Swinburne University of Technology

COS20007 Object Oriented Programming

Drawing Program - Saving and Loading

PDF generated at 04:15 on Monday 9th October, 2023

```
using System;
using SplashKitSDK;

namespace MyGame
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }
        public static void Main()
        {
            ShapeKind kindToAdd = ShapeKind.Circle;
            Drawing myDraw = new();
            Window window = new("Shape Drawer 3", 800, 600); //draw window
            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen(Color.White);
                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape myShape;
                    if (kindToAdd == ShapeKind.Circle)
                    {
                        MyCircle myCircle = new();
                        myShape = myCircle;
                    }
                    else if (kindToAdd == ShapeKind.Rectangle)
                    {
                        MyRectangle myRect = new();
                        myShape = myRect;
                    }
                    else
                    {
                        MyLine myLine = new();
                        myShape = myLine;
                    }
                    myShape.X = SplashKit.MouseX();
                    myShape.Y = SplashKit.MouseY();
                    myDraw.AddShape(myShape);
                }
                if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                {
                    myDraw.Background = SplashKit.RandomColor();
                }
                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    myDraw.SelectShapesAt(SplashKit.MousePosition());
                }
                List<Shape> select = new();
```

```
54                  select = myDraw.SelectedShapes;
55                  if (SplashKit.KeyTyped(KeyCode.DeleteKey) ||
    ↪  SplashKit.KeyTyped(KeyCode.BackspaceKey))
56                  {
57                      foreach (Shape s in select)
58                      {
59                          myDraw.RemoveShape(s);
60                      }
61                  }
62                  if (SplashKit.KeyTyped(KeyCode.RKey))
63                  {
64                      kindToAdd = ShapeKind.Rectangle;
65                  }
66                  if (SplashKit.KeyTyped(KeyCode.CKey))
67                  {
68                      kindToAdd = ShapeKind.Circle;
69                  }
70                  if (SplashKit.KeyTyped(KeyCode.LKey))
71                  {
72                      kindToAdd = ShapeKind.Line;
73                  }
74                  if (SplashKit.KeyTyped(KeyCode.SKey))
75                  {
76                      myDraw.Save("TestDrawing.txt");
77                  }
78                  if (SplashKit.KeyTyped(KeyCode.OKey))
79                  {
80                      try
81                      {
82                          myDraw.Load("TestDrawing.txt");
83                      }
84                      catch (Exception e)
85                      {
86                          Console.Error.WriteLine("Error loading file: {0}",
    ↪  e.Message);
87                      }
88                  }
89                  myDraw.Draw();
90                  SplashKit.RefreshScreen();
91              } while (!SplashKit.WindowCloseRequested("Shape Drawer 3"));
92          }
93      }
94  }
```

```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MyGame
{
    public static class ExtensionMethods
    {
        public static int ReadInteger(this StreamReader reader)
        {
            return Convert.ToInt32(reader.ReadLine());
        }

        public static float ReadSingle(this StreamReader reader)
        {
            return Convert.ToSingle(reader.ReadLine());
        }

        public static Color ReadColor(this StreamReader reader)
        {
            return Color.RGBColor(reader.ReadSingle(), reader.ReadSingle(),
↪   reader.ReadSingle());
        }

        public static void WriteColor(this StreamWriter writer, Color clr)
        {
            writer.WriteLine("{0}\n{1}\n{2}", clr.R, clr.G, clr.B);
        }
    }
}
```

```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MyGame
{
    public class Drawing
    {
        public readonly List<Shape> _shapes;
        private Color _background;
        public Color Background
        {
            get
            {
                return _background;
            }
            set
            {
                _background = value;
            }
        }
        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;
        }
        public Drawing() : this(Color.White) { }
        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> result = new();
                foreach (Shape s in _shapes)
                {
                    if (s.Selected == true)
                    {
                        result.Add(s);
                    }
                }
                return result;
            }
        }
        public int ShapeCount
        {
            get
            {
                return _shapes.Count;
            }
        }
        public void AddShape(Shape shape)
```

```
54          {
55              _shapes.Add(shape);
56          }
57
58          public void Draw()
59          {
60              SplashKit.ClearScreen(_background);
61              foreach (Shape shape in _shapes)
62              {
63                  shape.Draw();
64              }
65          }
66          public void SelectShapesAt(Point2D pt)
67          {
68              foreach (Shape s in _shapes)
69              {
70                  if (s.IsAt(pt))
71                  {
72                      s.Selected = true;
73                  }
74                  else
75                  {
76                      s.Selected = false;
77                  }
78              }
79          }
80
81          public void RemoveShape(Shape shape)
82          {
83              _shapes.Remove(shape);
84          }
85
86          public void Save(string filename)
87          {
88              StreamWriter writer = new StreamWriter(filename);
89              writer.WriteColor(Background);
90              writer.WriteLine(ShapeCount);
91              foreach (Shape s in _shapes)
92              {
93                  s.SaveTo(writer);
94              }
95              writer.Close();
96          }
97          public void Load(string filename)
98          {
99              StreamReader reader = new StreamReader(filename);
100             try
101             {
102                 Background = reader.ReadColor();
103                 int count = reader.ReadInteger();
104                 Shape s;
105                 string kind;
106                 _shapes.Clear();
```

```
107                 for (int i = 0; i < count; i++)
108                 {
109                     kind = reader.ReadLine();
110                     switch (kind)
111                     {
112                         case "Rectangle":
113                             s = new MyRectangle();
114                             break;
115                         case "Circle":
116                             s = new MyCircle();
117                             break;
118                         case "Line":
119                             s = new MyLine();
120                             break;
121                         default:
122                             throw new InvalidDataException("Uknown shape kind: " +
    ↪  kind);
123                     }
124                     s.LoadFrom(reader);
125                     AddShape(s);
126                 }
127             }
128             finally
129             {
130                 reader.Close();
131             }
132         }
133     }
134 }
```

```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MyGame
{
    public abstract class Shape
    {
        private bool _selected;
        private float _x, _y;
        private Color _color;
        public float X
        {
            get
            {
                return _x;
            }
            set
            {
                _x = value;
            }
        }
        public float Y
        {
            get
            {
                return _y;
            }
            set
            {
                _y = value;
            }
        }
        public bool Selected
        {
            get
            {
                return _selected;
            }
            set
            {
                _selected = value;
            }
        }
        public Color Color
        {
            get
            {
                return _color;
            }
```

```
54              set
55              {
56                  _color = value;
57              }
58          }
59          public Shape()
60          {
61              _color = SplashKit.ColorYellow();
62          }
63          public Shape(Color color)
64          {
65              _color = color;
66          }
67          public abstract bool IsAt(Point2D pt);
68
69          public abstract void DrawOutline();
70          public abstract void Draw();
71
72          public virtual void SaveTo(StreamWriter writer)
73          {
74              writer.WriteColor(Color);
75              writer.WriteLine(X);
76              writer.WriteLine(Y);
77          }
78          public virtual void LoadFrom(StreamReader reader)
79          {
80              Color = reader.ReadColor();
81              X = reader.ReadInteger();
82              Y = reader.ReadInteger();
83          }
84      }
85  }
```

```
using SplashKitSDK;
using Color = SplashKitSDK.Color;

namespace MyGame
{
    public class MyRectangle : Shape
    {
        private int _width, _height;
        public int Width
        {
            get
            {
                return _width;
            }
            set
            {
                _width = value;
            }
        }
        public int Height
        {
            get
            {
                return _height;
            }
            set
            {
                _height = value;
            }
        }
        public MyRectangle()
        {
            X = 0;
            Y = 0;
            Width = 100;
            Height = 100;
            Color = SplashKit.ColorGreen();
        }
        public MyRectangle(Color clr, float x, float y, int width, int height) :
    ↪  base(clr)
        {
            X = x;
            Y = y;
            Width = width;
            Height = height;
        }
        public override void Draw()
        {
            if (Selected) { DrawOutline(); }
            SplashKit.FillRectangle(Color, X, Y, Width, Height); //draw shape
        }
        public override void DrawOutline()
        {
```

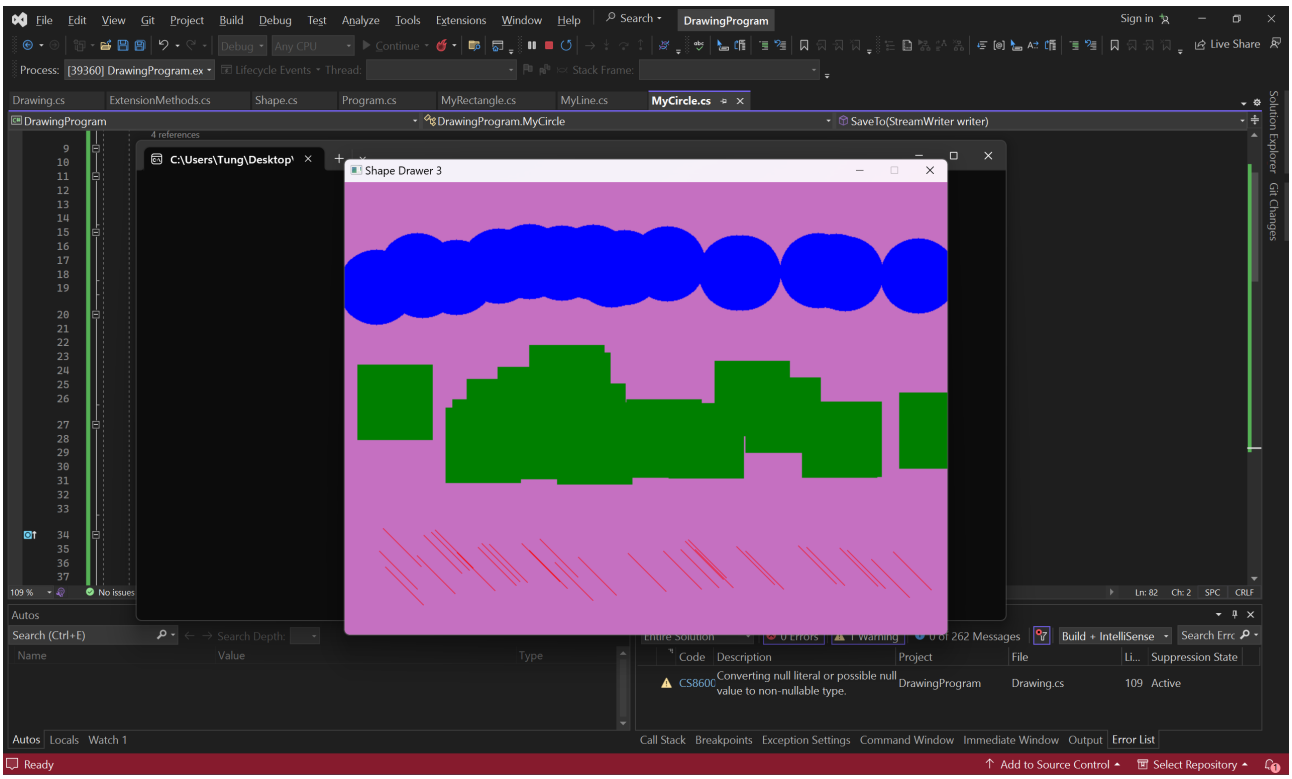
```
53              SplashKit.FillRectangle(SplashKit.ColorBlack(), X - 2, Y - 2, Width + 4,
    ↪  Height + 4); //draw shape
54          }
55          public override bool IsAt(Point2D pt) //the result return bool so need to set
    ↪  bool here, pt is param
56          {
57              if (pt.X >= X && pt.X <= X + Width && pt.Y >= Y && pt.Y <= Y + Height)
58              // mouse x-coor >= shape x-coor && mouse x-coor <= shape x-coor + shape
    ↪  width
59              // mouse y-coor >= shape y-coor && mouse y-coor <= shape y-coor + height
60              {
61                  return true; // before was true
62              }
63              else
64              {
65                  return false;
66              }
67          }
68
69          public override void SaveTo(StreamWriter writer)
70          {
71              writer.WriteLine("Rectangle");
72              base.SaveTo(writer);
73              writer.WriteLine(Width);
74              writer.WriteLine(Height);
75          }
76          public override void LoadFrom(StreamReader reader)
77          {
78              base.LoadFrom(reader);
79              Width = reader.ReadInteger();
80              Height = reader.ReadInteger();
81          }
82      }
83  }
```

```
using SplashKitSDK;
using Color = SplashKitSDK.Color;

namespace MyGame
{
    public class MyCircle : Shape
    {
        private int _radius;
        public int Radius
        {
            get
            {
                return _radius;
            }
            set
            {
                _radius = value;
            }
        }
        public MyCircle()
        {
            X = 0;
            Y = 0;
            _radius = 50;
            Color = SplashKit.ColorBlue();
        }
        public MyCircle(Color color, int x, int y, int radius)
        {
            Color = color;
            X = x;
            Y = y;
            _radius = radius;
        }
        public override void Draw()
        {
            if (Selected) DrawOutline();
            SplashKit.FillCircle(Color, X, Y, _radius);
        }
        public override void DrawOutline() //overidde
        {
            SplashKit.FillCircle(SplashKit.ColorBlack(), X, Y, _radius + 2); //draw
↪   shape
        }
        public override bool IsAt(Point2D pt)
        {
            //check if the mouse point in the circle or not /  return the centered
↪   point of circle
            /*if (SplashKit.PointInCircle(pt, SplashKit.CircleAt(X + Radius, Y +
↪   Radius, Radius)))
            {
                return true;
            }
            else
```

```
51              {
52                  return false;
53              } */
54
55              //Pythagorean theorem
56              double distancex = pt.X - X; // x-coor and cirlce center
57              double distancey = pt.Y - Y; // y-coor and circle center
58              double distance = distancex * distancex + distancey * distancey;
59              double radius2 = Radius * Radius;
60
61              if (distance <= radius2)
62              {
63                  return true;
64              }
65              else
66              {
67                  return false;
68              }
69          }
70          public override void SaveTo(StreamWriter writer)
71          {
72              writer.WriteLine("Circle");
73              base.SaveTo(writer);
74              writer.WriteLine(Radius);
75          }
76          public override void LoadFrom(StreamReader reader)
77          {
78              base.LoadFrom(reader);
79              Radius = reader.ReadInteger();
80          }
81      }
82  }
```

```
using SplashKitSDK;

namespace MyGame
{
    public class MyLine : Shape
    {
        private float _endX, _endY;
        public float EndX
        {
            get
            {
                return _endX;
            }
            set
            {
                _endX = value;
            }
        }
        public float EndY
        {
            get
            {
                return _endY;
            }
            set
            {
                _endY = value;
            }
        }
        public MyLine()
        {
            _endX = SplashKit.MouseX() + 50;
            _endY = SplashKit.MouseY() + 50;
            Color = SplashKit.ColorRed();
            X = SplashKit.MouseX();
            Y = SplashKit.MouseY();
        }
        public MyLine(Color color, float startX, float startY, float endX, float
↪   endY)
        {
            Color = color;
            X = startX;
            Y = startY;
            _endX = endX;
            _endY = endY;
        }
        public override void Draw()
        {
            if (Selected) DrawOutline();
            SplashKit.DrawLine(Color, X, Y, _endX, _endY);
            SplashKit.DrawLine(Color, X + 10, Y + 10, _endX, _endY);
        }
        public override void DrawOutline()
```

```
53          {
54              SplashKit.FillCircle(SplashKit.ColorBlack(), X, Y, 2); //draw circle at
    ↪  the start point
55              SplashKit.FillCircle(SplashKit.ColorBlack(), EndX, EndY, 2); //draw
    ↪  circle at the end point
56          }
57          public override bool IsAt(Point2D pt)
58          {
59              // Use SplashKit.PointOnLine to check if the point is on the line
60              Point2D startPoint = new() { X = X, Y = Y }; //define start point of line
61              Point2D endPoint = new() { X = EndX, Y = EndY }; //define end point of
    ↪  line
62              Line line = SplashKit.LineFrom(startPoint, endPoint); // define line
63
64              if (SplashKit.PointOnLine(pt, line))
65              {
66                  return true;
67              }
68              else
69              {
70                  return false;
71              }
72          }
73
74          public override void SaveTo(StreamWriter writer)
75          {
76              writer.WriteLine("Line");
77              base.SaveTo(writer);
78              writer.WriteLine(EndX);
79              writer.WriteLine(EndY);
80          }
81          public override void LoadFrom(StreamReader reader)
82          {
83              base.LoadFrom(reader);
84              EndX = reader.ReadInteger();
85              EndY = reader.ReadInteger();
86          }
87      }
88  }
```


Shape Drawer 3
C:\Users\Tung\Desktop\
Converting null literal or possible null value to non-nullable type.