Design Patterns

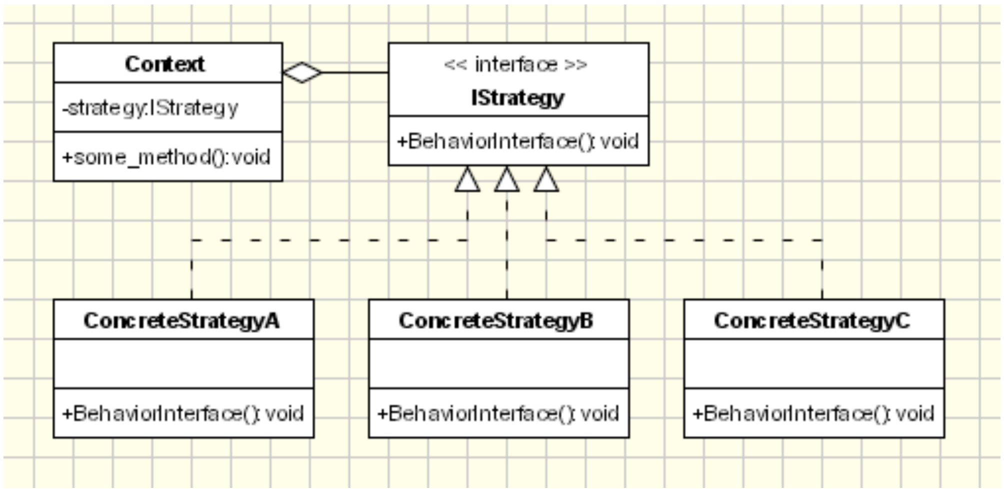
Strategy Pattern

Collin Kemner

October 25, 2016

Introduction

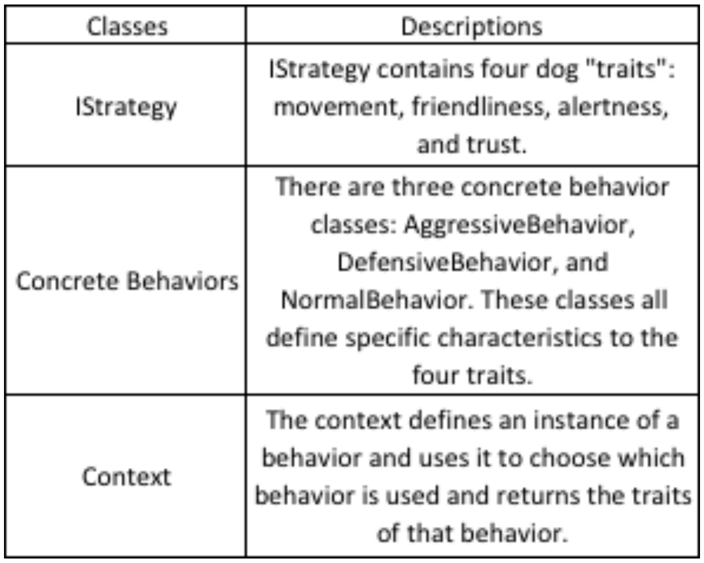
This paper will detail and perform an application demo which will utilize the strategy pattern. The application in question will be a “dog behavior simulator”, which will detail how a dog may act in certain aspects if it is in an aggressive, defensive, or normal stance. It’s a pretty simple application, but all that matters is that it showcases and correctly uses the strategy pattern.



UML Diagram

The strategy pattern, as oodesign.com puts it, “[The strategy pattern] defines a family of algorithms, encapsulates each one, and makes them interchangeable.” The UML diagram above showcases this as the IStrategy class defines an interface of commands, or in this case, traits of certain behaviors. These traits are then made into concrete strategy classes which then implements an algorithm based on its trait. The context class defines an interface in which the strategy gets to access and contains references to the strategy objects.

Short Descriptions of Classes



Classes	Descriptions
IStrategy	IStrategy contains four dog "traits": movement, friendliness, alertness, and trust.
Concrete Behaviors	There are three concrete behavior classes: AggressiveBehavior, DefensiveBehavior, and NormalBehavior. These classes all define specific characteristics to the four traits.
Context	The context defines an instance of a behavior and uses it to choose which behavior is used and returns the traits of that behavior.

The Application's Code

IStrategy Class

```
public interface IStrategy
    {
        string moveCommand(string name);
        string friendlinessCommand(string name);
        string alertnessCommand(string name);
        string trustCommand(string name);
    }
```

The first class to discuss is the IStrategy class, which defines the four traits. Each string takes a name variable of type string. The name will be the name of the dog specified in the application window.

Concrete Behavior Classes

```
public class AggressiveBehavior : IStrategy
    {
        public string moveCommand(string name)
        {
            return name + " will move in an aggressive fashion.";
        }
        public string friendlinessCommand(string name)
        {
            return name + "'s friendliness will go down.";
        }
        public string alertnessCommand(string name)
        {
            return name + " will become more alert and will act aggressively towards any 
threat.";
        }
        public string trustCommand(string name)
        {
            return name + "'s trust with strangers will go down.";
        }
    }

public class DefensiveBehavior : IStrategy
    {
        public string moveCommand(string name)
        {
            return name + " will move in a cautious, defensive fashion.";
        }
        public string friendlinessCommand(string name)
        {
            return name + "'s friendliness will go down.";
        }
        public string alertnessCommand(string name)
        {
            return name + " will become more alert and will act
in a defensive manner on behalf of its owner.";
        }
        public string trustCommand(string name)
        {
            return name + "'s trust with strangers will go
down.";
        }
    }

    public class NormalBehavior : IStrategy
    {
        public string moveCommand(string name)
        {
            return name + " will move on as normal.";
        }
        public string friendlinessCommand(string name)
        {
            return name + "'s friendliness will be at a normal level.";
        }
        public string alertnessCommand(string name)
        {
            return name + "'s alertness will be lower.";
        }
        public string trustCommand(string name)
        {
            return name + "'s trust with strangers will be good.";
        }
    }
```

Next are the concrete behavior classes. There are three for aggressive behavior, defensive behavior, and normal behavior and they extend the IStrategy class. Each behavior classes contain strings for the four traits and also take ir the name of the dog so that the strings address the right dog name. The traits are specific to its behavior, so each strings will have slightly different wording and will accommodate how a dog may act in a situation with that behavior.

Context Class

```
public class Dog
    {
        IStrategy behavior;

        public Dog(IStrategy _behavior)
        {
            this.behavior = _behavior;
        }

        public string move(string name)
        {
            return this.behavior.moveCommand(name);
        }
        public string friendliness(string name)
        {
            return this.behavior.friendlinessCommand(name);
        }
        public string alertness(string name)
        {
            return this.behavior.alertnessCommand(name);
        }
        public string trust(string name)
        {
            return this.behavior.trustCommand(name);
        }
    }
```

Next is the context class (named "Dog" in this case). This class is what will allow the client to access the behaviors and its traits. The class defines an instance of the IStrategy class, named "behavior" here. There are then four strings (named after the four traits) which takes is the dog name from the client. The four strings then return the traits' strings from the concrete behavior classes. The behavior selected is determined by which behavior from the client has been selected.

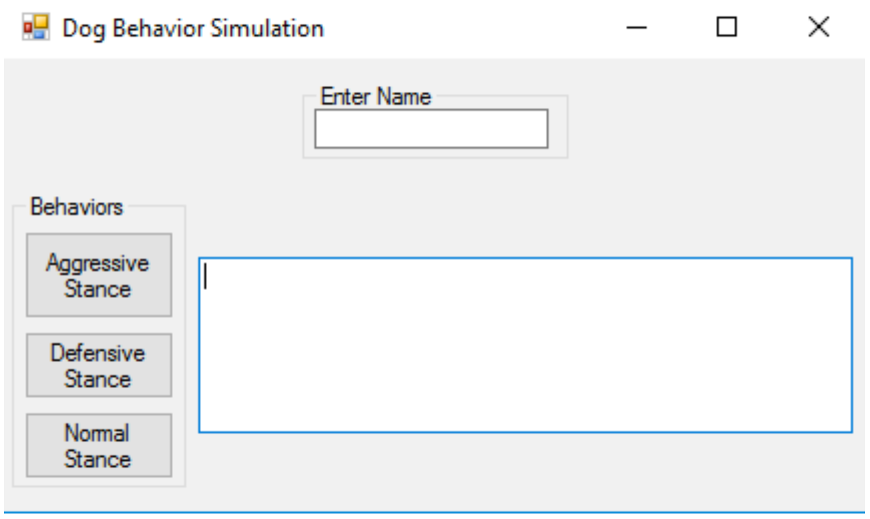
The Client

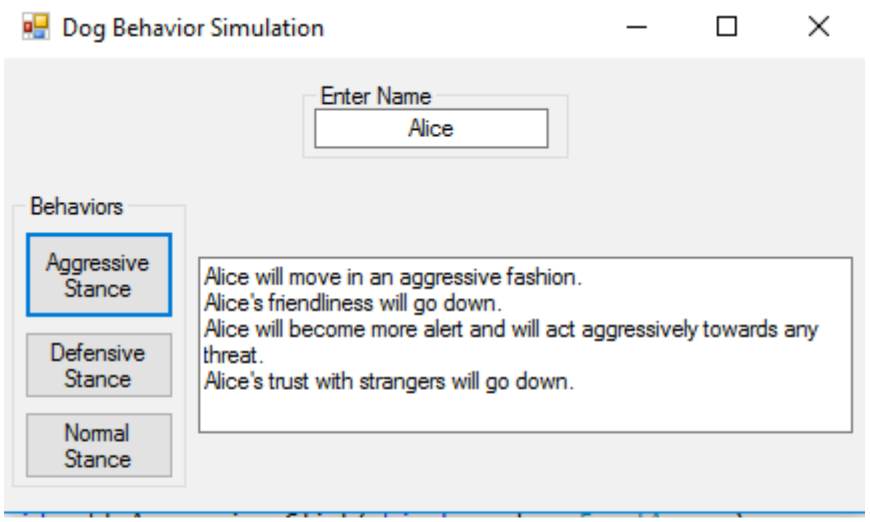
```
public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        Dog d1 = new Dog(new AggressiveBehavior());
        Dog d2 = new Dog(new DefensiveBehavior());
        Dog d3 = new Dog(new NormalBehavior());

        private void m_btnAggressive_Click(object sender, EventArgs e)
        {
            m_tbDisplay.Text = d1.move(m_tbName.Text) + System.Environment.NewLine +
d1.friendliness(m_tbName.Text) + System.Environment.NewLine + d1.alertness(m_tbName.Text)
+ System.Environment.NewLine + d1.trust(m_tbName.Text) + System.Environment.NewLine;
        }
        private void m_btnDefensive_Click(object sender, EventArgs e)
        {
            m_tbDisplay.Text = d2.move(m_tbName.Text) + System.Environment.NewLine +
d2.friendliness(m_tbName.Text) + System.Environment.NewLine + d2.alertness(m_tbName.Text)
+ System.Environment.NewLine + d2.trust(m_tbName.Text) + System.Environment.NewLine;
        }
        private void m_btnNormal_Click(object sender, EventArgs e)
        {
            m_tbDisplay.Text = d3.move(m_tbName.Text) + System.Environment.NewLine +
d3.friendliness(m_tbName.Text) + System.Environment.NewLine + d3.alertness(m_tbName.Text)
+ System.Environment.NewLine + d3.trust(m_tbName.Text) + System.Environment.NewLine;
        }
    }
```

Lastly is the client. The client initializes three separate instances of the context (Dog) class and has one of the three behaviors as a parameter for a specific context class. After this, there are three click events for the three behaviors. In the example of the aggressive behavior, if its button is pushed, the event is fired and sets the text box's text in the client to a string. This string is a combination of the four trait's strings put together (with the name of the dog in the parameter for each trait) and a new line in between each trait. This happens for each of the three behaviors.

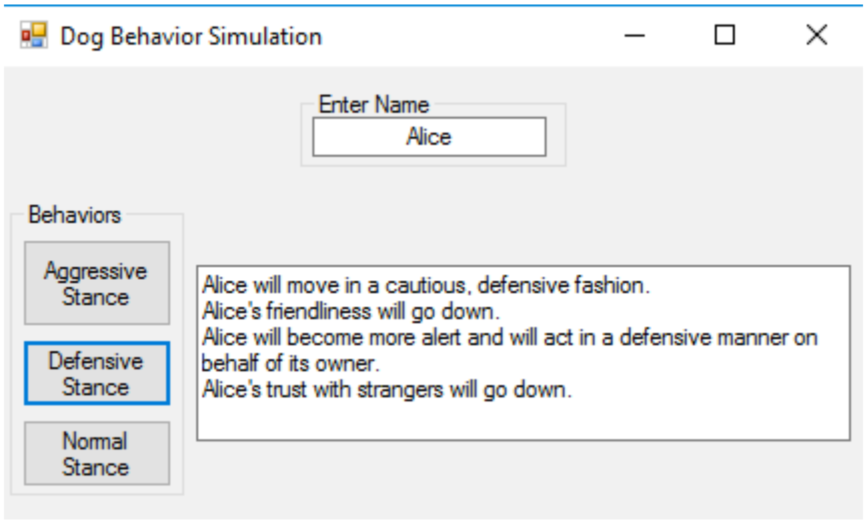
Screenshots of Application Working





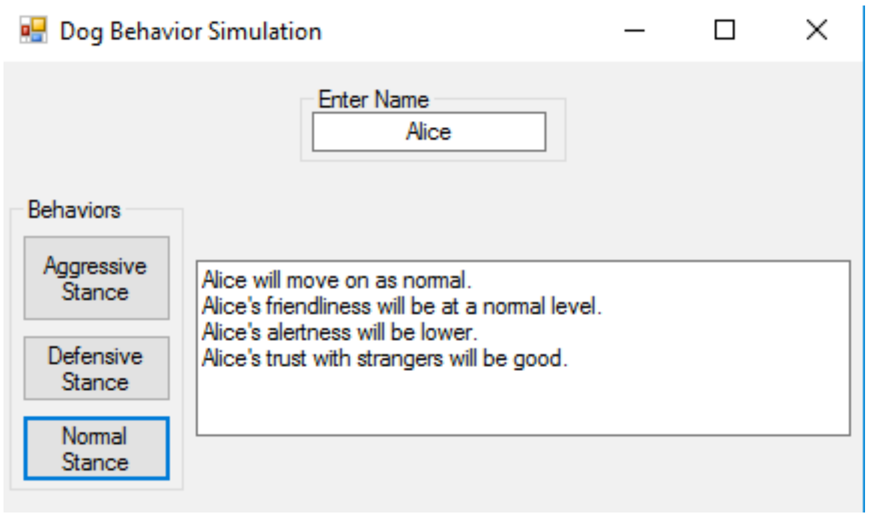
Alice

Alice will move in an aggressive fashion.
Alice's friendliness will go down.
Alice will become more alert and will act aggressively towards any threat.
Alice's trust with strangers will go down.

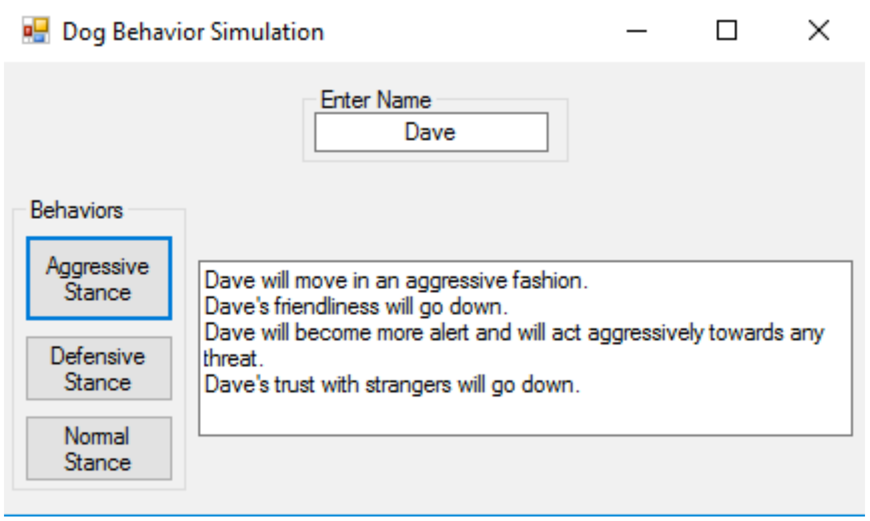


Alice

Alice will move in a cautious, defensive fashion.
Alice's friendliness will go down.
Alice will become more alert and will act in a defensive manner on behalf of its owner.
Alice's trust with strangers will go down.



```
Alice will move on as normal.
Alice's friendliness will be at a normal level.
Alice's alertness will be lower.
Alice's trust with strangers will be good.
```



```
Dave will move in an aggressive fashion.
Dave's friendliness will go down.
Dave will become more alert and will act aggressively towards any
threat.
Dave's trust with strangers will go down.
```

Conclusion and Observations

I'm satisfied to finally have completed this pattern even though I turned the assignment in late. It took me some time to figure out this pattern as I wasn't always certain I knew what I was doing, but eventually I took hold of what I needed to do and finally got it right. Even though this application is a simple one, all that matters in my eyes is that I completed the assignment with some understanding of what the strategy pattern is about (which is what I think I achieved).