COS20007 - Object Oriented Programming

5.3C - Drawing Program — Saving and Loading with Customized Payload

Student name: Nguyen Duc Manh
ID: 105547489

SWIN BUR NE

SWINBURNE UNIVERSITY OF TECHNOLOGY

```
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);
                    }
                }
                if (SplashKit.KeyTyped(KeyCode.SKey))
                {
                    myDrawing.Save(@"C:\Users\Bill\Desktop\COS20007\Week 5
                      \ShapeDrawer(Week5)\TextDrawing.txt");
                }
                if (SplashKit.KeyTyped(KeyCode.OKey))
                {
                    try
                    {
                        myDrawing.Load(@"C:\Users\Bill\Desktop\COS20007
                          \Week 5\ShapeDrawer(Week5)\TextDrawing.txt");
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine("Error loading file: {0}" +
                          e.Message);
                    }
                }
                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;
using System.IO;

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();

        public virtual void SaveTo(StreamWriter writer)
        {
            writer.WriteColor(_color);
            writer.WriteLine(X);
            writer.WriteLine(Y);
            //writer.WriteLine(Selected);
        }

        public virtual void LoadFrom(StreamReader reader)
        {
            FillColor = reader.ReadColor();
            X = reader.ReadInteger();
            Y = reader.ReadInteger();
        }
    }
}
```

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

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);
            }
        }

        public void Save(string filename)
        {
            StreamWriter writer = new StreamWriter(filename);

            try
            {
                writer.WriteColor(_background);
                writer.WriteLine(_shapes.Count);

                foreach (Shape s in _shapes)
                {
                    s.SaveTo(writer);
                }
            }
            finally
            {
                writer.Close();
            }
        }

        public void Load(string filename)
        {
            StreamReader reader = new StreamReader(filename);
            int count;
            string kind;
            Shape s;

            try
            {
                Background = reader.ReadColor();
                count = reader.ReadInteger();

                _shapes.Clear(); //clear the list before loading new
                  shapes

                for (int i = 0; i < count; i++)
                {
                    kind = reader.ReadLine();
                    if (kind == "Rectangle")
                    {
                        s = new MyRectangle();
                    }
                    else if (kind == "Circle")
                    {
                        s = new MyCircle();
                    }
                    else if (kind == "Line")
                    {
                        s = new MyLine();
                    }
                    else
                    {
                        throw new Exception("Unknown shape type: " +
                          kind);
                    }
                    s.LoadFrom(reader);
                    _shapes.Add(s);
                }
            }
            finally
            {
                reader.Close();
            }
        }

        //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;
        }

        public override void SaveTo(StreamWriter writer)
        {
            writer.WriteLine("Circle");
            base.SaveTo(writer);
            writer.WriteLine(X);
            writer.WriteLine(Y);
            writer.WriteLine(_radius);
        }

        public override void LoadFrom(StreamReader reader)
        {
            base.LoadFrom(reader);
            X = reader.ReadInteger();
            Y = reader.ReadInteger();
            _radius = reader.ReadInteger();
        }
    }
}

```

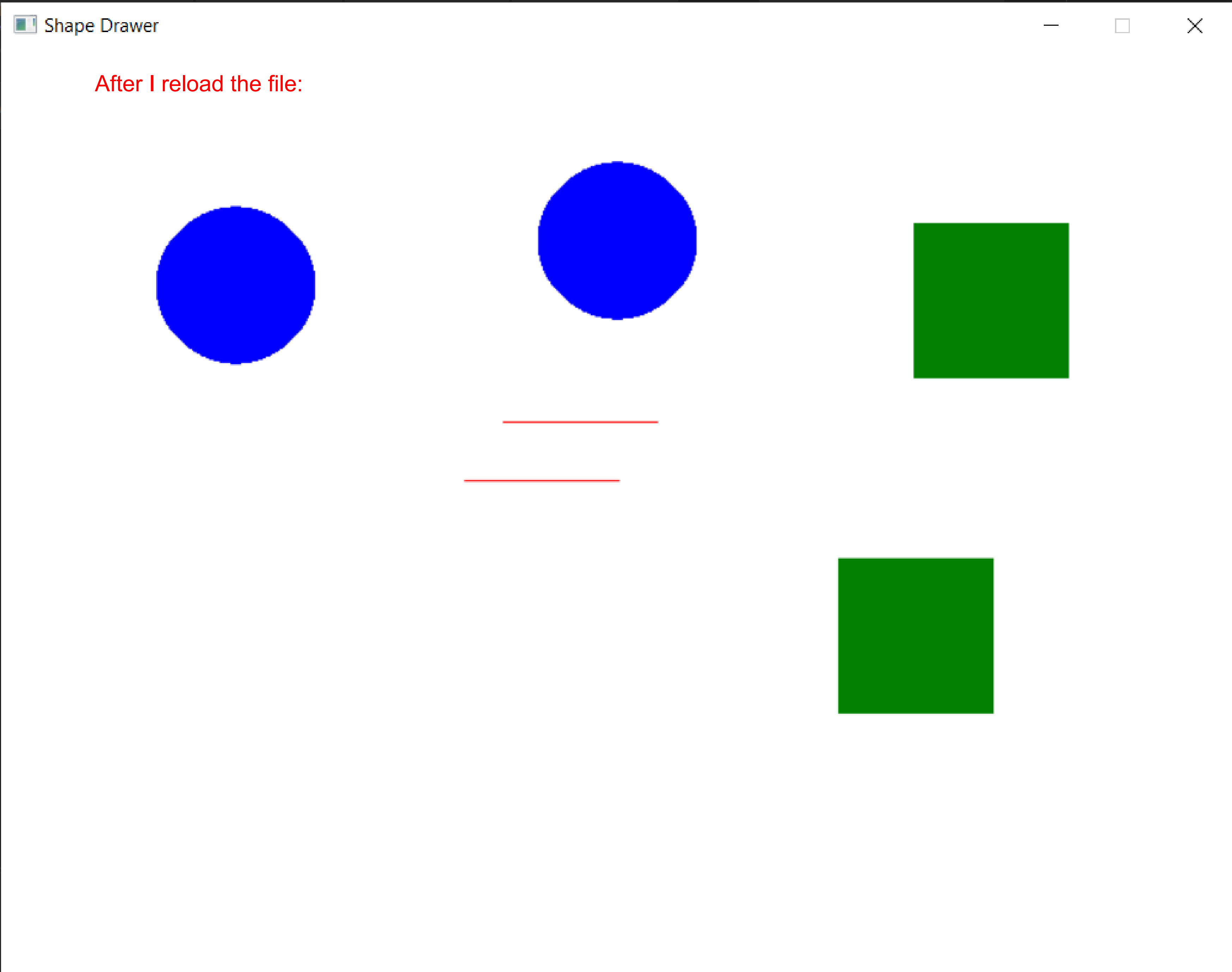
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;

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;
        }

        public override void SaveTo(StreamWriter writer)
        {
            writer.WriteLine("Line");
            base.SaveTo(writer);
            writer.WriteLine(X);
            writer.WriteLine(Y);
            writer.WriteLine(_endX);
            writer.WriteLine(_endY);
        }

        public override void LoadFrom(StreamReader reader)
        {
            base.LoadFrom(reader);
            X = reader.ReadInteger();
            Y = reader.ReadInteger();
            EndX = reader.ReadInteger();
            EndY = reader.ReadInteger();
        }
    }
}
```

```
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));
        }

        public override void SaveTo(StreamWriter writer)
        {
            writer.WriteLine("Rectangle");
            base.SaveTo(writer); //tell the base class to not be overriden
            writer.WriteLine(_width);
            writer.WriteLine(_height);
        }

        public override void LoadFrom(StreamReader reader)
        {
            base.LoadFrom(reader);
            Width = reader.ReadInteger();
            Height = reader.ReadInteger();
        }

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

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

```

```
using System;
using System.IO;
using SplashKitSDK;

namespace ShapeDrawer
{
    public static class ExtensionMethods
    {
        public static int ReadInteger(this StreamReader reader)
        {
            return Convert.ToInt32(reader.ReadLine());
        }
        public static float ReadSingle(this StreamReader reader)
        {
            return Convert.ToSingle(reader.ReadLine());
        }
        public static Color ReadColor(this StreamReader reader)
        {
            return Color.RGBColor(reader.ReadSingle(), reader.ReadSingle(),
            reader.ReadSingle());
        }
        public static void WriteColor(this StreamWriter writer, Color clr)
        {
            writer.WriteLine("{0}\n{1}\n{2}", clr.R, clr.G, clr.B);
        }
    }
}

```

```
1
1
1
6
Rectangle
0
0.49803922
0
593
113
101
101
Rectangle
0
0.49803922
0
544
331
101
101
Circle
0
0
1
152
153
152
153
51
Circle
0
0
1
400
124
400
124
51
Line
1
0
0
326
242
326
242
426
242
Line
```

```
1
0
0
301
280
301
280
401
280

```



After I reload the file: