod subclass is currently set using setMethod

```
public class FileReader //Context class
    {
        private ReadMethod method;

        public void setMethod(ReadMethod rm)
        {
            method = rm;
        }

        public string read()
        {
            System.IO.StreamReader reader = new System.IO.StreamReader("script.txt");

            try
            {
                using (reader)
                {
                    return method.read(reader);
                }
            }
            catch (Exception e)
            {
                return "File read error: " + e.Message;
            }
        }
    }
```

Form1.cs

In the form, each button sets the ReadMethod to its respective child and sets the text within the text box to the returned string from calling the read function.

```
public partial class Form1 : Form
    {
        FileReader fr = new FileReader();

        public Form1()
        {
            InitializeComponent();
        }

        private void plainButton_Click(object sender, EventArgs e)
        {
            fr.setMethod(new PlainMethod());
            scriptBox.Text = fr.read();
        }

        private void reverseButton_Click(object sender, EventArgs e)
        {
            fr.setMethod(new ReverseMethod());
            scriptBox.Text = fr.read();
        }

        private void alphaButton_Click(object sender, EventArgs e)
        {
            fr.setMethod(new AlphabetizeMethod());
            scriptBox.Text = fr.read();
        }

        private void reverseAlphaButton_Click(object sender, EventArgs e)
        {
            fr.setMethod(new ReverseAlphabetizeMethod());
            scriptBox.Text = fr.read();
        }
    }
```

Reflection

The most difficult part of this assignment was thinking of an original idea for an example program. I was originally going to create a sketch pad program with the different strategies as different pen types, but that was proving to be too difficult. I like the program I made instead, though, and it fulfilled all the requirements for the pattern with less effort. Overall, I am happy with my work in this assignment.