Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 4.1P: Drawing Program: Multiple Shape Kinds

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/04/25 22:37

Tutor:
Jai Cornes

April 26, 2022

```
using System;
using SplashKitSDK;



namespace Task4._1
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Line,
            Circle
        }
        public static void Main()
        {

            new Window("Shape Drawer", 800, 600);
            //_ = new ShapeDrawer();
            Drawing myDrawing = new Drawing();

            ShapeKind kindToAdd = ShapeKind.Rectangle;

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

                if (SplashKit.KeyTyped(KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line;
                }


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


                if (SplashKit.KeyTyped(KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle;
                }

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape myShape = new MyLine();


                    if (kindToAdd == ShapeKind.Rectangle)
                    {
```

```
54                      MyRectangle newRectangle = new MyRectangle();
55                      myShape = newRectangle;
56                  }
57                  else if (kindToAdd == ShapeKind.Circle)
58                  {
59                      MyCircle newCircle = new MyCircle();
60                      myShape = newCircle;
61                  }
62                  else if (kindToAdd == ShapeKind.Line)
63                  {
64                      MyLine newLine = new MyLine();
65                      myShape = newLine;
66                  }
67
68
69                  myShape.X = SplashKit.MouseX();
70                  myShape.Y = SplashKit.MouseY();
71
72
73                  myDrawing.AddShapes(myShape);
74              }
75
76
77
78
79              if (SplashKit.MouseClicked(MouseButton.RightButton))
80              {
81                  myDrawing.SelectShapesAt(SplashKit.MousePosition());
82              }
83
84
85
86
87
88              if (SplashKit.KeyTyped(KeyCode.SpaceKey))
89              {
90                  myDrawing.Background = SplashKit.RandomRGBColor(255);
91              }
92
93
94              if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
                ↪  (SplashKit.KeyTyped(KeyCode.DeleteKey)))
95              {
96                  foreach (Shape Shape in myDrawing.SelectedShapes)
97                  {
98                      myDrawing.RemoveShapes(Shape);
99                  }
100             }
101
102             myDrawing.Draw();
103
104             //SplashKit.ClearScreen();
105             SplashKit.RefreshScreen();
```

```
106             } while (!SplashKit.WindowCloseRequested("Shape Drawer"));
107         }
108     }
109 }
```

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



//using System.Collections.Generic.List;

namespace Task4._1
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;

        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> NewShapeSelected = new List<Shape>();

                foreach (Shape shape in _shapes)
                {
                    if (shape.Selected)
                    {
                        NewShapeSelected.Add(shape);
                    }
                }
                return NewShapeSelected;
            }

        }


        public Drawing() : this(Color.Red)
        { }
        //foreach (Shape shape in _shapes)
        //{
        //  shape.Draw();
        //}
        //}



        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;

        }
```

```
54
55
56          public int ShapeCount
57          {
58              get
59              {
60                  return _shapes.Count;
61              }
62          }
63
64
65          public void Draw()
66          {
67              SplashKit.ProcessEvents();
68              //SplashKit.ClearScreen(_background);
69              SplashKit.FillRectangle(_background, 0, 0, SplashKit.ScreenWidth(),
                ↪  SplashKit.ScreenHeight());
70
71              foreach (Shape shape in _shapes) shape.Draw();
72
73          }
74
75
76          public void AddShapes(Shape shape)
77          {
78              _shapes.Add(shape);
79          }
80
81
82          public Color Background
83          {
84              get
85              {
86                  return _background;
87              }
88              set
89              {
90                  _background = value;
91              }
92          }
93
94
95          public void SelectShapesAt(Point2D pt)
96          {
97              foreach (Shape Shapes in _shapes)
98              {
99                  if (Shapes.IsAt(pt))
100                 {
101                     Shapes.Selected = true;
102                 }
103                 else
104                 {
105                     Shapes.Selected = false;
```

```
106                 }
107             }
108         }
109
110         public void RemoveShapes(Shape shape)
111         {
112             _shapes.Remove(shape);
113         }
114
115
116     }
117
118
119
120
121
122
123
124 }
```

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

namespace Task4._1
{
    public abstract class Shape
    {
        private Color _color;
        private float _x, _y;
        //private int _width, _height;
        private bool _selected;

        public Shape() //contructor can only return the reference of the obj
        {
            Color = Color.Yellow;
            //_x = 0;
            //_y = 0;
            //_width = 230;
            //_height = 150;
        }

        public abstract void Draw();


        public abstract void DrawOutline();


        public float X
        {
            get { return _x; }

            set { _x = value; }
        }


        public Color Color
        {
            get { return _color; }
            set { _color = value; }

        }


        public float Y
        {
            get { return _y; }

            set { _y = value; }
        }
```

```
54
55
56          public bool Selected
57          {
58              get { return _selected; }
59              set { _selected = value; }
60          }
61
62          public abstract bool IsAt(Point2D point);
63
64
65
66
67      }
68      }
```

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

namespace Task4._1
{
    public class MyRectangle : Shape
    {
        private int height, width;

        public MyRectangle(Color clr, float x, float y, int width, int height)
        {
            Color = clr;
            X = x;
            Y = y;
            Height = height;
            Width = width;

        }

        public MyRectangle(): this(Color.Green,0,0,100,100)
        {
            //Color = Color.Green;
            //Height = 100;
            //Width = 100;
            //X = 0;
            //Y = 0;
        }

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


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

        public override bool IsAt (Point2D point)
        {
            if (point.X >= X && point.X <= X + width && point.Y >= Y && +point.Y <=
            ↪  Y + height)
            {
                return true;
            }
            return false;
```

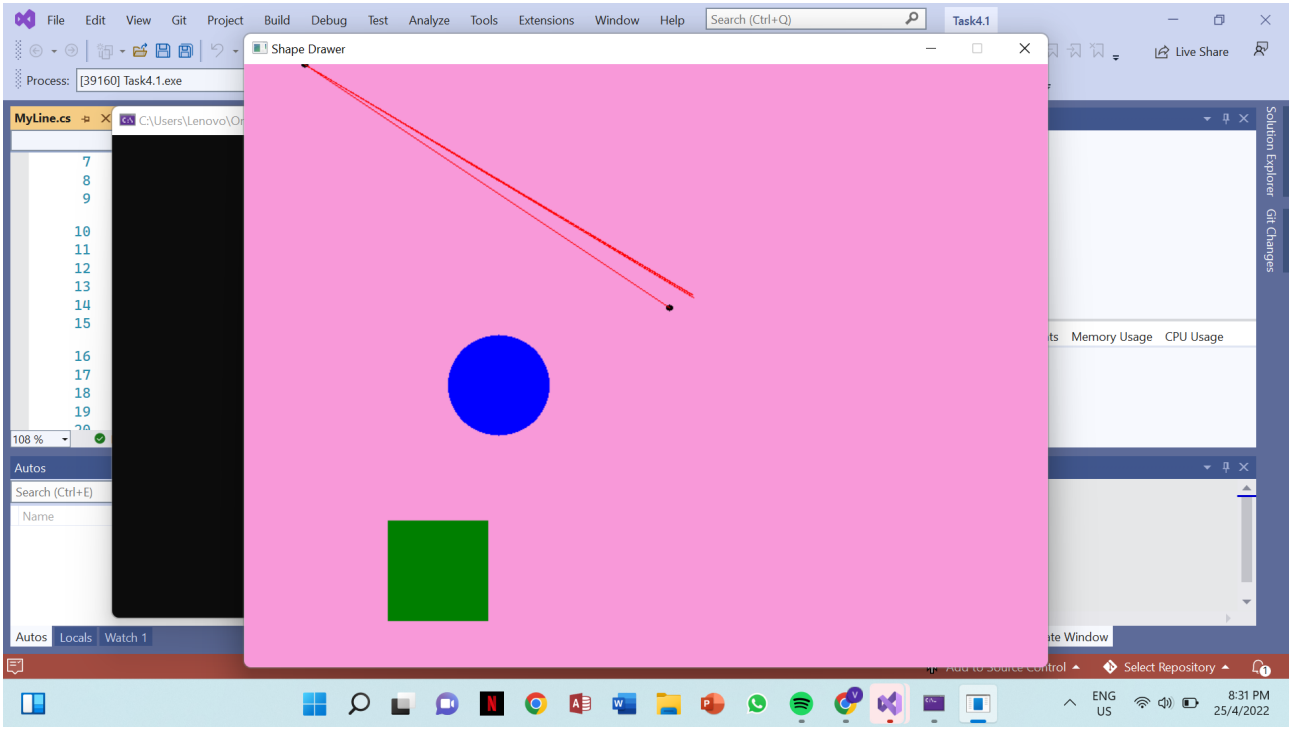
```
53          }
54
55          public override void Draw()
56          {
57              if (Selected)
58              {
59                  DrawOutline();
60              }
61
62              SplashKit.FillRectangle(Color, X, Y, width, height);
63          }
64
65          public override void DrawOutline()
66          {
67              SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, width + 4, height +
                ↪ 4);
68          }
69      }
70  }
```

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

namespace Task4._1
{
    public class MyCircle : Shape
    {
        private int radius;

        public int Radius
        {
            get
            {
                return radius;
            }

            set
            {
                radius = value;
            }
        }

        public MyCircle(Color clr, int radius)
        {
            Radius = radius;
            Color = clr;
        }


        public MyCircle(): this(Color.Blue,50)
        {
            //Color = Color.Blue;
            //Radius = 50;
        }

        public override bool IsAt(Point2D point)
        {
            if (point.X >= X && point.X <= X + Radius && point.Y >= Y && +point.Y
            ↪  <= Y + Radius)
            {
                return true;
            }
            return false;
        }


        public override void Draw()
        {
            if (Selected)
```

```
53              {
54                  DrawOutline();
55              }
56
57              SplashKit.FillCircle(Color, X, Y, radius);
58          }
59
60
61          public override void DrawOutline()
62          {
63              SplashKit.FillCircle(Color.Black, X, Y, + Radius + 2);
64          }
65      }
66  }
```

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

namespace Task4._1
{
    public class MyLine : Shape
    {

        private float _endX, _endY;


        public MyLine() : this(Color.Red,0,0, 60,0)
        {
            //Color = Color.Red;
        }


        public MyLine(Color clr, int x, int y, float xEnd, float yEnd)
        {
            Color = clr;
            X = x;
            Y = y;
            EndX = xEnd;
            EndY = yEnd;
        }


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

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

```
54          }
55
56
57          public override bool IsAt(Point2D point)
58          {
59              if (point.X >= X && point.X <= X + EndX && point.Y >= Y && +point.Y <=
                ↪  Y + EndY)
60              {
61                  return true;
62              }
63              return false;
64          }
65
66
67          public override void Draw()
68          {
69              if (Selected)
70              {
71                  DrawOutline();
72              }
73
74              SplashKit.DrawLine(Color, X, Y, EndX, EndY);
75          }
76
77
78          public override void DrawOutline()
79          {
80              SplashKit.FillCircle(Color.Black, X, Y, 3); //hard code the values for
                ↪  the end point
81              SplashKit.FillCircle(Color.Black, EndX, EndY, 3); //hard code the
                ↪  values for the end point
82          }
83      }
84  }
85
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Task4.1
Live Share
Process: [39160] Task4.1.exe
Shape Drawer
MyLine.cs
Memory Usage
CPU Usage
Autos
Search (Ctrl+E)
Name
Autos
Locals
Watch 1
Select Repository
ENG
US
8:31 PM
25/4/2022