5.2C Drawing Programs Shape Drawer

Object-Oriented Programming (Swinburne University of Technology)

Swinburne University of Technology

Object Oriented Programming (2021 S1)

Doubtfire Submission

Credit Task 5.2: Drawing Program - Saving

Submitted By:
Uthpala Harshani Bellanage
102625094
2021/04/20 14:38

Tutor:
Matt Noone

April 20, 2021

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

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

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

                foreach(Shape s in _shapes)
                {
                    if (s.Selected == true)
                        result.Add(s);
                }
                return result;
            }
        }
        public int ShapeCount
        {
            get { return _shapes.Count; }
        }
        public Color Background
        {
            get { return _background; }
            set { _background = value; }
        }
        public void Draw()
        {
            //SplashKit.ClearScreen(_background);

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

            }
        }
        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);
            //Shape s;
            try
            {
                writer.WriteLine(Background);
                writer.WriteLine(ShapeCount);

                foreach (Shape s in _shapes)
                {
                    s.SaveTo(writer);
                }
            }
            finally
            {
                writer.Close();
            }
        }
        public void Load(string filename)
        {
            //StreamReader reader;
            int count;
            Shape s;
            string kind;

            StreamReader reader = new StreamReader(filename);
            Background = reader.ReadColor();
            count = reader.ReadInteger();

            _shapes.Clear();

            try
            {
                for (int i = 0; i < count; i++)
                {
                    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 shaoe kind: " +
                            ↪  kind);
                    }

                    s.LoadFrom(reader);
                    _shapes.Add(s);
                }
            }
            finally
            {
                reader.Close();
            }
        }
    }
}
```

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

namespace ShapeDrawer
{
    public abstract class Shape
    {
        private Color _color;
        private float _x, _y;
        private bool _selected;
        public Shape(Color color)
        {
            this._color = color;
        }
        public Shape() : this(Color.Yellow)
        {

        }
        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();
        }
    }
}
```

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

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

        public MyRectangle(Color clr, float x, float y, int width, int height) :
         ↪  base(clr)
        {
            this.X = x;
            this.Y = y;
            this._width = width;
            this._height = height;
        }
        public MyRectangle() : this(Color.Green, 0, 0, 100, 100)
        {

        }
        public override void Draw()
        {
            SplashKit.FillRectangle(color, X, Y, _width, _height);

            if (Selected)
            {
                DrawOutline();
            }
        }
        public override void DrawOutline()
        {
            SplashKit.DrawRectangle(Color.Black, X-4, Y-4, _width+8, _height+8);
        }
        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();
        }
    }
}
```

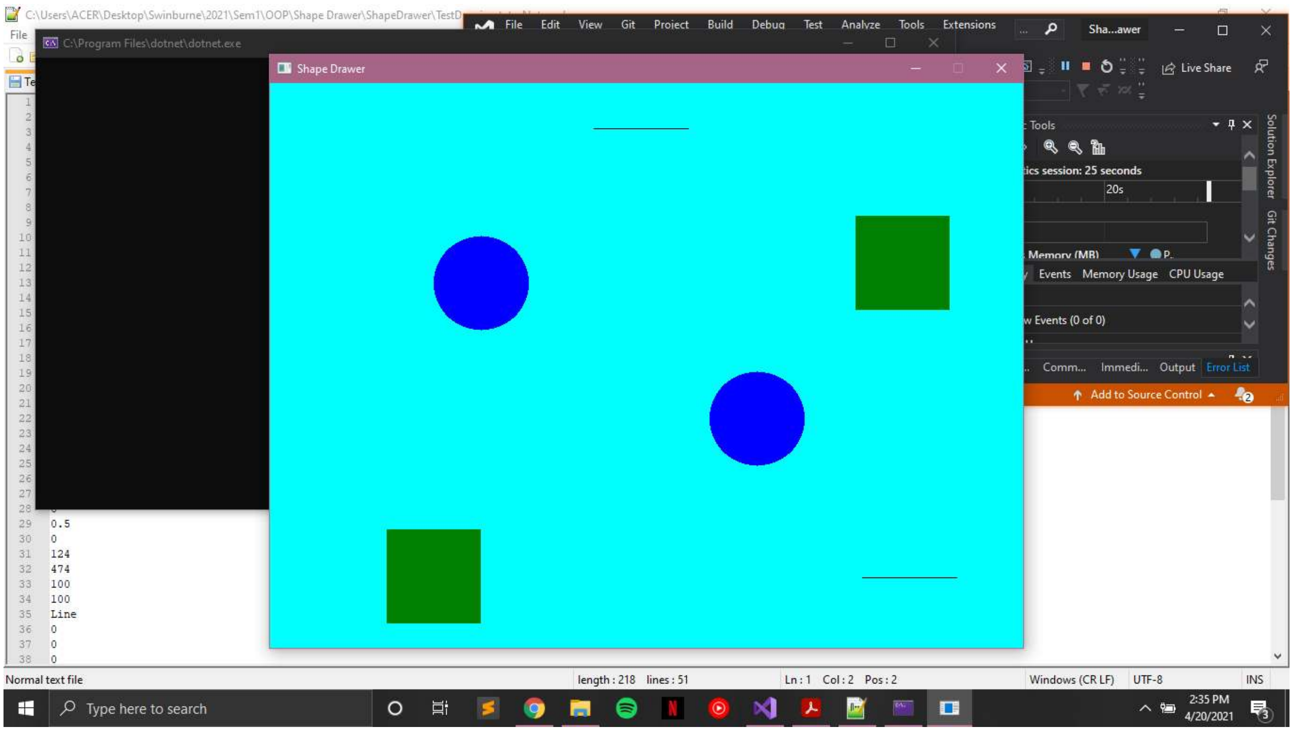
```
using System;
using System.Collections.Generic;
using System.Text;
using SplashKitSDK;
using System.IO;

namespace ShapeDrawer
{
    public class MyCircle : Shape
    {
        private int _radius;

        public int Radius
        {
            get { return _radius; }
            set { _radius = value; }
        }
        public MyCircle(Color color, int radius) : base()
        {
            this._radius = radius;
            this.color = color;
        }
        public MyCircle() : this(Color.Blue, 50)
        {

        }
        public override void Draw()
        {
            SplashKit.FillCircle(color, X, Y, _radius);
            if (Selected)
            {
                DrawOutline();
            }
        }
        public override void DrawOutline()
        {
            SplashKit.DrawCircle(Color.Black, X, Y, Radius + 2);
        }
        public override bool IsAt(Point2D pt)
        {
            if (pt.X >= (X-Radius) && (pt.X <= (X + Radius)) && (pt.Y >= Y -
             ↪  Radius) && (pt.Y <= Y + 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();
        }
    }
}
```

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

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

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

        public float EndX
        {
            get { return _endX; }
            set { _endX = value; }
        }
        public float EndY
        {
            get { return _endY; }
            set { _endY = value; }
        }
        public MyLine() : this(Color.Black, SplashKit.MouseX() + 100,
         ↪  SplashKit.MouseY(), SplashKit.MouseX() + 100, SplashKit.MouseY())
        {

        }
        public override void Draw()
        {
            SplashKit.DrawLine(color, X, Y, _endX, _endY);

            if (Selected)
            {
                DrawOutline();
            }
        }
        public override void DrawOutline()
        {
            SplashKit.DrawCircle(Color.Black, X, Y, 10);
            SplashKit.DrawCircle(Color.Black, _endX, _endY, 10);
        }
        public override bool IsAt(Point2D pt)
        {
            if (((pt.X >= X) && (pt.X <= _endX)) && (pt.Y >= _endY - 5) && (pt.Y <= _endY + 5))
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        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);
            //X = reader.ReadInteger();
            //Y = reader.ReadInteger();
            _endX = reader.ReadInteger();
            _endY = reader.ReadInteger();
        }
    }
}
```

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

namespace ShapeDrawer
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }
        public static void Main()
        {
            new Window("Shape Drawer", 800, 600);

            var myDrawing = new Drawing();

            ShapeKind kindToAdd = ShapeKind.Circle;

            do
            {
                SplashKit.ProcessEvents();
                //SplashKit.ClearScreen(screencolor);

                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.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();

                myDrawing.AddShape(newShape);
            }
            if (SplashKit.MouseClicked(MouseButton.RightButton))
            {
                Point2D pt;
                pt.X = SplashKit.MouseX();
                pt.Y = SplashKit.MouseY();
                myDrawing.SelectShapesAt(pt);
            }
            if (SplashKit.KeyTyped(KeyCode.SpaceKey))
            {
                myDrawing.Background = SplashKit.RandomRGBColor(255);
            }
            if ((SplashKit.KeyTyped(KeyCode.DeleteKey)) || (SplashKit.KeyTyped(KeyCode.BackspaceKey)))
            {
                foreach (var s in myDrawing.SelectedShapes)
                {
                    myDrawing.RemoveShape(s);
                }
            }
            if (SplashKit.KeyTyped(KeyCode.SKey))
            {
                myDrawing.Save("C:\\Users\\ACER\\Desktop\\Swinburne\\2021\\Sem1\\OOP\\Shape Drawer\\ShapeDrawer\\TestDrawing.txt");
            }
            if (SplashKit.KeyTyped(KeyCode.OKey))
            {
                try
                {
                    myDrawing.Load("C:\\Users\\ACER\\Desktop\\Swinburne\\2021\\Sem1\\OOP\\Shape Drawer\\ShapeDrawer\\TestDrawing.txt");
                }
                catch (Exception e)
                {
                    Console.Error.WriteLine("Error loading file: {0}", e.Message);
                }
            }

            SplashKit.ClearScreen(myDrawing.Background);
            myDrawing.Draw();
            SplashKit.RefreshScreen();

        } while (!SplashKit.WindowCloseRequested("Shape Drawer"));
```

```
            }
        }
}
```


C:\Users\ACER\Desktop\Swinburne\2021\Sem1\OOP\Shape Drawer\ShapeDrawer\TestD
C:\Program Files\dotnet\dotnet.exe
Shape Drawer
File Edit View Git Project Build Debug Test Analyze Tools Extensions
Sha...awer
Live Share
Solution Explorer
Git Changes
Tools
ics session: 25 seconds
20s
Memory (MB)
Events Memory Usage CPU Usage
w Events (0 of 0)
Comm... Immedi... Output Error List
Add to Source Control
Type here to search
2:35 PM
4/20/2021

```
0.5
0
124
474
100
100
Line
0
0
0
```

Normal text file length : 218 lines : 51 Ln : 1 Col : 2 Pos : 2 Windows (CR LF) UTF-8 INS