COS20007 - Object Oriented Programming

4.1P - Drawing Program - Multiple shape kinds with your own attributes

Student name: Nguyen Duc Manh
ID: 105547489

```
using System;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Program
    {
        enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }

        public static void Main()
        {
            ////the first Shape is the type of object, which is the class i've
              made earlier
            //Shape myShape = new Shape(); //Shape() to call the constructor
            Window window = new Window("Shape Drawer", 800, 600);
            Drawing myDrawing = new Drawing();
            ShapeKind kindToAdd = ShapeKind.Circle;
            do
            {
                //8.4
                if (SplashKit.KeyTyped(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
                }
                else if (SplashKit.KeyTyped(KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle;
                }
                else if (SplashKit.KeyTyped(KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line;
                }

                //earlier code
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();
                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape newShape = null;
                    var lines = myDrawing.AllShapes.OfType<MyLine>().ToList();
                    int linesCount = lines.Count;
                    switch (kindToAdd)
                    {
                        case ShapeKind.Circle:
```

```
                            newShape = new MyCircle();
                            break;
                        case ShapeKind.Line:
                            if (linesCount < 9)
                            {
                                float X = SplashKit.MouseX();
                                float Y = SplashKit.MouseY();
                                newShape = new MyLine(Color.Red, X, Y, X +
                                  100, Y);
                            }
                            break;
                        default:
                            newShape = new MyRectangle();
                            break;

                    }
                    if (newShape != null)
                    {
                        newShape.X = SplashKit.MouseX();
                        newShape.Y = SplashKit.MouseY();
                        myDrawing.AddShape(newShape);
                    }
                }
                if (SplashKit.KeyDown(KeyCode.SpaceKey))
                {
                    myDrawing.Background = Color.RandomRGB(255);
                }
                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    myDrawing.SelectShapeAt(SplashKit.MousePosition());
                }
                if (SplashKit.KeyDown(KeyCode.DeleteKey) || SplashKit.KeyDown
                  (KeyCode.BackspaceKey))
                {
                    foreach (Shape newShape in myDrawing.SelectedShapes)
                    {
                        myDrawing.RemoveShape(newShape);
                    }
                }
                myDrawing.Draw();
                SplashKit.RefreshScreen();
            } while (!window.CloseRequested);
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

namespace ShapeDrawer
{
    public abstract class Shape
    {
        //private fields
        Color _color;
        float _x, _y;
        bool _selected; //bool field is "false" by default

        public Shape(): this(Color.Yellow) //Constructor
        {
            //other steps
        }

        public Shape(Color color) //Overloaded constructor
        {
            _color = color;
            _x = 10;
            _y = 10;
        }
        //Properties
        public Color FillColor
        {
            get
            {
                return _color;
            }
            set
            {
                _color = value;
            }
        }

        public float X
        {
            get
            {
                return _x;
            }
            set
            {
```

```
                _x = value;
            }
        }

        public float Y
        {
            get
            {
                return _y;
            }
            set
            {
                _y = value;
            }
        }

        public bool Selected
        {
            get { return _selected; }
            set { _selected = value; }
        }

        //methods
        public abstract void Draw();

        public abstract bool IsAt(Point2D pt);

        public abstract void DrawOutLine();
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Drawing
    {
        readonly List<Shape> _shapes;
        Color _background;

        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;
        }

        public Drawing() : this(Color.White) //default constructor -
          initializes objs with predefined values
        {
            //other steps
        }

        //methods
        public void AddShape(Shape shape)
        {
            _shapes.Add(shape);
        }

        public void RemoveShape(Shape shape)
        {
            _ = _shapes.Remove(shape); //to discard the value it returns
        }

        public void Draw()
        {
            SplashKit.ClearScreen(_background);
            foreach (Shape shape in _shapes)
            {
                shape.Draw();
            }
        }

        public void SelectShapeAt(Point2D pt)
        {
            foreach (Shape shape in _shapes)
```

```
            {
                shape.Selected = shape.IsAt(pt);
            }
        }

        //properties
        public Color Background
        {
            get
            {
                return _background;
            }
            set
            {
                _background = value;
            }
        }

        public int ShapeCount => _shapes.Count;

        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> result = new List<Shape>();
                foreach (Shape shape in _shapes)
                {
                    if (shape.Selected)
                    {
                        result.Add(shape);
                    }
                }
                return result;
            }
        }

        public List<Shape> AllShapes
        {
            get
            {
                return _shapes;
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class MyCircle : Shape //Shape is the base class
    {
        int _radius = 50;

        //constructor
        public MyCircle(): this(Color.Blue, 50 + 1) //SWH02701
        {
            //other steps
        }

        public MyCircle(Color color, int radius) : base(color)
        {
            _radius = radius;
        }

        //method
        public override void Draw()
        {
            if (Selected)
            {
                DrawOutLine();
            }
            SplashKit.FillCircle(FillColor, X, Y, _radius);
        }

        public override void DrawOutLine()
        {
            SplashKit.DrawCircle(Color.Black, X, Y, _radius + 5);
        }

        public override bool IsAt(Point2D pt)
        {
            double distance = SplashKit.PointPointDistance(pt, new Point2D()
              { X = this.X, Y = this.Y });
            if (distance <= _radius)
            {
                return true;
            }
            return false;
        }
```

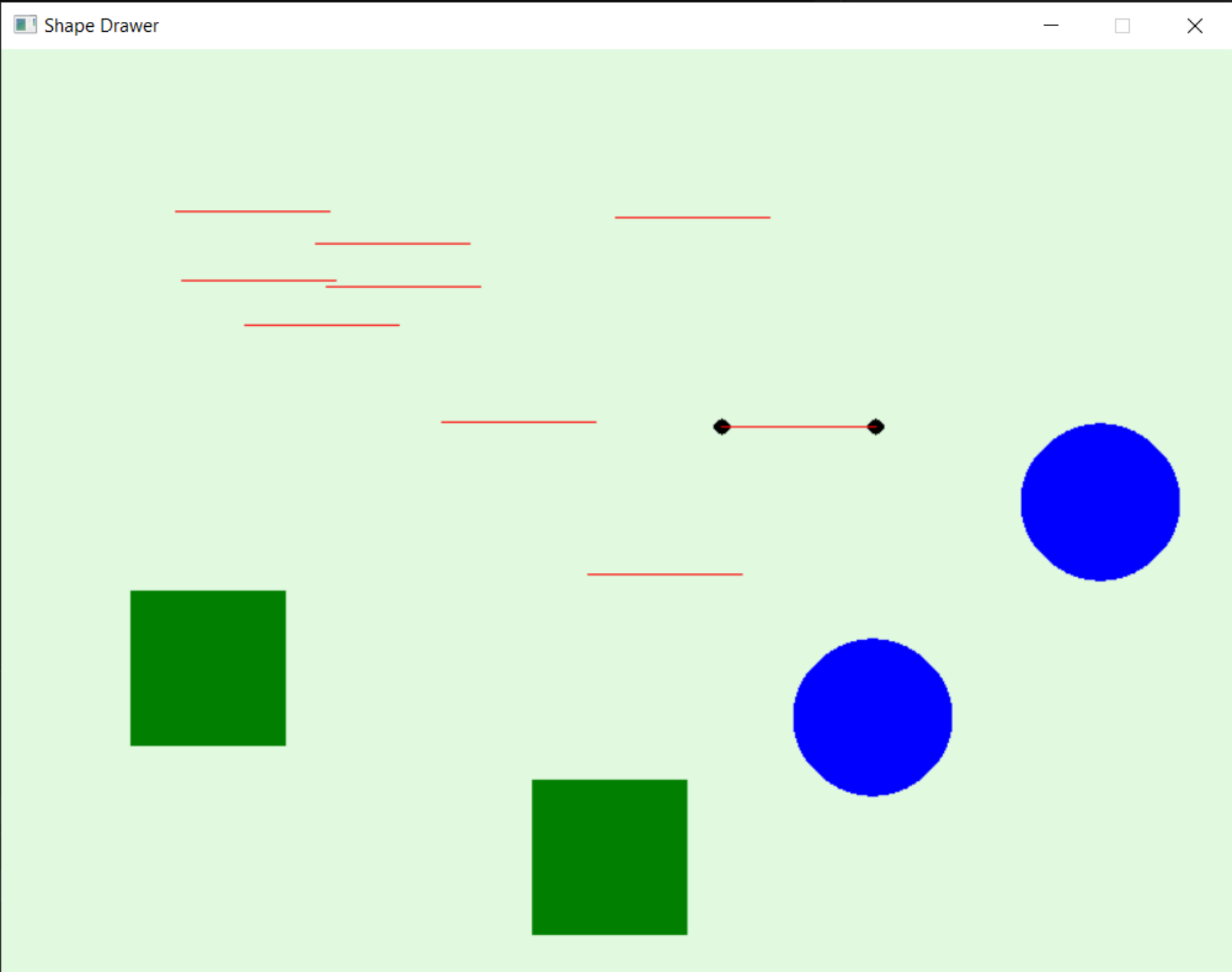
```
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class MyRectangle : Shape //Shape is the base class
    {
        int _width, _height;

        //constructor

        public MyRectangle(): this(Color.Green, 0.0f, 0.0f, 101, 101) //
          SWH02701
        {
            //other steps
        }

        public MyRectangle(Color color, float x, float y, int width, int
          height) : base(color)
        {
            Width = width;
            Height = height;
            X = x; //X Y belongs to the Shape class
            Y = y;
        }

        //method
        public override void Draw()
        {
            if (Selected)
            {
                DrawOutLine();
            }
            SplashKit.FillRectangle(FillColor, X, Y, Width, Height);
        }

        public override void DrawOutLine()
        {
            SplashKit.DrawRectangle(Color.Black, X - 7, Y - 7, _width + 14,
              _height + 14); //105547489
        }

        public override bool IsAt(Point2D pt)
        {
            return SplashKit.PointInRectangle(pt, SplashKit.RectangleFrom(X,
```

```
                Y, _width, _height));
        }


        //properties
        public int Width
        {
            get { return _width; }
            set { _width = value; }
        }

        public int Height
        {
            get { return _height; }
            set { _height = value; }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class MyLine : Shape
    {
        float _endX, _endY;

        public MyLine()
        {
            FillColor = Color.Red;
        }

        public MyLine(Color color, float startX, float startY, float endX,
          float endY)
        {
            FillColor = color;
            X = startX;
            Y = startY;
            _endX = endX;
            _endY = endY;
        }

        public float EndX
        {
            get { return _endX; }
            set { _endX = value; }
        }

        public float EndY
        {
            get { return _endY; }
            set { _endY = value; }
        }

        public override void Draw()
        {
            if (Selected)
            {
                DrawOutLine();
            }
            SplashKit.DrawLine(FillColor, X, Y, _endX, _endY);
        }
```

```
        public override void DrawOutLine()
        {
            SplashKit.FillCircle(Color.Black, X, Y, 5);
            SplashKit.FillCircle(Color.Black, _endX, _endY, 5);
        }

        public override bool IsAt(Point2D pt)
        {
            double distance1 = SplashKit.PointPointDistance(pt, new Point2D()
              { X = X, Y = Y });
            double distance2 = SplashKit.PointPointDistance(pt, new Point2D()
              { X = _endX, Y = _endY });
            double result = distance1 + distance2;
            if ((int)result == 100)
            {
                return true;
            } return false;
        }
    }
}
```


Shape Drawer