

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

---

## 4.1P - Drawing Program - Multiple Shape Kinds

---

PDF generated at 17:57 on Monday 3<sup>rd</sup> April, 2023

```
1  using System;
2  using System.ComponentModel.Design;
3  using System.Globalization;
4  using System.Reflection.Metadata.Ecma335;
5  using SplashKitSDK;
6
7  namespace ShapeDrawer
8  {
9      public class Program
10     {
11         private enum ShapeKind
12         {
13             Rectangle,
14             Circle,
15             Line
16         }
17
18
19         public static void Main()
20         {
21             Window window = new Window("Shape Drawer", 800, 600);
22             Drawing NewDrawing = new Drawing();
23
24             ShapeKind kindToAdd = ShapeKind.Circle;
25
26             do
27             {
28                 SplashKit.ProcessEvents();
29                 SplashKit.ClearScreen();
30
31                 if (SplashKit.KeyTyped(KeyCode.RKey))
32                 {
33                     kindToAdd = ShapeKind.Rectangle;
34                 }
35                 if (SplashKit.KeyTyped(KeyCode.CKey))
36                 {
37                     kindToAdd = ShapeKind.Circle;
38                 }
39                 if (SplashKit.KeyTyped(KeyCode.LKey))
40                 {
41                     kindToAdd = ShapeKind.Line;
42                 }
43                 if (SplashKit.MouseClicked(MouseButton.LeftButton))
44                 {
45                     Shape newShape;
46
47                     if (kindToAdd == ShapeKind.Circle)
48                     {
49                         MyCircle newCircle = new MyCircle();
50                         newShape = newCircle;
51                     }
52                     else if (kindToAdd == ShapeKind.Rectangle)
53                     {
```

```
54         MyRectangle newRect = new MyRectangle();
55         newShape = newRect;
56     }
57     else
58     {
59         MyLine newLine = new MyLine();
60         newShape = newLine;
61     }
62     newShape.X = SplashKit.MouseX();
63     newShape.Y = SplashKit.MouseY();
64     NewDrawing.AddShape(newShape);
65 }
66
67
68     if (SplashKit.KeyTyped(KeyCode.SpaceKey))
69     {
70         NewDrawing.Background = Color.RandomRGB(255);
71     }
72
73     if (SplashKit.MouseClicked(MouseButton.RightButton))
74     {
75         NewDrawing.SelectShapeAt(SplashKit.MousePosition());
76     }
77
78     if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
↪ SplashKit.KeyTyped(KeyCode.DeleteKey))
79     {
80         foreach (Shape s in NewDrawing.SelectedShape())
81         {
82             NewDrawing.RemoveShape(s);
83         }
84     }
85
86     NewDrawing.Draw();
87     SplashKit.RefreshScreen();
88
89 }
90 while (!window.CloseRequested);
91
92 }
93
94 }
95
96
97 }
```

```
1  using SplashKitSDK;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8
9
10 namespace ShapeDrawer
11 {
12     internal class Drawing
13     {
14         private readonly List<Shape> _shapes;
15         private Color _background;
16
17         public Drawing(Color background)
18
19         {
20             _shapes = new List<Shape>();
21             _background = background;
22
23         }
24         public Drawing() : this(Color.White) { }
25
26         public int ShapeCount
27         {
28             get
29             {
30                 return _shapes.Count;
31             }
32         }
33
34         public void AddShape(Shape s)
35         {
36             _shapes.Add(s);
37         }
38
39
40         public void Draw()
41         {
42             SplashKit.ClearScreen(_background);
43             foreach (Shape s in _shapes)
44             {
45                 s.Draw();
46             }
47         }
48
49         public Color Background
50         {
51             get
52             {
53                 return _background;
```

```
54         }
55         set
56         {
57             _background = value;
58         }
59     }
60
61     public void SelectShapeAt(Point2D pt)
62     {
63         foreach (Shape s in _shapes)
64         {
65             if (s.IsAt(pt))
66             {
67                 s.Selected = true;
68             }
69             else
70             {
71                 s.Selected = false;
72             }
73         }
74     }
75
76     public List<Shape> SelectedShape()
77     {
78         List<Shape> result = new List<Shape>();
79         foreach (Shape s in _shapes)
80         {
81             if (s.Selected)
82             {
83                 result.Add(s);
84             }
85         }
86         return result;
87     }
88
89     public void RemoveShape(Shape s)
90     {
91         _shapes.Remove(s);
92     }
93 }
94 }
```

```
1  using SplashKitSDK;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Numerics;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace ShapeDrawer
10 {
11     public abstract class Shape
12     {
13         private Color _color;
14         private float _x;
15         private float _y;
16
17         private bool _selected;
18
19         public Shape(Color color)
20         {
21             _color = color;
22             _x = 0;
23             _y = 0;
24         }
25
26         public Shape() : this(Color.Yellow) { }
27
28         public abstract void Draw();
29
30         public abstract bool IsAt(Point2D pt);
31
32         public float X
33         {
34             get
35             {
36                 return _x;
37             }
38             set
39             {
40                 _x = value;
41             }
42         }
43
44         public float Y
45         {
46             get
47             {
48                 return _y;
49             }
50             set
51             {
52                 _y = value;
53             }
54         }
55     }
56 }
```

```
54     }
55
56     public Color Color
57     {
58         get
59         {
60             return _color;
61         }
62         set
63         {
64             _color = value;
65         }
66     }
67     public bool Selected
68     {
69         get
70         {
71             return _selected;
72         }
73         set
74         {
75             _selected = value;
76         }
77     }
78
79     public abstract void DrawOutline();
80
81
82 }
83 }
```

```
1  using SplashKitSDK;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ShapeDrawer
9  {
10     internal class MyRectangle : Shape
11     {
12         int _width;
13         int _height;
14
15         public MyRectangle() : this(Color.Green, 0, 0, 100, 100) { }
16
17
18         public MyRectangle(Color clr, float x, float y, int width, int height): base
↪      (clr)
19         {
20             X = x;
21             Y = y;
22             Width = width;
23             Height = height;
24         }
25
26         public int Width
27         {
28             get
29             {
30                 return _width;
31             }
32             set
33             {
34                 _width = value;
35             }
36         }
37         public int Height
38         {
39             get
40             {
41                 return _height;
42             }
43             set
44             {
45                 _height = value;
46             }
47         }
48         public override void Draw()
49         {
50             if (Selected)
51             {
52                 DrawOutline();
```



```
53         }
54         SplashKit.FillRectangle (Color, X, Y, Width, Height);
55     }
56     public override void DrawOutline()
57     {
58         SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, Width + 4, Height +
↵ 4);
59     }
60     public override bool IsAt(Point2D pt)
61     {
62         if (pt.X > X && pt.X <= X + _width && pt.Y > Y && pt.Y <= Y + _width)
63         {
64             return true;
65         }
66         else
67         {
68             return false;
69         }
70     }
71 }
72 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Security.Cryptography.X509Certificates;
5  using System.Text;
6  using System.Threading.Tasks;
7  using SplashKitSDK;
8
9  namespace ShapeDrawer
10 {
11     internal class MyCircle : Shape
12     {
13         private int _radius;
14
15         public MyCircle(Color color, float x, float y, int radius): base(color)
16         {
17             _radius = 50;
18         }
19
20         public MyCircle() : this(Color.Blue, 0, 0, 50) { }
21
22
23         public int Radius
24         {
25             get
26             {
27                 return _radius;
28             }
29             set
30             {
31                 _radius = value;
32             }
33         }
34
35         public override void Draw()
36         {
37             if (Selected)
38                 DrawOutline();
39             SplashKit.FillCircle(Color, X, Y, _radius);
40         }
41
42         public override void DrawOutline()
43         {
44             SplashKit.FillCircle(Color.Black, X, Y, _radius + 2);
45         }
46
47         public override bool IsAt(Point2D pt)
48         {
49             if (SplashKit.PointInCircle(pt, SplashKit.CircleAt(X, Y, Radius)))
50             {
51                 return true;
52             }
53             else
```

```
54         {  
55             return false;  
56         }  
57  
58     }  
59 }  
60 }
```

```
1  using SplashKitSDK;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ShapeDrawer
9  {
10     internal class MyLine : Shape
11     {
12         private float _endX;
13         private float _endY;
14
15         public MyLine(Color color, float startX, float startY, float endX, float
↵ endY): base(color)
16         {
17             EndX = X;
18             EndY = Y;
19
20         }
21         public MyLine() : this (Color.Red, 250, 250, 0, 0) { }
22
23         public float EndX
24         {
25             get
26             {
27                 return _endX;
28             }
29             set
30             {
31                 _endX = value;
32             }
33         }
34
35         public float EndY
36         {
37             get
38             {
39                 return _endY;
40             }
41             set
42             {
43                 _endY = value;
44             }
45         }
46
47         public override void Draw()
48         {
49             if (Selected)
50             {
51                 DrawOutline();
52             }
53         }
54     }
55 }
```

```
53         SplashKit.DrawLine(Color, 250, 250, X, Y);
54     }
55
56     public override void DrawOutline()
57     {
58         SplashKit.FillCircle(Color.Black, X, Y, 5);
59         SplashKit.FillCircle(Color.Black, 250, 250, 5);
60     }
61
62     public override bool IsAt(Point2D pt)
63     {
64         if (SplashKit.PointOnLine(pt, SplashKit.LineFrom(250, 250, X, Y)))
65         {
66             return true;
67         }
68         else
69         {
70             return false;
71         }
72     }
73 }
74 }
```

