SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Drawing Program - Saving and Loading

PDF generated at 16:09 on Wednesday $4^{\rm th}$ October, 2023

File 1 of 8 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
                             newCircle.X = SplashKit.MouseX();
42
                              newCircle.Y = SplashKit.MouseY();
43
                              newShape = newCircle;
45
                         }
46
                         else if (kindToAdd == ShapeKind.Rectangle)
47
48
                             MyRectangle newRect = new MyRectangle();
49
                             newRect.X = SplashKit.MouseX();
50
                             newRect.Y = SplashKit.MouseY();
51
                             newShape = newRect;
52
                         }
53
```

File 1 of 8 Program class

```
else
54
                          {
55
                              MyLine newLine = new MyLine();
56
                              newLine.X = SplashKit.MouseX();
                              newLine.Y = SplashKit.MouseY();
58
                              newShape = newLine;
59
                          }
60
                          myDrawing.AddShape(newShape);
61
                     if (SplashKit.KeyTyped(KeyCode.SKey))
63
                     {
64
                          string folder =
65
        "C:/Users/lequa/OneDrive/Documents/COS20007/ShapeDrawer 5.3/TestDrawing.txt";
                          try
66
                          {
67
                              myDrawing.Save(folder);
                              Console.WriteLine($"Drawing saved {folder}");
69
                          }
70
                          catch (Exception e)
71
                          {
72
                              Console.Error.WriteLine("Error saviing file: {0}",
        e.Message);
                          }
74
                          /* string saveFolderPath =
75
        @"C:\Users\lequa\OneDrive\Documents\COS20007\ShapeDrawer 5.3";
                           string saveFileName = "TestDrawing.txt";
76
                           string savePath = System. IO. Path. Combine (saveFolderPath,
        saveFileName);
                           myDrawing.Save(savePath);
78
79
                           Console. WriteLine($"Drawing saved to {savePath}"); */
80
                     }
81
                     if (SplashKit.KeyTyped(KeyCode.OKey))
                     {
83
                          string folder =
84
        "C:/Users/lequa/OneDrive/Documents/COS20007/ShapeDrawer 5.3/TestDrawing.txt";
85
                          try
86
                          {
87
                              myDrawing.Load(folder);
                              Console.WriteLine($"Drawing loaded from {folder} ");
89
90
                          }
91
                          catch (Exception e)
92
                              Console.Error.WriteLine("Error loadding file: {0}",
94
        e.Message);
                          }
95
96
                     }
98
                        (SplashKit.MouseClicked(MouseButton.RightButton))
                     if
99
                     {
100
```

File 1 of 8 Program class

```
myDrawing.SelectedShapeAt(SplashKit.MousePosition());
101
                      }
102
103
                      if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
        SplashKit.KeyTyped(KeyCode.DeleteKey))
                      {
105
                          myDrawing.RemoveShape();
106
                      }
107
108
                      if (SplashKit.KeyTyped(KeyCode.SpaceKey))
109
                      {
110
                          myDrawing.Background = SplashKit.RandomRGBColor(255);
111
                      }
112
113
                     myDrawing.Draw();
114
                      SplashKit.RefreshScreen();
116
117
                 } while (!SplashKit.WindowCloseRequested("Drawing Shape"));
118
             }
119
        }
120
    }
121
```

File 2 of 8 ExtensionMethods class

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

File 3 of 8 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
            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 _selectedShapes)
26
27
                     if (s.Selected)
29
                          _selectedShapes.Add(s);
30
                     }
31
                 }
32
                 return _selectedShapes;
            }
34
35
            public int ShapeCount
36
37
                 get
38
                 {
39
                     return _shapes.Count;
40
                 }
41
            }
42
43
            public Color Background
                 get
46
                 {
47
                     return _background;
48
                 }
49
                 set
50
                 {
51
                     _background = value;
52
53
```

File 3 of 8 Drawing class

```
}
54
55
             public void Draw()
56
                 SplashKit.ClearScreen(_background);
58
                 foreach (Shape s in _shapes)
60
61
                      s.Draw();
             }
64
65
             public void SelectedShapeAt(Point2D pt)
66
67
                  foreach (Shape s in _shapes)
68
                      if (s.IsAt(pt))
70
                      {
                           s.Selected = true;
72
                      }
73
                      else
                      {
                           s.Selected = false;
                  }
             }
79
             public void AddShape(Shape s)
82
                  _shapes.Add(s);
84
85
             public void RemoveShape()
87
                 foreach (Shape s in _shapes.ToList())
                  {
89
                      if (s.Selected)
90
                           _shapes.Remove(s);
92
                      }
93
                 }
94
             }
95
             public void Save(string filename)
96
             {
                 StreamWriter writer = new StreamWriter(filename);
99
                 writer.WriteColor(Background);
100
                 writer.WriteLine(ShapeCount);
101
                  foreach(Shape s in _shapes)
102
103
                      s.SaveTo(writer);
104
105
                 writer.Close();
106
```

File 3 of 8 Drawing class

```
}
107
             public void Load (string filename)
108
             {
109
                  StreamReader reader = new StreamReader(filename);
                  Background = reader.ReadColor();
111
                  int count = reader.ReadInteger();
112
                  _shapes.Clear();
113
                  for (int i = 0; i < count; i++)
114
                      string kind = reader.ReadLine();
116
                      Shape s;
117
                      if (kind == "Rectangle")
118
119
                           s = new MyRectangle();
120
                      } else if (kind == "Circle")
121
122
                           s = new MyCircle();
123
                      }else
124
125
                           continue;
126
                      }
                      s.LoadFrom(reader);
128
                      _shapes.Add(s);
129
130
                  }
131
                  reader.Close();
132
133
             }
134
135
         }
136
    }
137
```

File 4 of 8 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 4 of 8 Shape class

```
54
55
            public bool Selected
56
                 get
58
                 {
59
                     return _selected;
60
                 }
61
62
                 set
                 {
63
                      _selected = value;
64
                 }
65
            }
66
67
            public abstract void DrawOutline();
68
            public virtual void SaveTo(StreamWriter writer)
70
                 writer.WriteColor(Color);
71
                 writer.WriteLine(X);
72
                 writer.WriteLine(Y);
73
            }
            public virtual void LoadFrom(StreamReader reader)
75
            {
76
                 Color = reader.ReadColor();
77
                 X = reader.ReadInteger();
78
                 Y = reader.ReadInteger();
79
            }
80
        }
81
   }
82
```

File 5 of 8 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 5 of 8 MyRectangle class

```
}
53
                }
54
                return false;
55
            }
            public override void SaveTo(StreamWriter writer)
57
             {
58
                writer.WriteLine("Rectangle");
59
                base.SaveTo(writer);
60
                writer.WriteLine(Width);
                writer.WriteLine(Height);
62
            }
63
            public override void LoadFrom(StreamReader reader)
64
65
                base.LoadFrom(reader);
66
                Width = reader.ReadInteger();
67
                Height = reader.ReadInteger();
            }
69
        }
70
   }
71
```

File 6 of 8 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; } set { _radius = value; } }
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
            public override void SaveTo(StreamWriter writer)
41
42
                writer.WriteLine("Circle");
43
                base.SaveTo(writer);
                writer.WriteLine(Radius);
            }
46
            public override void LoadFrom(StreamReader reader)
47
48
                base.LoadFrom(reader);
49
                Radius = reader.ReadInteger();
50
            }
51
        }
52
   }
53
```

File 7 of 8 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)
            {
                X = startX;
16
                Y = startY;
17
                _endX = endX;
18
                _endY = endY;
19
21
22
            public MyLine() : this(Color.RandomRGB(255), 0, 0, 20, 20) { }
23
24
            public float EndX
25
26
                get { return _endX; }
                set { _endX = value; }
28
29
            public float EndY
30
31
                get { return _endY; }
                set { _endY = value; }
33
34
35
            public override void Draw()
36
            {
                if (Selected)
38
                {
39
                    DrawOutline();
40
41
                SplashKit.DrawLine(Color, X, Y, _endX, _endY);
42
43
            public override void DrawOutline()
            {
45
                SplashKit.DrawCircle(Color.Black, X, Y, 5);
46
                SplashKit.DrawCircle(Color.Black, _endX, _endY, 5);
47
48
            public override bool IsAt(Point2D p)
49
            {
50
                // Calculate the distance from the point to the line
51
                double distance = Math.Abs((EndY - Y) * p.X - (EndX - X) * p.Y + EndX * Y
52
        - EndY * X)
```

File 7 of 8 MyLine class

```
/ Math.Sqrt(Math.Pow(EndY - Y, 2) + Math.Pow(EndX - X,
53
       2));
54
                // Define a tolerance value for how close the point can be to the line
                double tolerance = 5.0; // Adjust as needed
56
57
                // Check if the distance is within the tolerance
58
                return distance <= tolerance;</pre>
59
60
            public override void SaveTo(StreamWriter writer)
61
            {
62
                writer.WriteLine("Line");
63
                base.SaveTo(writer); // This will write Color, X, and Y
64
                writer.WriteLine(EndX); // Write EndX
65
                writer.WriteLine(EndY); // Write EndY
66
            }
68
            public override void LoadFrom(StreamReader reader)
69
70
                base.LoadFrom(reader); // This reads Color, X, and Y
71
                EndX = reader.ReadSingle(); // Read EndX
                EndY = reader.ReadSingle(); // Read EndY
73
            }
74
        }
75
   }
76
```

