Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Saving and Loading

PDF generated at 03:01 on Saturday 30th September, 2023

Produced by OOP Submission Formatter

```
using System;
using SplashKitSDK;
namespace Task5._3
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle, Circle, Line
        }

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

            Drawing dr = new Drawing();
            ShapeKind kindToAdd = ShapeKind.Circle;
            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape newShape;
                    if (kindToAdd == ShapeKind.Circle)
                    {
                        newShape = new MyCircle();
                    }
                    else if (kindToAdd == ShapeKind.Rectangle)
                    {
                        newShape = new MyRectangle();
                    }
                    else
                    {
                        newShape = new MyLine();
                    }

                    newShape.X = SplashKit.MouseX();

                    newShape.Y = SplashKit.MouseY();

                    dr.AddShape(newShape);
                }

                if (SplashKit.KeyTyped(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
                }

                if (SplashKit.KeyTyped(KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle;
                }
                if (SplashKit.KeyTyped(KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line;
                }
                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {

                    dr.SelectShapesAt(SplashKit.MousePosition());
                }

                if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                {

                    dr.Background1 = SplashKit.RandomRGBColor(255);
                }

                if (SplashKit.KeyTyped(KeyCode.DeleteKey) ||
↪   SplashKit.KeyTyped(KeyCode.BackspaceKey))
                {

                    foreach (Shape s in dr.Selectedshapes)
                    {
                        dr.DeleteShape(s);
                    }
                }
                if (SplashKit.KeyTyped(KeyCode.SKey))
                {
                    dr.Save("Desktop\\TestDrawing.txt");
                }
                if (SplashKit.KeyTyped(KeyCode.OKey))
                {
                    try
                    {
                        dr.Load("Desktop\\TestDrawing.txt");
                    }
                    catch(Exception e)
                    {
                        Console.Error.WriteLine("Error Loadinhg file: {0}",
↪   e.Message);
                    }
                }



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

}

```

```
using System;
using SplashKitSDK;
namespace Task5._3
{
    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);
        }
    }
}

```

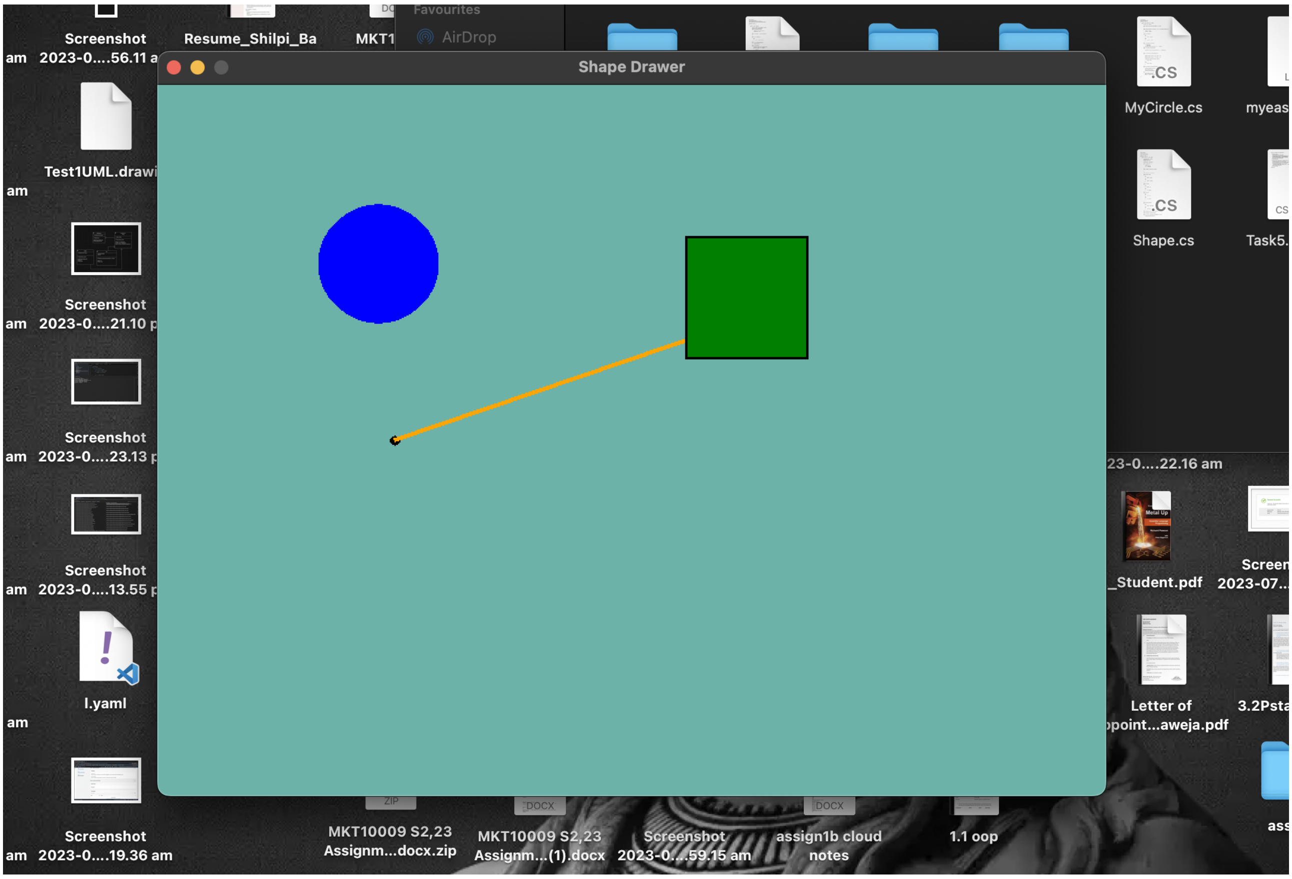
```
using System;
using SplashKitSDK;
using System.Collections.Generic;
using System.Reflection.PortableExecutable;

namespace Task5._3

{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;
        public Drawing(Color bg)
        {
            _shapes = new List<Shape>();
            _background = bg;

        }
        public Drawing() : this(Color.White)
        {

        }
        public int ShapeCount
        {
            get
            {
                return _shapes.Count;
            }
        }
        public void AddShape(Shape s)
        {
            _shapes.Add(s);
        }
        public void Draw()
        {
            SplashKit.ClearScreen(Background1);
            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> _result = new List<Shape>();


                foreach (Shape s in _shapes)
                {
                    if (s.Selected == true)
                    {
                        _result.Add(s);
                    }
                }
                return _result;
            }
        }
        public Color Background1
        {
            get
            {
                return _background;
            }
            set
            {
                _background = value;
            }
        }

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

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

            writer.WriteColor(Background1);
            writer.WriteLine(ShapeCount);

            foreach(Shape s in _shapes)
            {
                s.SaveTo(writer);
            }
            writer.Close();
        }
        public void Load(string filename)
        {
            StreamReader reader = new StreamReader(filename);
            try {

            int count;
            Shape s;
            string kind;
            Background1 = reader.ReadColor();
            count = reader.ReadInteger();

            _shapes.Clear();

                for (int i = 0; i < count; i += 1)
                {
                    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();
            }

        }
    }
}
```

```
using System;

using SplashKitSDK;

namespace Task5._3
{
    public abstract class Shape
    {
        protected Color _color;
        private float _x;
        private float _y;

        private bool _selected;

        public Shape(Color c)
        {
            _color = c;
            _x = (float)0;
            _y = (float)0;

        }
        public Shape():this(Color.Yellow)
        {

        }

        public abstract void Draw();

        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 bool IsAt(Point2D pt);

        public abstract void Outline();

        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();
        }
    }
}
```

```csharp
using System;
using SplashKitSDK;

namespace Task5._3
{
    public class MyRectangle:Shape
    {
        private int _width;
        private int _height;
        public MyRectangle():this(Color.Green, 0, 0, 100, 100)
        {

        }
        public MyRectangle(Color clr, float x, float y, int width, int height):base(clr)
        {

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

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

        }
        public override void Draw()
        {
            if (Selected == true)
            {
                Outline();
            }
            SplashKit.FillRectangle(Color, X, Y, _width, _height);

        }
        public override void Outline()
        {
            SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, _width + 4, _height + 4);
        }

        public override bool IsAt(Point2D pt)
        {

                if (pt.X >= X && pt.X < (X + _width) && pt.Y >= Y && pt.Y <= (Y + _height))
            {

                return true;
            }
            else
            {
                return false;
            }
        }
        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();
        }
    }
}

```

```
using System;
using SplashKitSDK;
namespace Task5._3
{
    public class MyCircle : Shape
    {
        private int _radius;

        public MyCircle():this(Color.Blue, 0, 0, 50)
        {

        }
        public MyCircle(Color clor,int x, int Y, int radius):base(clor)
        {
            _color = clor;
            _radius = radius;
        }

        public int Radius
        {
            get
            {
                return _radius;
            }
            set
            {
                _radius = value;
            }

        }
        public override void Draw()
        {
            if (Selected)
                Outline();
            SplashKit.FillCircle(Color, X, Y, _radius);
        }
        public override void Outline()
        {

            SplashKit.FillCircle(Color.Black, X , Y , Radius+2);


        }
        public override bool IsAt(Point2D pt)
        {

            double point1 = (pt.X - X) * (pt.X - X);

            double point2 = (pt.Y - Y) * (pt.Y - Y);

            if(Math.Sqrt(point1+point2)<_radius)
            {
                return true;
            }
            else
            {
                return false;
            }

        }
        public override void SaveTo(StreamWriter writer)
        {
            writer.WriteLine("Circle");

            base.SaveTo(writer);
            writer.WriteLine(Radius);
        }
        public override void LoadFrom(StreamReader reader)
        {
            base.LoadFrom(reader);
            Radius = reader.ReadInteger();
        }
    }
}

```

```
using System;
using System.Numerics;
using SplashKitSDK;
namespace Task5._3
{
    public class MyLine : Shape
    {
        private float _endX;
        private float _endY;


        public MyLine() : this(Color.Orange, 0, 0, 200, 300)
        {

        }
        public MyLine(Color clr, float startX, float startY, float endX, float endY) : base(clr)
        {

            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)
                Outline();
            SplashKit.DrawLine(Color, X, Y, EndX, EndY, SplashKit.OptionLineWidth(5));
        }
        public override void Outline()
        {


            SplashKit.FillCircle(Color.Black, X, Y, 4);
            SplashKit.FillCircle(Color.Black, EndX, EndY, 4);

        }
        public override bool IsAt(Point2D pt)
        {


            Line l = SplashKit.LineFrom(X, Y, EndX, EndY);


            return SplashKit.PointOnLine(pt, l, 10);
        }
        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();
        }
    }
}
```


Shape Drawer
Screenshot 2023-0....56.11 a
Resume_Shilpi_Ba
MKT1
Favourites
AirDrop
MyCircle.cs
myeas
Shape.cs
Task5.
Test1UML.drawi
Screenshot 2023-0....21.10 p
Screenshot 2023-0....23.13 p
Screenshot 2023-0....13.55 p
l.yaml
Screenshot 2023-0....19.36 am
23-0....22.16 am
_Student.pdf
Screen 2023-07...
Letter of ppoint...aweja.pdf
3.2Psta
ass
MKT10009 S2,23 Assignm...docx.zip
MKT10009 S2,23 Assignm...(1).docx
Screenshot 2023-0....59.15 am
assign1b cloud notes
1.1 oop