Selected files

6 printable files

Drawing.cs
ExetnsionMethods.cs
MyLine.cs
MyRectangle.cs
Program.cs
Shape.cs

Drawing.cs

```
using System;
using System.IO;
using System.Text;
using SplashKitSDK;
using System.Collections.Generic;


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

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

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

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

        }

        public void SelectShapesAt(Point2D pt)
        {
            foreach (Shape s in _shapes)
            {
                if (s.IsAt(pt))
                    s.Selected = true;
                else
                    s.Selected = false;
            }
        }

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

        public int ShapeCount
        {
            get { return _shapes.Count; }
        }

        public void AddShape(Shape s)
        {
            _shapes.Add(s);
        }

        public void RemoveShape(Shape s)
        {
            _shapes.Remove(s);
        }

        public void Save(string filename)
        {
            StreamWriter writer = new StreamWriter(filename);
            writer.WriteColor(_background);
            writer.WriteLine(ShapeCount);

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

            writer.Close();
        }

        public void Load(string filename)
        {
            StreamReader reader = new StreamReader(filename);
            _background = reader.ReadColor();
            int count = reader.ReadInteger();

            _shapes.Clear();
            try
            {
                Shape s;

                for (int i = 0; i < count; i++)
                {
                    string kind = reader.ReadLine();

                    switch (kind)
                    {
                        case "Rectangle":
                            s = new MyRectangle();
                            break;
                        case "Circle":
                            s = new MyCircle();
                            break;
                        case "Line":
                            s = new MyLine();
                            break;
                        default:
                            throw new InvalidDataException("Unknown shape kind: " + kind);
                    }
                    s.LoadFrom(reader);
                    AddShape(s);
                }
            }
            finally
            {
                reader.Close();
            }
        }
    }
}


```

ExetnsionMethods.cs

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

namespace Drawing_Program__Saving_and_Loading
{
    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);
        }
    }
}


```

MyLine.cs

```
using System;
using SplashKitSDK;

namespace Drawing_Program__Saving_and_Loading
{
    public class MyLine : Shape
    {
        private float _endX;
        private float _endY;
        private int _thickness;

        public MyLine(Color color, float endX, float endY, int thickness) : base(color)
        {
            _endX = endX;
            _endY = endY;
            _thickness = thickness;
        }

        public MyLine() : this(Color.Black, 0, 0, 0)
        {

        }

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

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

        public int Thickness
        {
            get { return _thickness; }
            set { _thickness = value; }
        }

        public override void Draw()
        {
            if (Selected)
                DrawOutline();
            SplashKit.DrawLine(Color, X, Y, _endX, _endY);
        }

        public override void DrawOutline()
        {

        }

        public override bool IsAt(Point2D pt)
        {

            float minX = Math.Min(X, _endX) - _thickness / 2;
            float minY = Math.Min(Y, _endY) - _thickness / 2;
            float maxX = Math.Max(X, _endX) + _thickness / 2;
            float maxY = Math.Max(Y, _endY) + _thickness / 2;


            return pt.X >= minX && pt.X <= maxX && pt.Y >= minY && pt.Y <= maxY;
        }

        public override void SaveTo(StreamWriter writer)
        {
            writer.WriteLine("Line");
            base.SaveTo(writer);
            writer.WriteLine(EndX);
            writer.WriteLine(EndY);
        }

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



```

MyRectangle.cs

```
using System;
using SplashKitSDK;

namespace Drawing_Program__Saving_and_Loading
{
     public class MyRectangle : Shape
     {
          private int _width;
          private int _height;

          public MyRectangle(Color color, float x, float y, int width, int height)
: base(color)
          {
            X = x;
            Y = y;
            _width = width;
            _height = height;
        }

        public MyRectangle() : this(Color.Green, 0, 0, 100, 100)
        {

        }

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

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

        public override void Draw()
        {
            SplashKit.FillRectangle(Color, X, Y, Width, Height);
        }

        public override void DrawOutline()
        {
            SplashKit.FillRectangle(Color.Black, X, Y, Width + 2, Height + 2);
        }

        public override bool IsAt(Point2D pt)
        {
            return pt.X >= X && pt.X <= X + Width && pt.Y >= Y && pt.Y <= Y +
Height;
        }

        public override void SaveTo(StreamWriter writer)
        {
            writer.WriteLine("Rectangle");
            base.SaveTo(writer);
            writer.WriteLine(Width);
            writer.WriteLine(Height);
        }

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


```

Program.cs

```
using System;
using SplashKitSDK;

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

        public static void Main()
        {
            Window window = new Window("Multiple Shape", 800, 600);
            Drawing myDrawing = new Drawing();

            ShapeKind kindToAdd = ShapeKind.Circle;
            do
            {
                SplashKit.ProcessEvents();

                SplashKit.ClearScreen();

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape newShape;
                    switch (kindToAdd)
                    {
                        case ShapeKind.Circle:
                            newShape = new MyCircle();
                            newShape.X = SplashKit.MouseX();
                            newShape.Y = SplashKit.MouseY();
                            myDrawing.AddShape(newShape);
                            break;

                        case ShapeKind.Rectangle:
                            newShape = new MyRectangle();
                            newShape.X = SplashKit.MouseX();
                            newShape.Y = SplashKit.MouseY();
                            myDrawing.AddShape(newShape);
                            break;

                        case ShapeKind.Line:
                            newShape = new MyLine();
                            newShape.X = SplashKit.MouseX();
                            newShape.Y = SplashKit.MouseY();
                            myDrawing.AddShape(newShape);
                            break;
                    }
                }

                if (SplashKit.KeyDown(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle; // Press 'R' to draw rectangles
                }

                if (SplashKit.KeyDown(KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle; // Press 'C' to draw circles
                }

                if (SplashKit.KeyDown(KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line; // Press 'L' to draw lines
                }

                if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                {
                    myDrawing.Background = SplashKit.RandomRGBColor(255);
                }

                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    myDrawing.SelectShapesAt(SplashKit.MousePosition());
                }

                if (SplashKit.KeyDown(KeyCode.DeleteKey) || SplashKit.KeyDown(KeyCode.BackspaceKey))
                {
                    foreach (Shape s in myDrawing.SelectedShapes)
                    {
                        myDrawing.RemoveShape(s);
                    }
                }

                if (SplashKit.KeyDown(KeyCode.SKey))
                {
                    string filePath = "/Users/thuanduc/Documents/thuan's folder/work/COS20007/week 5/5.3/TextDrawing.txt";
                    myDrawing.Save(filePath); // press "S" to save the Drawing
                }
                if (SplashKit.KeyDown(KeyCode.OKey))
                {
                    try
                    {
                        string filePath = "/Users/thuanduc/Documents/thuan's folder/work/COS20007/week 5/5.3/TextDrawing.txt";
                        myDrawing.Load(filePath);  // press "O" to loading the test Drawing file.
                    } catch (Exception e)
                    {
                        Console.Error.WriteLine("Error loading file: {0}", e.Message);
                    }
                }
                myDrawing.Draw();

                SplashKit.RefreshScreen();
            } while (!window.CloseRequested);
        }
    }
}

```

Shape.cs

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

namespace Drawing_Program__Saving_and_Loading
{
    public abstract class Shape
    {
        private Color _color;
        private float _x;
        private float _y;
        private bool _selected;

        public Shape(Color color)
        {
            _color = color;
        }

        public Color Color
        {
            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; }
        }

        public abstract void Draw();
        public abstract void DrawOutline();
        public abstract bool IsAt(Point2D pt);

        public virtual void SaveTo(StreamWriter writer)
        {
            writer.WriteColor(_color);
            writer.WriteLine(_x);
            writer.WriteLine(_y);
        }

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

}
}
```