Swinburne University of Technology

COS20007 Object Oriented Programming

# 4.1P - Drawing Program - Multiple Shape Kinds

PDF generated at 17:57 on Monday 3rd April, 2023

```
using System;
using System.ComponentModel.Design;
using System.Globalization;
using System.Reflection.Metadata.Ecma335;
using SplashKitSDK;

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


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

            ShapeKind kindToAdd = ShapeKind.Circle;

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

                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)
                    {
                        MyCircle newCircle = new MyCircle();
                        newShape = newCircle;
                    }
                    else if (kindToAdd == ShapeKind.Rectangle)
                    {
                        MyRectangle newRect = new MyRectangle();
                        newShape = newRect;
                    }
                    else
                    {
                        MyLine newLine = new MyLine();
                        newShape = newLine;
                    }
                    newShape.X = SplashKit.MouseX();
                    newShape.Y = SplashKit.MouseY();
                    NewDrawing.AddShape(newShape);
                }


                if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                {
                    NewDrawing.Background = Color.RandomRGB(255);
                }

                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    NewDrawing.SelectShapeAt(SplashKit.MousePosition());
                }

                if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
 ↪  SplashKit.KeyTyped(KeyCode.DeleteKey))
                {
                    foreach (Shape s in NewDrawing.SelectedShape())
                    {
                        NewDrawing.RemoveShape(s);
                    }
                }

                NewDrawing.Draw();
                SplashKit.RefreshScreen();

            }
            while (!window.CloseRequested);

        }

    }


}
```

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



namespace ShapeDrawer
{
    internal 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 int ShapeCount
        {
            get
            {
                return _shapes.Count;
            }
        }

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


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

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

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

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

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

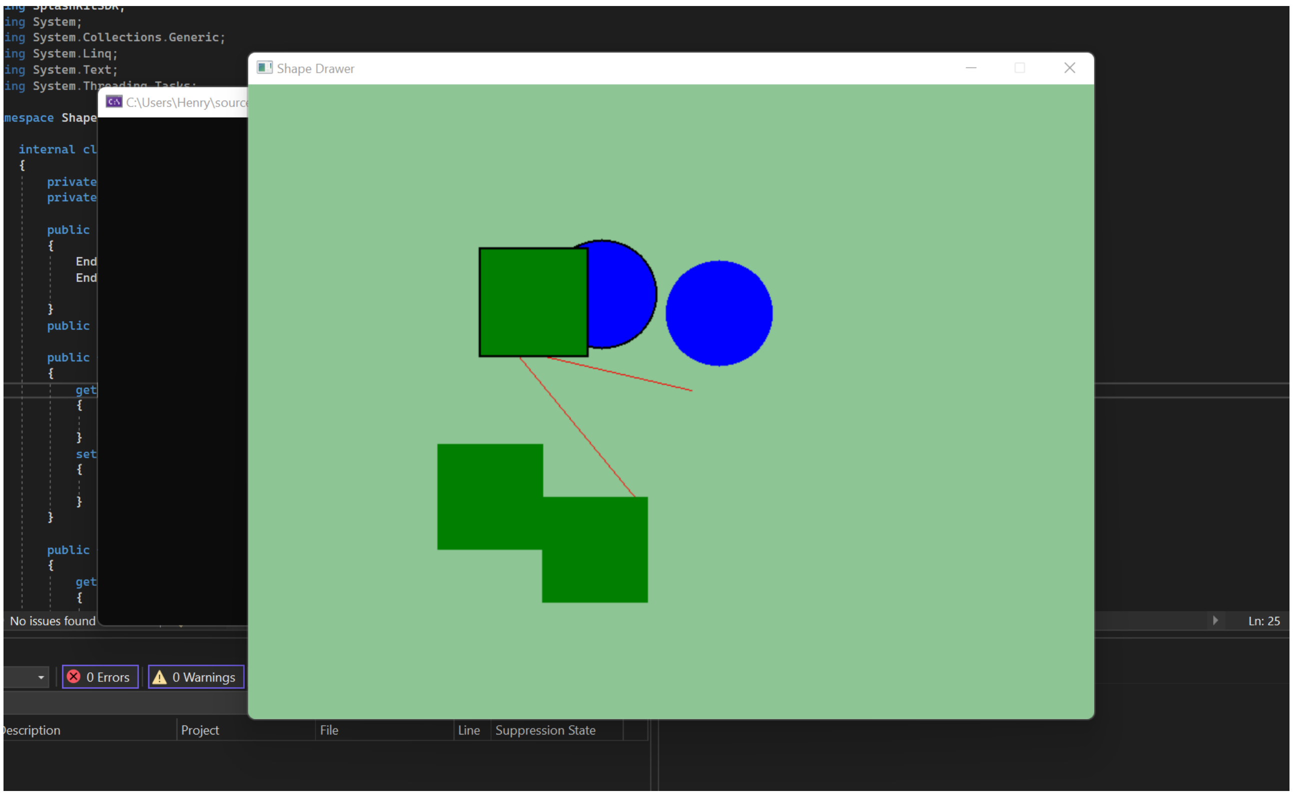
```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;

namespace ShapeDrawer
{
    public abstract class Shape
    {
        private Color _color;
        private float _x;
        private float _y;

        private bool _selected;

        public Shape(Color color)
        {
            _color = color;
            _x = 0;
            _y = 0;
        }

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

        public abstract void Draw();

        public abstract bool IsAt(Point2D pt);

        public float X
        {
            get
            {
                return _x;
            }
            set
            {
                _x = value;
            }

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

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

        public abstract void DrawOutline();


    }
}
```

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

namespace ShapeDrawer
{
    internal class MyRectangle : Shape
    {
        int _width;
        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)
        {
            X = x;
            Y = y;
            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)
            {
            DrawOutline();
            }
            SplashKit.FillRectangle (Color, X, Y, Width, Height);
        }
        public override void DrawOutline()
        {
            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 +_width)
            {
            return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

```csharp
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
{
    internal class MyCircle : Shape
    {
        private int _radius;

        public MyCircle(Color color, float x, float y, int radius): base(color)
        {
            _radius = 50;
        }

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


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

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

        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color.Black, X, Y, _radius + 2);
        }

        public override bool IsAt(Point2D pt)
        {
            if (SplashKit.PointInCircle(pt, SplashKit.CircleAt(X, Y, Radius)))
            {
                return true;
            }
            else
```

```
                {
                    return false;
                }

            }
        }
}
```

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

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

        public MyLine(Color color, float startX, float startY, float endX, float
    ↪   endY): base(color)
        {
            EndX = X;
            EndY = Y;

        }
        public MyLine() : this (Color.Red, 250, 250, 0, 0) { }

        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(Color, 250, 250, X, Y);
        }

        public override void DrawOutline()
        {
            SplashKit.FillCircle(Color.Black, X, Y, 5);
            SplashKit.FillCircle(Color.Black, 250, 250, 5);
        }

        public override bool IsAt(Point2D pt)
        {
            if (SplashKit.PointOnLine(pt, SplashKit.LineFrom(250, 250, X, Y)))
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```



```
ing System;
ing System.Collections.Generic;
ing System.Linq;
ing System.Text;
ing System.Threading.Tasks;

mespace Shape

  internal cl
  {
      private
      private

      public
      {
          End
          End

      }
      public

      public
      {
          get
          {

          }
          set
          {

          }
      }

      public
      {
          get
          {
```

No issues found

0 Errors 0 Warnings