```
1  using System;
2  using System.IO;
3  using SplashKitSDK;
4
5  namespace DrawingShape
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 }
28
```

```
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 DrawingShape
9  {
10     public abstract class Shape
11     {
12         private Color _color;
13         private float _x, _y;
14         private int _width, _height;
15         private bool _selected;
16
17         public Shape(Color color)
18         {
19             _color = Color.Green;
20             _x = 0;
21             _y = 0;
22             _width = 100;
23             _height = 100;
24         }
25         public Color Color
26         {
27             get { return _color; }
28             set { _color = value; }
29         }
30         public float X
31         {
32             get { return _x; }
33             set { _x = value; }
34         }
35         public float Y
36         {
37             get { return _y; }
38             set { _y = value; }
39         }
40
41         public bool Selected { get; internal set; }
42
43         public virtual void SaveTo(StreamWriter writer)
44         {
45             writer.WriteColor(_color);
46             writer.WriteLine(_x);
47             writer.WriteLine(_y);
48         }
49
```

```
50          public virtual void LoadFrom(StreamReader reader)
51          {
52              _color = reader.ReadColor();
53              _x = reader.ReadInteger();
54              _y = reader.ReadInteger();
55          }
56
57          public abstract void Draw();
58
59          public abstract bool IsAt(Point2D pt);
60
61          public abstract void DrawOutline();
62      }
63  }
64
```

```
 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 DrawingShape
 9 {
10     internal class Drawing
11     {
12         private readonly List<Shape> _shapes;
13         private Color _background;
14
15         public Color Background
16         {
17             get { return _background; }
18             set { _background = value; }
19         }
20
21         public List<Shape> Shapes
22         {
23             get { return _shapes; }
24         }
25
26         public Drawing(Color background)
27         {
28             _shapes = new List<Shape>();
29             _background = background;
30         }
31
32         public Drawing() : this(Color.White)
33         {
34
35         }
36
37         public int ShapeCount
38         {
39             get { return _shapes.Count; }
40         }
41
42         public void AddShape(Shape s)
43         {
44             _shapes.Add(s);
45         }
46
47         public void Draw()
48         {
49             SplashKit.ClearScreen(_background);
```

```
50              foreach (Shape s in _shapes)
51              {
52                  s.Draw();
53              }
54              SplashKit.RefreshScreen();
55          }
56
57          public List<Shape> SelectedShapes
58          {
59              get
60              {
61                  List<Shape> result = new List<Shape>();
62                  foreach (Shape s in _shapes)
63                  {
64                      if (s.Selected)
65                      {
66                          result.Add(s);
67                      }
68                  }
69                  return result;
70              }
71          }
72
73          public void SelectShapesAt(Point2D pt)
74          {
75              foreach (Shape s in _shapes)
76              {
77                  if(s.IsAt(pt))
78                  {
79                      s.Selected = true;
80                  }
81                  else
82                  {
83                      s.Selected = false;
84                  }
85              }
86          }
87
88          public void Save(string filename)
89          {
90              StreamWriter writer = new StreamWriter(filename);
91              try
92              {
93                  writer.WriteColor(_background);
94                  writer.WriteLine(ShapeCount);
95                  foreach (Shape s in _shapes)
96                  {
97                      s.SaveTo(writer);
98                  }
```

```
 99              }
100              finally
101              {
102                  writer.Close();
103              }
104          }
105
106          public void Load(string filename)
107          {
108              StreamReader reader = new StreamReader(filename);
109              try
110              {
111                  _background = reader.ReadColor();
112                  int count = reader.ReadInteger();
113                  for (int i = 0; i < count; i++)
114                  {
115                      string kind = reader.ReadLine();
116                      Shape s;
117                      switch (kind)
118                      {
119                          case "Line":
120                              s = new MyLine();
121                              break;
122                          case "Rectangle":
123                              s = new MyRectangle();
124                              break;
125                          case "Circle":
126                              s = new MyCircle();
127                              break;
128                          default:
129                              throw new InvalidDataException("Unknown shape
                                kind: " + kind);
130                      }
131                      s.LoadFrom(reader);
132                      _shapes.Add(s);
133                  }
134              }
135              finally
136              {
137                  reader.Close();
138              }
139          }
140      }
141  }
142
```

```
1  using SplashKitSDK;
2
3  namespace DrawingShape
4  {
5      internal class MyRectangle : Shape
6      {
7          private int _width, _height;
8
9          public MyRectangle() : base(color: Color.Green)
10         {
11             _width = 100;
12             _height = 100;
13         }
14
15         public MyRectangle(Color color, int x, int y, int width, int
             height) : base(color)
16         {
17             Color = color;
18             X = x;
19             Y = y;
20             Width = width;
21             Height = height;
22         }
23
24         public int Width
25         {
26             get { return _width; }
27             set { _width = value; }
28         }
29
30         public int Height
31         {
32             get { return _height; }
33             set { _height = value; }
34         }
35
36         public override void SaveTo(StreamWriter writer)
37         {
38             writer.WriteLine("Rectangle");
39             base.SaveTo(writer);
40             writer.WriteLine(_width);
41             writer.WriteLine(_height);
42         }
43
44         public override void LoadFrom(StreamReader reader)
45         {
46             base.LoadFrom(reader);
47             _width = reader.ReadInteger();
48             _height = reader.ReadInteger();
```

```
49          }
50
51          public override void Draw()
52          {
53              if (Selected)
54              {
55                  DrawOutline();
56              }
57              SplashKit.FillRectangle(Color, X, Y, _width, _height);
58          }
59
60          public override void DrawOutline()
61          {
62              SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, _width + 4,
                  _height + 4);
63          }
64
65          public override bool IsAt(Point2D pt)
66          {
67              return SplashKit.PointInRectangle(pt, SplashKit.RectangleFrom
                  (X, Y, _width, _height));
68          }
69      }
70  }
71
```

```
1  using SplashKitSDK;
2
3  namespace DrawingShape
4  {
5      internal class MyCircle : Shape
6      {
7          int _radius;
8          public MyCircle() : base(color: Color.Green)
9          {
10             _radius = 50;
11         }
12
13         public MyCircle(Color color, int x, int y, int radius) : base
             (color)
14         {
15             Color = color;
16             X = x;
17             Y = y;
18             _radius = radius;
19         }
20
21         public int Radius
22         {
23             get { return _radius; }
24             set { _radius = value; }
25         }
26
27         public override void SaveTo(StreamWriter writer)
28         {
29             writer.WriteLine("Circle");
30             base.SaveTo(writer);
31             writer.WriteLine(_radius);
32         }
33
34         public override void LoadFrom(StreamReader reader)
35         {
36             base.LoadFrom(reader);
37             _radius = reader.ReadInteger();
38         }
39
40         public override void Draw()
41         {
42             if (Selected)
43             {
44                 DrawOutline();
45             }
46             SplashKit.FillCircle(Color, X, Y, _radius);
47         }
48
```

```
49          public override void DrawOutline()
50          {
51              SplashKit.FillCircle(Color.Black, X, Y, _radius+2);
52          }
53
54          public override bool IsAt(Point2D pt)
55          {
56              double a = (double)(pt.X - X);
57              double b = (double)(pt.Y - Y);
58              if (Math.Sqrt(a * a + b * b) < _radius)
59              {
60                  return true;
61              }
62              return false;
63          }
64      }
65  }
66
```

```
1  using SplashKitSDK;
2
3  namespace DrawingShape
4  {
5      public class MyLine : Shape
6      {
7          private float _endX, _endY;
8
9          public MyLine() : this(Color.Green)
10         {
11         }
12
13         public MyLine(Color color) : base(color)
14         {
15             Color = color;
16             _endX = 700;
17             _endY = 500;
18         }
19
20         public float EndX
21         {
22             get { return _endX; }
23             set { _endX = value; }
24         }
25
26         public float EndY
27         {
28             get { return _endY; }
29             set { _endY = value; }
30         }
31
32         public override void SaveTo(System.IO.StreamWriter writer)
33         {
34             writer.WriteLine("Line");
35             base.SaveTo(writer);
36             writer.WriteLine(_endX);
37             writer.WriteLine(_endY);
38         }
39
40         public override void LoadFrom(System.IO.StreamReader reader)
41         {
42             base.LoadFrom(reader);
43             _endX = reader.ReadSingle();
44             _endY = reader.ReadSingle();
45         }
46
47         public override void Draw()
48         {
49             if(Selected)
```

```
50              {
51                  DrawOutline();
52              }
53              SplashKit.DrawLine(Color, X, Y, _endX, _endY);
54          }
55
56          public override void DrawOutline()
57          {
58              SplashKit.FillCircle(Color.Black, X, Y, 5);
59              SplashKit.FillCircle(Color.Black, _endX, _endY, 5);
60          }
61
62          public override bool IsAt(Point2D pt)
63          {
64              return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y,
                  _endX, _endY));
65          }
66
67      }
68  }
69
```

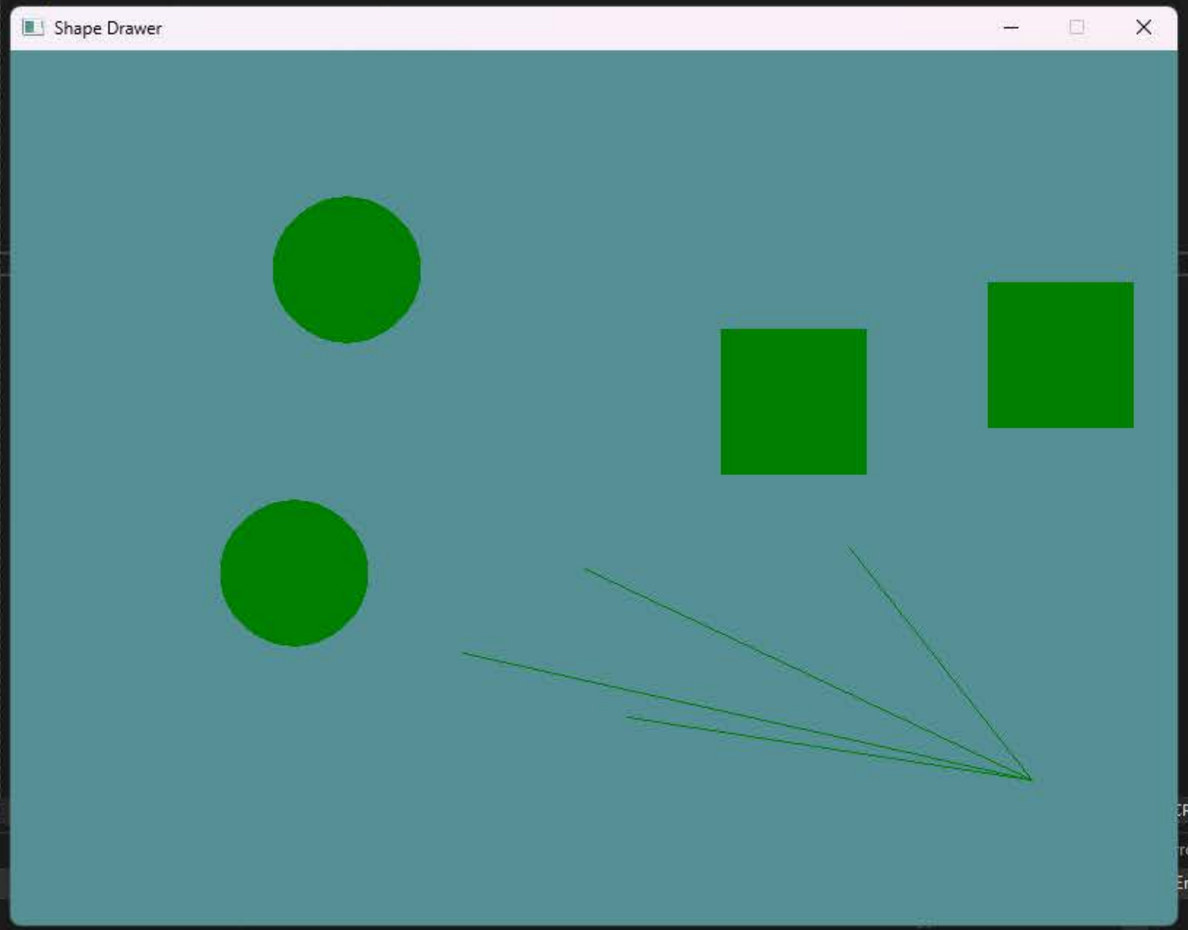
```
 1 using System;
 2 using System.Collections.Generic;
 3 using SplashKitSDK;
 4
 5 namespace DrawingShape
 6 {
 7     public class Program
 8     {
 9         private enum ShapeKind
10         {
11             Rectangle,
12             Circle,
13             Line
14         }
15
16         public static void Main()
17         {
18             ShapeKind kindToAdd = ShapeKind.Circle;
19             Window window = new Window("Shape Drawer", 800, 600);
20             Drawing myDrawing = new Drawing();
21             do
22             {
23                 SplashKit.ProcessEvents();
24                 SplashKit.ClearScreen();
25                 if (SplashKit.KeyTyped(KeyCode.RKey))
26                 {
27                     kindToAdd = ShapeKind.Rectangle;
28                 }
29                 if (SplashKit.KeyTyped(KeyCode.CKey))
30                 {
31                     kindToAdd = ShapeKind.Circle;
32                 }
33                 if (SplashKit.KeyTyped(KeyCode.LKey))
34                 {
35                     kindToAdd = ShapeKind.Line;
36                 }
37                 if (SplashKit.MouseClicked(MouseButton.LeftButton))
38                 {
39                     Shape newShape;
40                     switch (kindToAdd)
41                     {
42                         case ShapeKind.Circle:
43                             newShape = new MyCircle();
44                             newShape.X = SplashKit.MouseX();
45                             newShape.Y = SplashKit.MouseY();
46                             break;
47
48                         case ShapeKind.Line:
49                             newShape = new MyLine();
```

```
50                              newShape.X = SplashKit.MouseX();
51                              newShape.Y = SplashKit.MouseY();
52                              break;
53
54                          default:
55                              newShape = new MyRectangle();
56                              newShape.X = SplashKit.MouseX();
57                              newShape.Y = SplashKit.MouseY();
58
59                              break;
60                      }
61                      myDrawing.AddShape(newShape);
62                  }
63                  Point2D pt = SplashKit.MousePosition();
64                  if(SplashKit.KeyTyped(KeyCode.SpaceKey))
65                  {
66                      myDrawing.Background = SplashKit.RandomRGBColor(255);
67                  }
68                  if(SplashKit.MouseClicked(MouseButton.RightButton))
69                  {
70                      foreach(Shape s in myDrawing.Shapes)
71                      {
72                          if(s.IsAt(pt))
73                          {
74                              s.Selected = !s.Selected;
75                          }
76                      }
77                  }
78                  if(SplashKit.KeyTyped(KeyCode.DeleteKey)||
                      SplashKit.KeyTyped(KeyCode.BackspaceKey))
79                  {
80                      for (int i = myDrawing.ShapeCount - 1; i >= 0; i--)
81                      {
82                          if (myDrawing.Shapes[i].Selected)
83                          {
84                              myDrawing.Shapes.RemoveAt(i);
85                          }
86                      }
87                  }
88                  if (SplashKit.KeyTyped(KeyCode.SKey))
89                  {
90                      try
91                      {
92                          myDrawing.Save("D:\\workspace\\COS20007\\COS20007-
                       working\\5.3C - Drawing Program - Saving and Loading\
                       \DrawingShape\\TestDrawing.txt");
93                      }
94                      catch (Exception e)
95                      {
```

```
 96                          Console.WriteLine("Error saving file: " +
                         e.Message);
 97                          }
 98                  }
 99                  if (SplashKit.KeyTyped(KeyCode.OKey))
100                  {
101                      try
102                      {
103                          myDrawing.Load("D:\\workspace\\COS20007\\COS20007-
                         working\\5.3C - Drawing Program - Saving and Loading\
                         \DrawingShape\\TestDrawing.txt");
104                      }
105                      catch (Exception e)
106                      {
107                          Console.WriteLine("Error loading file: " +
                         e.Message);
108                      }
109                  }
110                  myDrawing.Draw();
111                  SplashKit.RefreshScreen();
112              } while (!window.CloseRequested);
113          }
114      }
115  }
116
```

DrawingShape.Drawing | SelectShapesAt(Point2D pt)

```
0 references
public void SelectShapesAt(Point2D pt)
{
```




```
0.336375
0.5638905
0.58464307
8
Circle
0
0.5
0
230
150
50
Circle
0
0.5
0
194
358
50
Rectangle
0
0.5
0
487
191
100
100
Rectangle
0
0.5
0
670
159
100
100
Line
0
0.5
```

No issues found | Ln: 67 | Ch: 1 | TABS | CR

Error List

Entire Solution | 0 Errors | 1 Warning | 0 of 7 Messages | Build + IntelliSense | Search Error List

Code	Description	Project	File	Line	Suppression State
CS8600	Converting null literal or possible null value to non-nullable type.	DrawingShape	Drawing.cs	115	Active