

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

---

## 3.3P - Drawing Program - A Drawing Class

---

PDF generated at 12:37 on Tuesday 11<sup>th</sup> April, 2023

```
1  using DrawingProgram.lib;
2
3  namespace DrawingProgram
4  {
5      public class Program
6      {
7          public static void Main()
8          {
9              Window window = new Window("Shape Drawer", 800, 600);
10
11              Drawing drawing = new Drawing();
12              do
13              {
14                  SplashKit.ProcessEvents();
15                  SplashKit.ClearScreen();
16
17                  // add new shape
18                  if (SplashKit.MouseClicked(MouseButton.LeftButton))
19                  {
20                      drawing.AddShape(new Shape(SplashKit.MouseX(),
↪      SplashKit.MouseY()));
21                  }
22
23                  // delete a shape
24                  if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
↪      SplashKit.KeyTyped(KeyCode.DeleteKey))
25                  {
26                      drawing.DeleteShape();
27                  }
28
29                  // select a shape
30                  if (SplashKit.MouseClicked(MouseButton.RightButton))
31                  {
32                      drawing.SelectShapesAt(SplashKit.MousePosition());
33                  }
34
35                  // change background color
36                  if (SplashKit.KeyTyped(KeyCode.SpaceKey))
37                  {
38                      drawing.Background = Color.Random();
39                  }
40
41                  drawing.Draw();
42                  SplashKit.RefreshScreen();
43
44              } while (!window.CloseRequested);
45          }
46      }
47  }
```

```
1  using DrawingProgram.lib;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace DrawingProgram
9  {
10     public class Drawing
11     {
12         //variables
13         private readonly List<Shape> _shapes;
14         private Color _background;
15
16         //properties
17
18         //number of shapes in list, readonly
19         public int ShapeCount
20         {
21             get
22             {
23                 return _shapes.Count;
24             }
25         }
26
27         // background color
28         public Color Background
29         {
30             get
31             {
32                 return _background;
33             }
34             set
35             {
36                 _background = value;
37             }
38         }
39
40         //list of shapes that are currently selected
41         public List<Shape> SelectedShapes
42         {
43             get
44             {
45                 List<Shape> result = new List<Shape>();
46
47                 foreach (Shape s in _shapes)
48                 {
49                     if (s.Selected == true)
50                     {
51                         result.Add(s);
52                     }
53                 }
54             }
55         }
56     }
57 }
```

```
54
55         return result;
56     }
57 }
58
59 //constructor that accepts color as a parameter for the background
60 public Drawing(Color background)
61 {
62     _shapes = new List<Shape>();
63     _background = background;
64 }
65
66 //default constructor
67 public Drawing() : this(Color.White)
68 {
69
70 }
71
72 //methods
73 public void AddShape(Shape shape)
74 {
75     _shapes.Add(shape);
76 }
77
78 public void Draw()
79 {
80     SplashKit.ClearScreen();
81     foreach (Shape shape in _shapes)
82     {
83         shape.Draw();
84     }
85 }
86
87 public void SelectShapesAt(Point2D pt)
88 {
89     // checks if mouse position is over a shape, if true then its selected
90     // property is set to true
91     foreach (Shape s in _shapes)
92     {
93         if (s.IsAt(pt))
94         {
95             s.Selected = true;
96         }
97         else
98         {
99             s.Selected = false;
100         }
101     }
102 }
103
104 public void DeleteShape()
105 {
106     foreach (Shape s in _shapes.ToList())
```

```
107         {  
108             if (s.Selected)  
109             {  
110                 _shapes.Remove(s);  
111             }  
112         }  
113     }  
114 }  
115 }
```

```
1  using System;
2  using DrawingProgram.lib;
3
4  namespace DrawingProgram
5  {
6      public class Shape
7      {
8          // local variables
9          private Color _color;
10         private float _x;
11         private float _y;
12         private int _width;
13         private int _height;
14         private bool _selected;
15
16         // constructor
17         public Shape(float x, float y)
18         {
19             _color = Color.Green;
20             _x = x;
21             _y = y;
22             _width = 100;
23             _height = 100;
24             _selected = false;
25         }
26
27         // properties
28         public Color Color
29         {
30             get
31             {
32                 return _color;
33             }
34             set
35             {
36                 _color = value;
37             }
38         }
39
40         public float X
41         {
42             get
43             {
44                 return _x;
45             }
46             set
47             {
48                 _x = value;
49             }
50         }
51
52         public float Y
53         {
```

```
54         get
55         {
56             return _y;
57         }
58         set
59         {
60             _y = value;
61         }
62     }
63
64     public int Width
65     {
66         get
67         {
68             return _width;
69         }
70         set
71         {
72             _width = value;
73         }
74     }
75
76     public int Height
77     {
78         get
79         {
80             return _height;
81         }
82         set
83         {
84             _height = value;
85         }
86     }
87
88     public bool Selected
89     {
90         get
91         {
92             return _selected;
93         }
94         set
95         {
96             _selected = value;
97         }
98     }
99
100     // methods
101     public void Draw()
102     {
103         if (_selected)
104         {
105             DrawOutline();
106         }
```

```
107         SplashKit.FillRectangle(_color, _x, _y, _width, _height);
108     }
109
110     public bool IsAt(Point2D pt)
111     {
112         if (pt.X > _x && pt.Y > _y)
113         {
114             if (pt.X < _x + _width && pt.Y < _y + _height)
115             {
116                 return true;
117             }
118             else
119             {
120                 return false;
121             }
122         }
123         else
124         {
125             return false;
126         }
127     }
128
129     public void DrawOutline()
130     {
131         SplashKit.FillRectangle(Color.Black, _x - 2, _y - 2, _width + 4, _height
↵ + 4);
132     }
133 }
134 }
```



