

# Selected files

## 8 printable files

Week\_5/5.3/DrawingProgram/Drawing.cs  
Week\_5/5.3/DrawingProgram/ExtensionMethods.cs  
Week\_5/5.3/DrawingProgram/MyCircle.cs  
Week\_5/5.3/DrawingProgram/MyLine.cs  
Week\_5/5.3/DrawingProgram/Program.cs  
Week\_5/5.3/DrawingProgram/MyRectangle.cs  
Week\_5/5.3/DrawingProgram/Shape.cs  
Week\_5/5.3/DrawingProgram/TestDrawing.txt

### Week\_5/5.3/DrawingProgram/Drawing.cs

```
1  using System;
2  using System.IO;
3  using System.Collections.Generic;
4  using SplashKitSDK;
5  using System.Linq;
6  using System.Threading.Tasks;
7
8  namespace DrawingProgram
9  {
10     public class Drawing
11     {
12         private readonly List<Shape> _shapes;
13         private Color _background;
14
15         // Constructor
16         public Drawing(Color background)
17         {
18             _shapes = new List<Shape>();
19             _background = background;
20         }
21
22         public Drawing() : this(Color.White)
23         {
24         }
25
26         // Properties
27         public Color Background
28         {
29             get { return _background; }
30             set { _background = value; }
31         }
32
33         public int ShapeCount
34         {
35             get { return _shapes.Count; }
36         }
37     }
```

```
38 public List<Shape> SelectedShapes
39 {
40     get
41     {
42         List<Shape> result = new List<Shape>();
43         foreach (Shape s in _shapes)
44         {
45             if (s.Selected)
46             {
47                 result.Add(s);
48             }
49         }
50         return result;
51     }
52 }
53
54 // Methods
55 public void AddShape(Shape s)
56 {
57     _shapes.Add(s);
58 }
59
60 public void RemoveShape(Shape s)
61 {
62     _ = _shapes.Remove(s);
63 }
64
65 public void Draw()
66 {
67     SplashKit.ClearScreen(_background);
68     foreach (Shape s in _shapes)
69     {
70         s.Draw();
71     }
72 }
73
74 public void SelectShapesAt(Point2D pt)
75 {
76     foreach (Shape s in _shapes)
77     {
78         s.Selected = s.IsAt(pt);
79     }
80 }
81
82 public void RemoveSelectedShapes()
83 {
84     foreach (Shape s in SelectedShapes)
85     {
86         RemoveShape(s);
87     }
88 }
```

```

89
90     public void Save(string filename)
91     {
92         StreamWriter writer = new StreamWriter(filename);
93
94         try
95         {
96             writer.WriteColor(Background);
97             writer.WriteLine(ShapeCount);
98
99             foreach (Shape s in _shapes)
100             {
101                 s.SaveTo(writer);
102             }
103         }
104         finally
105         {
106             writer.Close();
107         }
108     }
109
110     public void Load(string filename)
111     {
112         StreamReader reader = new StreamReader(filename);
113         try
114         {
115
116             Background = reader.ReadColor();
117             int count = reader.ReadInteger();
118             _shapes.Clear();
119
120             for (int i = 0; i < count; i++)
121             {
122                 string kind = reader.ReadLine();
123                 Shape s;
124
125                 switch (kind)
126                 {
127                     case "Rectangle":
128                         s = new MyRectangle();
129                         break;
130                     case "Circle":
131                         s = new MyCircle();
132                         break;
133                     case "Line":
134                         s = new MyLine();
135                         break;
136                     default:
137                         throw new InvalidDataException("Unknown shape kind: " +
kind);
138                 }

```

```

139
140         s.LoadFrom(reader);
141         AddShape(s);
142     }
143 }
144 finally
145 {
146     reader.Close();
147 }
148 }
149 }
150 }

```

#### Week\_5/5.3/DrawingProgram/ExtensionMethods.cs

```

1 using System;
2 using System.IO;
3 using SplashKitSDK;
4
5 namespace DrawingProgram
6 {
7     public static class ExtensionMethods
8     {
9         public static int ReadInteger(this StreamReader reader)
10        {
11            return Convert.ToInt32(reader.ReadLine());
12        }
13        public static float ReadSingle(this StreamReader reader)
14        {
15            return Convert.ToSingle(reader.ReadLine());
16        }
17        public static Color ReadColor(this StreamReader reader)
18        {
19            return Color.RGBColor(reader.ReadSingle(), reader.ReadSingle(),
20            reader.ReadSingle());
21        }
22        public static void WriteColor(this StreamWriter writer, Color clr)
23        {
24            writer.WriteLine("{0}\n{1}\n{2}", clr.R, clr.G, clr.B);
25        }
26    }
27 }

```

#### Week\_5/5.3/DrawingProgram/MyCircle.cs

```

1 using System;
2 using System.IO;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Threading.Tasks;
6
7 using SplashKitSDK;
8

```

```
9 namespace DrawingProgram
10 {
11     public class MyCircle : Shape
12     {
13         private float _radius;
14
15         public MyCircle() : this(Color.Blue, 0.0f, 0.0f, 129)
16         {
17         }
18
19         public MyCircle(Color color, float x, float y, float radius) : base(color)
20         {
21             X = x;
22             Y = y;
23             Radius = radius;
24         }
25
26         public float Radius
27         {
28             get => _radius;
29             set => _radius = value;
30         }
31
32         public override void Draw()
33         {
34             SplashKit.FillCircle(Color, X, Y, _radius);
35
36             if (Selected)
37             {
38                 DrawOutline();
39             }
40         }
41
42         public override bool IsAt(Point2D pt)
43         {
44             Circle c = SplashKit.CircleAt(X, Y, _radius);
45             return SplashKit.PointInCircle(pt, c);
46         }
47
48         public override void DrawOutline()
49         {
50             SplashKit.DrawCircle(Color.Black, X, Y, _radius + 2);
51         }
52
53         public override void SaveTo(StreamWriter writer)
54         {
55             writer.WriteLine("Circle");
56             base.SaveTo(writer);
57             writer.WriteLine(Radius);
58         }
59     }
```

```

60         public override void LoadFrom(StreamReader reader)
61         {
62             base.LoadFrom(reader);
63             Radius = reader.ReadInteger();
64         }
65     }
66 }

```

#### Week\_5/5.3/DrawingProgram/MyLine.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using SplashKitSDK;
6
7  namespace DrawingProgram
8  {
9      public class MyLine : Shape
10     {
11         private float _endX, _endY;
12
13         public MyLine() : this(Color.Red, 0.0f, 0.0f, 88, 88)
14         {
15         }
16
17         public MyLine(Color color, float x, float y, float endX, float endY) :
18         base(color)
19         {
20             X = x;
21             Y = y;
22             EndX = endX;
23             EndY = endY;
24         }
25
26         public float EndX
27         {
28             get => _endX;
29             set => _endX = value;
30         }
31
32         public float EndY
33         {
34             get => _endY;
35             set => _endY = value;
36         }
37
38         public override void Draw()
39         {
40             SplashKit.DrawLine(Color, X, Y, EndX, EndY);

```

```

41         if (Selected)
42         {
43             DrawOutline();
44         }
45     }
46
47     public override bool IsAt(Point2D pt)
48     {
49         return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y, EndX, EndY));
50     }
51
52     public override void DrawOutline()
53     {
54         SplashKit.DrawCircle(Color.Black, X, Y, 5);
55         SplashKit.DrawCircle(Color.Black, EndX, EndY, 5);
56     }
57
58     public override void SaveTo(StreamWriter writer)
59     {
60         writer.WriteLine("Line");
61         base.SaveTo(writer);
62         writer.WriteLine(EndX);
63         writer.WriteLine(EndY);
64     }
65
66     public override void LoadFrom(StreamReader reader)
67     {
68         base.LoadFrom(reader);
69         EndX = reader.ReadInteger();
70         EndY = reader.ReadInteger();
71     }
72 }
73 }

```

#### Week\_5/5.3/DrawingProgram/Program.cs

```

1  using System;
2  using SplashKitSDK;
3
4  namespace DrawingProgram
5  {
6      public class Program
7      {
8          private enum ShapeKind
9          {
10              Rectangle,
11              Circle,
12              Line
13          }
14
15          public static void Main()
16          {

```

```

17 Window window = new Window("Shape Drawer", 800, 600);
18 Drawing myDrawing = new Drawing();
19
20 ShapeKind kindToAdd = ShapeKind.Circle;
21
22 // My ID: 104844794 => Last digit: 4
23 // So I'm only able to draw a maximum of X lines within the timeframe
24 int maxLines = 4;
25
26 do
27 {
28     SplashKit.ProcessEvents();
29     SplashKit.ClearScreen();
30
31     if (maxLines <= 0 && kindToAdd == ShapeKind.Line)
32     {
33         kindToAdd = ShapeKind.Circle;
34     }
35
36     // If the user presses the L key and has lines left to draw, they
will draw lines
37     if (SplashKit.KeyTyped(KeyCode.LKey) && maxLines > 0)
38     {
39         kindToAdd = ShapeKind.Line;
40     }
41
42     if (SplashKit.KeyTyped(KeyCode.RKey))
43     {
44         kindToAdd = ShapeKind.Rectangle;
45     }
46
47     // If the user presses the C key or has run out of lines to draw,
they will draw circles
48     if (SplashKit.KeyTyped(KeyCode.CKey))
49     {
50         kindToAdd = ShapeKind.Circle;
51     }
52
53     if (SplashKit.KeyTyped(KeyCode.SKey))
54     {
55         myDrawing.Save("TestDrawing.txt");
56     }
57
58     if (SplashKit.KeyTyped(KeyCode.OKey))
59     {
60         try
61         {
62             myDrawing.Load("TestDrawing.txt");
63         }
64         catch (Exception e)
65         {

```



```

66         Console.Error.WriteLine("Error loading file: {0}",
e.Message);
67     }
68 }
69
70     if (SplashKit.MouseClicked(MouseButton.LeftButton))
71     {
72         Shape myShape;
73
74         switch (kindToAdd)
75         {
76             case ShapeKind.Circle:
77                 myShape = new MyCircle();
78                 break;
79             case ShapeKind.Line:
80                 myShape = new MyLine();
81                 maxLines--;
82                 break;
83             default:
84                 myShape = new MyRectangle();
85                 break;
86         }
87
88         myShape.X = SplashKit.MouseX();
89         myShape.Y = SplashKit.MouseY();
90         myDrawing.AddShape(myShape);
91     }
92
93     if (SplashKit.KeyTyped(KeyCode.SpaceKey))
94     {
95         myDrawing.Background = SplashKit.RandomRGBColor(255);
96     }
97
98     if (SplashKit.MouseClicked(MouseButton.RightButton))
99     {
100         Point2D pt = new Point2D();
101         pt.X = SplashKit.MouseX();
102         pt.Y = SplashKit.MouseY();
103
104         myDrawing.SelectShapesAt(pt);
105     }
106
107     if (SplashKit.KeyTyped(KeyCode.DeleteKey) ||
SplashKit.KeyTyped(KeyCode.BackspaceKey))
108     {
109         myDrawing.RemoveSelectedShapes();
110     }
111
112     myDrawing.Draw();
113     SplashKit.RefreshScreen();
114 } while (!window.CloseRequested);

```

```
115     }
116 }
117 }
118
```

### Week\_5/5.3/DrawingProgram/MyRectangle.cs

```
1  using System;
2  using System.IO;
3  using System.Collections.Generic;
4  using SplashKitSDK;
5  using System.Linq;
6  using System.Threading.Tasks;
7
8  namespace DrawingProgram
9  {
10     public class MyRectangle : Shape
11     {
12         private float _width, _height;
13
14         public MyRectangle() : this(Color.Green, 0.0f, 0.0f, 194, 194)
15         {
16         }
17
18         public MyRectangle(Color color, float x, float y, float width, float height)
19         : base(color)
20         {
21             X = x;
22             Y = y;
23             Width = width;
24             Height = height;
25         }
26
27         public float Width
28         {
29             get => _width;
30             set => _width = value;
31         }
32
33         public float Height
34         {
35             get => _height;
36             set => _height = value;
37         }
38
39         // Methods
40         public override void Draw()
41         {
42             SplashKit.FillRectangle(Color, X, Y, Width, Height);
43
44             if (Selected)
45             {

```

```

45         DrawOutline();
46     }
47 }
48
49 public override bool IsAt(Point2D pt)
50 {
51     return (pt.X >= X) && (pt.X <= (X + _width))
52         && (pt.Y >= Y) && (pt.Y <= (Y + _height));
53 }
54
55 public override void DrawOutline()
56 {
57     SplashKit.DrawRectangle(Color.Black, X - 9, Y - 9, Width + 18, Height +
58 18);
59 }
60
61 public override void SaveTo(StreamWriter writer)
62 {
63     writer.WriteLine("Rectangle");
64     base.SaveTo(writer);
65     writer.WriteLine(Width);
66     writer.WriteLine(Height);
67 }
68
69 public override void LoadFrom(StreamReader reader)
70 {
71     base.LoadFrom(reader);
72     Width = reader.ReadInteger();
73     Height = reader.ReadInteger();
74 }
75 }

```

### Week\_5/5.3/DrawingProgram/Shape.cs

```

1 using SplashKitSDK;
2
3 namespace DrawingProgram
4 {
5     public abstract class Shape
6     {
7         // Fields
8         private Color _color;
9         private float _x, _y;
10        private bool _selected;
11
12        // Constructor
13        public Shape() : this(Color.Yellow)
14        {
15        }
16
17        public Shape(Color color)

```

```
18     {
19         _color = color;
20         _x = 0.0f;
21         _y = 0.0f;
22         _selected = false;
23     }
24
25     // Properties
26     public float X
27     {
28         get { return _x; }
29         set { _x = value; }
30     }
31
32     public float Y
33     {
34         get { return _y; }
35         set { _y = value; }
36     }
37
38     public Color Color
39     {
40         get { return _color; }
41         set { _color = value; }
42     }
43
44     public bool Selected
45     {
46         get { return _selected; }
47         set { _selected = value; }
48     }
49
50     // Methods
51     public abstract void Draw();
52     public abstract void DrawOutline();
53     public abstract bool IsAt(Point2D pt);
54     public virtual void SaveTo(StreamWriter writer)
55     {
56         writer.WriteColor(Color);
57         writer.WriteLine(X);
58         writer.WriteLine(Y);
59     }
60
61     public virtual void LoadFrom(StreamReader reader)
62     {
63         Color = reader.ReadColor();
64         X = reader.ReadInteger();
65         Y = reader.ReadInteger();
66     }
67
68 }
```

**Week\_5/5.3/DrawingProgram/TestDrawing.txt**

```
1
1
1
6
Circle
0
0
1
248
110
129
Rectangle
0
0.5
0
452
115
194
194
Rectangle
0
0.5
0
634
53
194
194
Line
1
0
0
158
474
88
88
Line
1
0
0
233
471
88
88
Line
1
0
0
348
453
88
88
```

## Screenshot of running program:

