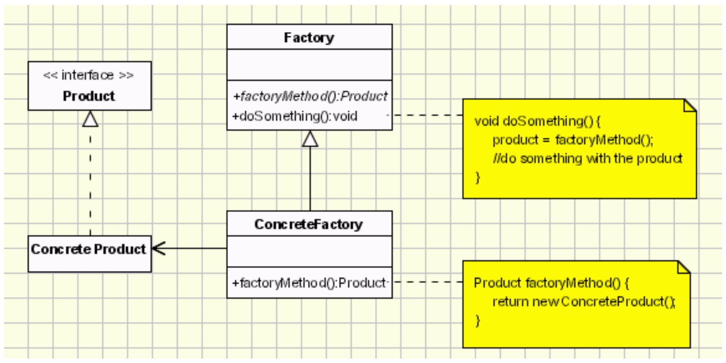
Alex Lang

Design Patterns

Factory Method Pattern

9/17/16

For this assignment, I created a program that made use of the Factory Method Pattern. The Factory Method Pattern acts in a similar way as libraries do, it creates an interface for objects to be created from its subclasses, but usually is done by user choices. The UML Diagram for the Factory Method Pattern is pictured to the right. In this diagram, it is shown that the factory method one of the abstract classes for the Factory Method. It serves to create an abstract method that will later be used by the Concrete Factory class in some similar way. The Concrete Factory class inherits from the factory class, and it is where most of the magic happens with this particular design pattern. The product in this pattern, defines the interface for the objects the factory method produces, and the concrete product class inherits from that, and defines it.



To demonstrate this pattern, I made a hero creation program, in which the user gets a few radio buttons to choose from and from those options, can customize their hero’s appearance to their liking. For this program, I had classes for the three categories you could customize your character from, by weight, hair color, and mood. I also had my Factory and Concrete Factory classes as well.

The first class I created was the abstract factory class. The code for that class is as follows.

```
public abstract class Factory     //Abstract factory class
    {

         public abstract void drawHero(Form1 form, string hair, string weight, string mood);

    }
```

In here I defined what would later get all of the information I would need to be able to draw our hero. Since it is abstract, nothing is defined here, except for the method title.

Next, I created most of the Concrete Factory Class, and this is where everything was pulled together.

```
class ConcreteFactory : Factory    //Concrete Factory class, inherits from factory.
    {

        string hair;
        string weight;
        string mood;
        Weight wt = new Weight();
        Mood md = new Mood();
        Hair hr = new Hair();

        public override void drawHero(Form1 f, string hair, string weight, string mood)
        {
            this.hair = hair;
            this.weight = weight;
            this.mood = mood;
            Graphics graphics = f.CreateGraphics();                  //Creates graphics on the main page
            Pen face = new Pen(Color.Blue, 4);

            f.CreateGraphics().Clear(Form1.ActiveForm.BackColor);
            graphics.DrawEllipse(wt.getWeight(weight), 680, 30, 80, 80);     //Draw Head
            graphics.DrawLine(wt.getWeight(weight), 720, 110, 720, 220);     //Draws the body
            graphics.DrawLine(wt.getWeight(weight), 640, 150, 800, 150);     //Draws Arms
            graphics.DrawLine(wt.getWeight(weight), 720, 218, 640, 275);     //Draws the Legs
            graphics.DrawLine(wt.getWeight(weight), 720, 218, 800, 275);

            graphics.DrawLine(face, 710, 50, 710, 70);
            graphics.DrawLine(face, 730, 50, 730, 70);                       //Draws eyes

            if (md.isHappy(mood))
            {
                graphics.DrawLine(face, 710, 80, 720, 95);                  //If happy, make happy
                graphics.DrawLine(face, 730, 80, 720, 95);
            }

            if (md.isNeutral(mood))
            {
                graphics.DrawLine(face, 710, 80, 730, 80);                 //If neutral, make neutral
            }

            if (md.isMad(mood))
            {
                graphics.DrawLine(face, 700, 40, 715, 50);                 //If mad, make mad
                graphics.DrawLine(face, 740, 40, 725, 50);
                graphics.DrawLine(face, 710, 95, 720, 80);
                graphics.DrawLine(face, 730, 95, 720, 80);
            }
            graphics.DrawLine(hr.getHair(hair), 700, 38, 700, 20);          //Creates the strands of hair
            graphics.DrawLine(hr.getHair(hair), 720, 30, 720, 12);
            graphics.DrawLine(hr.getHair(hair), 740, 38, 740, 20);

            graphics.Dispose();

        }

    }
```

The concrete factory class inherits from the factory class, and it is here that I defined the drawHero method. The drawhero method is fed in the string variables weight, hair, and mood from the form, and defines the local variables there. Not only are the string variables passed in, but the main form is passed in so we can create graphics on it. This is needed so we can write the line Graphics graphics = f.CreateGraphics(); and can establish a form that can be drawn on. The next few lines have to do with the weight class. Here is where I used the weight string, once it is passed in to this class, it is then passed into the weight class, where it is determined which Pen object to send back. The pen objects are what determine the different colors and thicknesses for this hero creation.

```
class Weight
    {
        Pen weightPen;


        public Pen getWeight(string weight)         //Returns a pen object, depends on user input
        {

            if(weight == "skinny")
            {
                weightPen = new Pen(Color.Blue, 2);
            }

            if(weight == "medium")
            {
                weightPen = new Pen(Color.Blue, 5);
            }

            if(weight == "large")
            {
                weightPen = new Pen(Color.Blue, 15);
            }

            return weightPen;
        }
    }
```

Here the class determines what Pen object to return based on the string passed in. For example, if “skinny” is passed in, the Pen object returned will be blue, and will have a width of 2 pixels.

After all of the parts of the hero that are drawn based on the weight are done, the program then moves on to the mood. This class works a little different than the Weight and Hair class. This class is different because rather than having different colors or thicknesses based on user input, the moods of the hero change entire lines being drawn on the hero.

```
public class Mood
    {
       public bool isHappy(string mood) //returns a pen object, returns true if string fills requirements.
        {
            if (mood == "happy")
                return true;
            return false;
        }

        public bool isNeutral(string mood)
        {
            if (mood == "neutral")
                return true;
            return false;
        }

        public bool isMad(string mood)
        {
            if (mood == "mad")
                return true;
            return false;
        }
    }
```

Similar to the weight class, the string value is passed in for testing. In this class however, Booleans are returned instead of Pen objects. If the string for example, is happy, the Boolean returned will be true, since it fits the required condition. Back in the concrete factory class, it is checked to see if the mood happy is true. If it isn’t, it will test to see if the next mood is true, and it will draw that instead.

Finally, I created the hair class, which is extremely similar to the weight class, in which strings are passed into the class and Pen objects are returned based on user input.

```
class Hair
    {
        Pen hairPen;

        public Pen getHair(string hair)
        {

            if (hair == "brown")                     //Determines color of hair, depending on user input.
            {
                hairPen = new Pen(Color.SaddleBrown, 3);
            }

            if (hair == "black")
            {
                hairPen = new Pen(Color.Black, 3);
            }

            if (hair == "blonde")
            {
                hairPen = new Pen(Color.Yellow, 3);
            }

            return hairPen;
        }
     }
```

For example, if the hair string sent in is “blonde”, A pen object that is yellow, and is three pixels wide will be returned.

The main form by far was the easiest part of the project, as I only had to call the one drawHero method from the Concrete Factory class. The only other coding for the most part in this form was setting string values based on radio selection in the form.

```
public partial class Form1 : Form
    {


        public string hairColor = "brown";
        public string weight = "skinny";         //Default values.
        public string mood = "happy";
        ConcreteFactory CF = new ConcreteFactory();


        public Form1()
        {
            InitializeComponent();
        }
        private void createBtn_Click(object sender, EventArgs e)
        {
            CF.drawHero(this, hairColor, weight, mood);  //Creates the hero based on user input
        }

        //Radio buttons that send in available options.

        private void brownRadio_CheckedChanged(object sender, EventArgs e)
        {
            hairColor = "brown";
        }

        private void blackRadio_CheckedChanged(object sender, EventArgs e)
        {
            hairColor = "black";
        }

        private void blondeRadio_CheckedChanged(object sender, EventArgs e)
        {
            hairColor = "blonde";
        }

        private void skinnyRadio_CheckedChanged(object sender, EventArgs e)
        {
            weight = "skinny";
        }

        private void mediumRadio_CheckedChanged(object sender, EventArgs e)
        {
            weight = "medium";
        }

        private void largeRadio_CheckedChanged(object sender, EventArgs e)
        {
            weight = "large";
        }

        private void happyRadio_CheckedChanged(object sender, EventArgs e)
        {
            mood = "happy";
        }

        private void neutralRadio_CheckedChanged(object sender, EventArgs e)
        {
            mood = "neutral";
        }

        private void madRadio_CheckedChanged(object sender, EventArgs e)
        {
            mood = "mad";
        }
    }
```

The “this” being sent in the drawHero refers to the main form that the user interacts with. This was the only way I could find to make graphics appear on the main form.

Conclusion: The concept of the whole Factory Method Pattern to me was extremely confusing for the longest time. I think that this was the fact that it was just barely talked about in class too much, and most of the information found on this pattern was online or in youtube videos. I think I made this project a lot more difficult than it needed to be with all the artwork, finding how to draw on a form from a separate class took me a few hours to figure out by itself, but I'm still very glad that I went with this route. I made sure to make this project a lot of fun, and I really enjoyed it.