SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Drawing Program - Multiple Shape Kinds

PDF generated at 22:30 on Tuesday $19^{\rm th}$ September, 2023

File 1 of 7 Program class

```
using System;
   using SplashKitSDK;
2
   namespace ShapeDrawer
   {
5
        public class Program
6
            private enum ShapeKind
                Rectangle,
                Circle,
11
                Line
12
            }
13
            public static void Main()
            {
15
                 Drawing myDrawing = new Drawing();
                 ShapeKind kindToAdd = ShapeKind.Circle;
17
18
                new Window("Drawing Shape", 800, 600);
19
                 do
20
                 {
                     SplashKit.ProcessEvents();
22
                     SplashKit.ClearScreen();
23
                     if (SplashKit.KeyTyped(KeyCode.RKey))
24
                     {
25
                         kindToAdd = ShapeKind.Rectangle;
26
                     }
27
                     if (SplashKit.KeyTyped(KeyCode.LKey))
28
                     {
29
                         kindToAdd = ShapeKind.Line;
30
                     }
31
                        (SplashKit.KeyTyped(KeyCode.CKey))
32
                     {
                         kindToAdd = ShapeKind.Circle;
34
                     }
35
                     if (SplashKit.MouseClicked(MouseButton.LeftButton))
36
                     {
37
                         Shape newShape;
38
                         if (kindToAdd == ShapeKind.Circle)
39
                         {
40
                              MyCircle newCircle = new MyCircle();
41
42
                              newShape = newCircle;
43
44
                         }
45
                         else if (kindToAdd == ShapeKind.Rectangle)
46
47
                              MyRectangle newRect = new MyRectangle();
48
49
                              newShape = newRect;
50
                         }
51
                         else
52
                          {
53
```

File 1 of 7 Program class

```
MyLine newLine = new MyLine();
54
55
                             newShape = newLine;
56
                         }
                         newShape.X = SplashKit.MouseX();
58
                         newShape.Y = SplashKit.MouseY();
59
                         myDrawing.AddShape(newShape);
60
                     }
61
62
                     if (SplashKit.MouseClicked(MouseButton.RightButton))
63
                     {
64
                         myDrawing.SelectedShapeAt(SplashKit.MousePosition());
65
                     }
66
67
                     if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
68
       SplashKit.KeyTyped(KeyCode.DeleteKey))
69
                         List<Shape> selectedShapes = myDrawing.SelectedShapes();
70
                         foreach (Shape selectedShape in selectedShapes)
71
                         {
72
                                  myDrawing.RemoveShape(selectedShape);
                         }
74
                     }
75
76
                        (SplashKit.KeyTyped(KeyCode.SpaceKey))
78
                         myDrawing.Background = SplashKit.RandomRGBColor(255);
79
                     }
81
                     myDrawing.Draw();
82
83
                     SplashKit.RefreshScreen();
84
                } while (!SplashKit.WindowCloseRequested("Drawing Shape"));
86
            }
87
        }
88
   }
89
```

File 2 of 7 Drawing class

```
using System;
   using System.Linq;
   using System.Collections.Generic;
   using SplashKitSDK;
   namespace ShapeDrawer
6
        public class Drawing
            private readonly List<Shape> _shapes;
            private Color _background;
12
            public Drawing(Color background)
13
                 _shapes = new List<Shape>();
15
                 _background = background;
            }
17
18
            public Drawing() : this(Color.White)
19
            {
20
            }
22
            public List<Shape> SelectedShapes()
23
24
                 List<Shape> selectedShapes = new List<Shape>();
25
                 foreach (Shape s in _shapes)
26
27
                     if (s.Selected)
29
                          selectedShapes.Add(s);
30
                     }
31
                 }
32
                 return selectedShapes;
34
            public int ShapeCount
35
36
                 get
37
                     return _shapes.Count;
39
                 }
40
            }
41
42
            public Color Background
43
            {
                 get
                 {
46
                     return _background;
47
48
                 set
49
                 {
50
                     _background = value;
51
                 }
52
            }
53
```

File 2 of 7 Drawing class

```
54
             public void Draw()
55
56
                 SplashKit.ClearScreen(_background);
58
                 foreach (Shape s in _shapes)
60
                      s.Draw();
61
             }
             public void SelectedShapeAt(Point2D pt)
65
66
                 foreach (Shape s in _shapes)
67
                 {
68
                      if (s.IsAt(pt))
                      {
70
                          s.Selected = true;
72
                      else
73
                      {
                          s.Selected = false;
                      }
76
                 }
             }
78
79
             public void AddShape(Shape s)
                 _shapes.Add(s);
82
             }
83
84
             public void RemoveShape(Shape s)
85
                 _shapes.Remove(s);
87
        }
89
   }
90
```

File 3 of 7 Shape class

```
using System;
    using SplashKitSDK;
2
   namespace ShapeDrawer
    {
5
        public abstract class Shape
6
             private Color _color;
             private float _x, _y;
             private bool _selected;
10
             private int _witdh, _height;
11
             public Shape(Color clr)
12
13
                  _color = clr;
14
             }
15
16
             public Color Color
17
             {
18
                  get
19
                  {
20
                      return _color;
                  }
22
                  set
23
                  {
24
                      _color = value;
25
                  }
26
             }
27
28
             public float X
29
             {
30
                  get
31
                  {
32
                      return _x;
                  }
34
                  set
35
                  {
36
                      _x = value;
37
                  }
38
             }
39
40
             public float Y
41
42
                  get
43
                  {
44
45
                      return _y;
                  }
46
                  set
47
                  {
48
                      _y = value;
49
                  }
50
             }
51
             public abstract void Draw();
52
             public abstract bool IsAt(Point2D p);
53
```

File 3 of 7 Shape class

```
54
55
             public bool Selected
56
                 get
58
                  {
59
                      return _selected;
60
                 }
61
                  set
62
                  {
63
                      _selected = value;
64
                 }
65
             }
66
67
             public abstract void DrawOutline();
68
70
        }
71
    }
72
```

File 4 of 7 MyRectangle class

```
using System;
   using System.Collections.Generic;
   using System.Linq;
   using System.Text;
   using System. Threading. Tasks;
   using SplashKitSDK;
   namespace ShapeDrawer
        public class MyRectangle : Shape
10
11
            private int _width, _height;
12
13
            public MyRectangle(Color clr, float x, float y, int width, int height) :
        base(clr)
            {
15
                X = x;
16
                Y = y;
17
                Width = width;
18
                Height = height;
19
            }
21
            public MyRectangle() : this(Color.Green, 0, 0, 100, 100) { }
22
23
            public int Width // Corrected typo
24
25
                get { return _width; }
26
                set { _width = value; }
28
            public int Height
29
30
                get { return _height; }
31
                set { _height = value; }
33
            public override void Draw()
34
35
                if (Selected)
36
                     DrawOutline();
38
39
                SplashKit.FillRectangle(Color, X, Y, Width, Height);
40
            }
41
            public override void DrawOutline()
42
            {
43
                SplashKit.FillRectangle(Color, X - 2, Y - 2, Width + 4, Height + 4);
45
            public override bool IsAt(Point2D p)
46
47
                if ((p.X > X) \&\& (p.X < (X + _width)))
48
49
                     if ((p.Y > Y) \&\& (p.Y < (Y + _height)))
50
                     {
51
                         return true;
52
```

File 4 of 7 MyRectangle class

File 5 of 7 MyCircle class

```
using System;
   using System.Collections.Generic;
   using System.Linq;
   using System.Text;
   using System. Threading. Tasks;
   using SplashKitSDK;
   namespace ShapeDrawer
        public class MyCircle: Shape
        {
10
            private int _radius;
11
12
            public MyCircle(Color clr,float x, float y, int radius) : base(clr)
13
                X = X;
15
                Y = y;
                _radius = radius;
17
            }
18
            public MyCircle() : this(Color.Blue,0,0, 50) { }
19
            public int Radius { get { return _radius; } }
20
            public override void Draw()
22
                if (Selected)
23
                     DrawOutline();
24
                SplashKit.FillCircle(Color, X, Y, _radius);
25
            }
26
            public override void DrawOutline()
27
                SplashKit.FillCircle(Color, X - 2, Y - 2, _radius + 2);
29
30
            public override bool IsAt(Point2D p)
31
32
                double a = (double)(p.X - X);
                double b = (double)(p.Y - Y);
34
                if (Math.Sqrt(a * a + b * b) < _radius)</pre>
35
                {
36
                     return true;
37
38
                return false;
39
            }
40
        }
41
   }
42
```

File 6 of 7 MyLine class

```
using System;
   using System.Collections.Generic;
   using System.Linq;
   using System. Text;
   using System. Threading. Tasks;
   using SplashKitSDK;
   namespace ShapeDrawer
       public class MyLine : Shape
10
11
            private float _endX;
12
            private float _endY;
13
            public MyLine(Color clr,float startX, float startY, float endX, float endY) :
       base(clr)
            {
                _endX = endX;
16
                _endY = endY;
17
                X = startX;
18
                Y = startY;
19
            }
21
            public MyLine() : this(Color.RandomRGB(255), 0, 0, 20, 20) { }
22
23
            public float EndX
25
                get { return _endX; }
26
                set { _endX = value; }
28
            public float EndY
29
30
                get { return _endY; }
31
                set { _endY = value; }
33
34
            public override void Draw()
35
36
                if (Selected)
                {
38
                    DrawOutline();
39
40
                SplashKit.DrawLine(Color, X, Y, _endX, _endY);
41
42
            public override void DrawOutline()
43
                SplashKit.DrawCircle(Color.Black, X, Y, 5);
45
                SplashKit.DrawCircle(Color.Black, _endX, _endY, 5);
46
47
            public override bool IsAt(Point2D p)
48
49
                // Calculate the distance from the point to the line
50
                double distance = Math.Abs((EndY - Y) * p.X - (EndX - X) * p.Y + EndX * Y
51
        - EndY * X)
```

File 6 of 7 MyLine class

```
/ Math.Sqrt(Math.Pow(EndY - Y, 2) + Math.Pow(EndX - X,
52
                                                                                                                       2));
53
                                                                                                                                                                                                                                                                                    // Define a tolerance value for how close the point can be to the line
                                                                                                                                                                                                                                                                                 double tolerance = 5.0; // Adjust as needed
    55
  56
                                                                                                                                                                                                                                                                                    // Check if the distance is within the tolerance % \left( 1\right) =\left( 1\right) \left( 1
57
                                                                                                                                                                                                                                                                                 return distance <= tolerance;</pre>
  58
                                                                                                                                                                                                           }
  59
    60
                                                                                                                                     }
    61
                                                            }
  62
```

