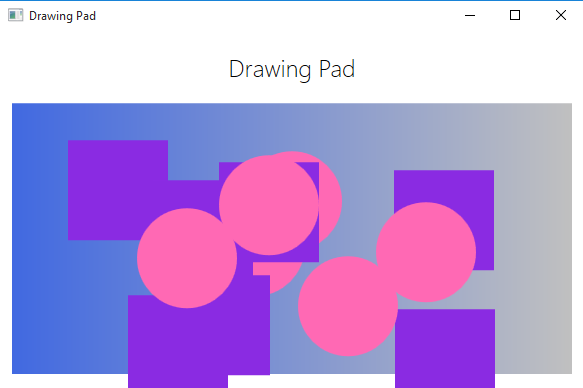
Student: Brian Johnston

Class:COP2362

Insturctur: Hamilton

I worked alone

Screenshot



CODE:

Circle.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Media;

using System.Windows.Shapes;

using System.Windows.Controls;

namespace Drawing

{

class Circle : DrawingShape, IDraw, IColor

{

public Circle(int diameter)

: base(diameter)

{

}

public override void Draw(Canvas canvas)

{

if (this.shape != null)

{

canvas.Children.Remove(this.shape);

}

else

{

this.shape = new Ellipse();

}

base.Draw(canvas);

}

}

}

DrawingPad.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Drawing

{

public partial class DrawingPadWindow : Window

{

public DrawingPadWindow()

{

InitializeComponent();

}

private void drawingCanvas\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

Point mouseLocation = e.GetPosition(this.drawingCanvas);

Square mySquare = new Square(100);

if (mySquare is IDraw)

{

IDraw drawSquare = mySquare;

drawSquare.X = (int)mouseLocation.X;

drawSquare.Y = (int)mouseLocation.Y;

drawSquare.Draw(drawingCanvas);

}

if (mySquare is IColor)

{

IColor colorSquare = mySquare;

colorSquare.Color = Colors.BlueViolet;

}

}

private void drawingCanvas\_MouseRightButtonDown(object sender, MouseButtonEventArgs e)

{

Point mouseLocation = e.GetPosition(this.drawingCanvas);

Circle myCircle = new Circle(100);

if (myCircle is IDraw)

{

IDraw drawCircle = myCircle;

drawCircle.X = (int)mouseLocation.X;

drawCircle.Y = (int)mouseLocation.Y;

drawCircle.Draw(drawingCanvas);

}

if (myCircle is IColor)

{

IColor colorCircle = myCircle;

colorCircle.Color = Colors.HotPink;

}

}

}

}

DrawingShape.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Media;

using System.Windows.Shapes;

using System.Windows.Controls;

namespace Drawing

{

abstract class DrawingShape

{

protected int \_size;

protected int \_x = 0, \_y = 0;

protected Shape shape = null;

public DrawingShape(int size)

{

this.\_size = size;

}

public int X

{

get { return this.\_x; }

set { this.\_x = value; }

}

public int Y

{

get { return this.\_y; }

set { this.\_y = value; }

}

public Color Color

{

set

{

if (this.shape != null)

{

SolidColorBrush brush = new SolidColorBrush(value);

this.shape.Fill = brush;

}

}

}

public virtual void Draw(Canvas canvas)

{

if (this.shape == null)

{

throw new InvalidOperationException("Shape is null");

}

this.shape.Height = this.\_size;

this.shape.Width = this.\_size;

Canvas.SetTop(this.shape, this.\_y);

Canvas.SetLeft(this.shape, this.\_x);

canvas.Children.Add(this.shape);

}

}

}

Icolor.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Media;

namespace Drawing

{

interface IColor

{

Color Color { set; }

}

}

Idraw.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Controls;

namespace Drawing

{

interface IDraw

{

int X { get; set; }

int Y { get; set; }

void Draw(Canvas canvas);

}

}

Square.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Media;

using System.Windows.Shapes;

using System.Windows.Controls;

namespace Drawing

{

class Square : DrawingShape, IDraw, IColor

{

public Square(int sideLength) : base(sideLength)

{

}

public override void Draw(Canvas canvas)

{

if (this.shape != null)

{

canvas.Children.Remove(this.shape);

}

else

{

this.shape = new Rectangle();

}

base.Draw(canvas);

}

}

}