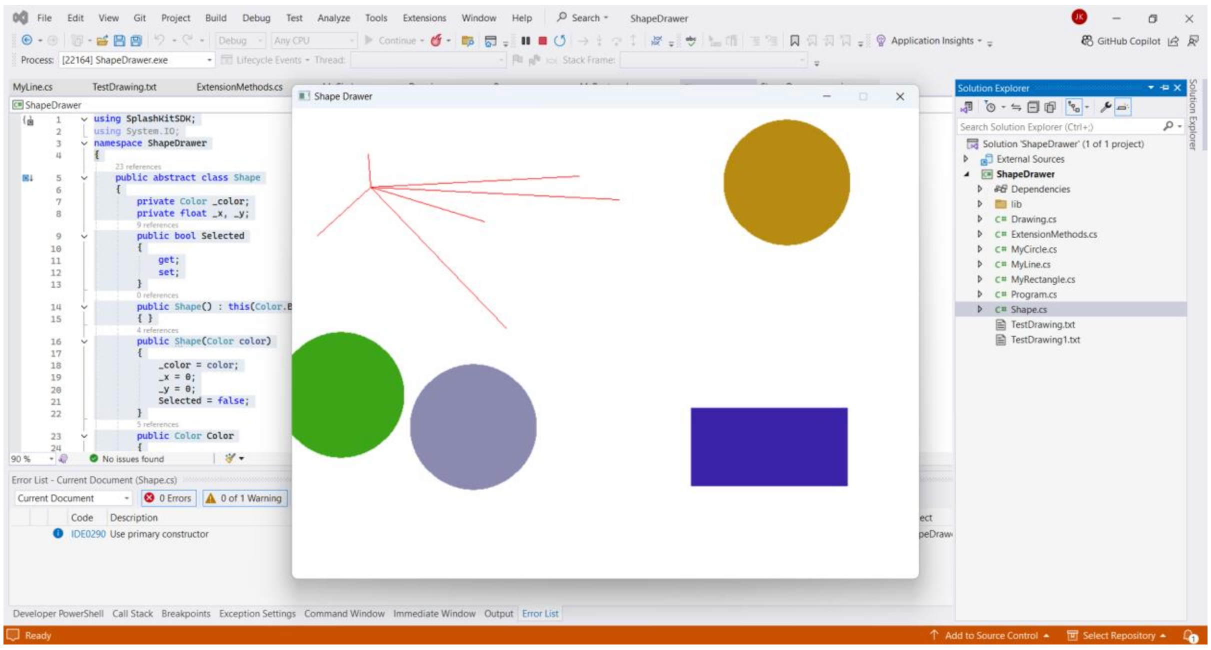


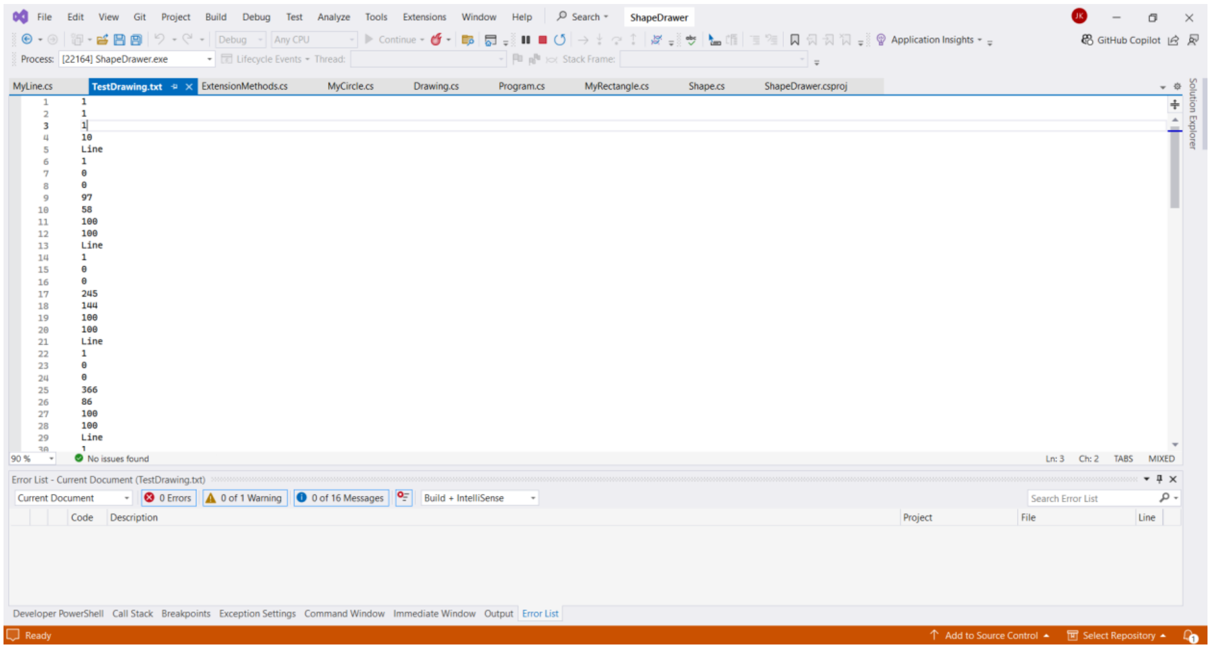
```
using SplashKitSDK;
using System.IO;
namespace ShapeDrawer
{
    public abstract class Shape
    {
        private Color _color;
        private float _x, _y;
        public bool Selected
        {
            get;
            set;
        }
        public Shape() : this(Color.B
        { }
        public Shape(Color color)
        {
            _color = color;
            _x = 0;
            _y = 0;
            Selected = false;
        }
        public Color Color
        {
```

Error List - Current Document (Shape.cs)

0 Errors | 0 of 1 Warning

Code	Description
IDE0290	Use primary constructor

Drawing

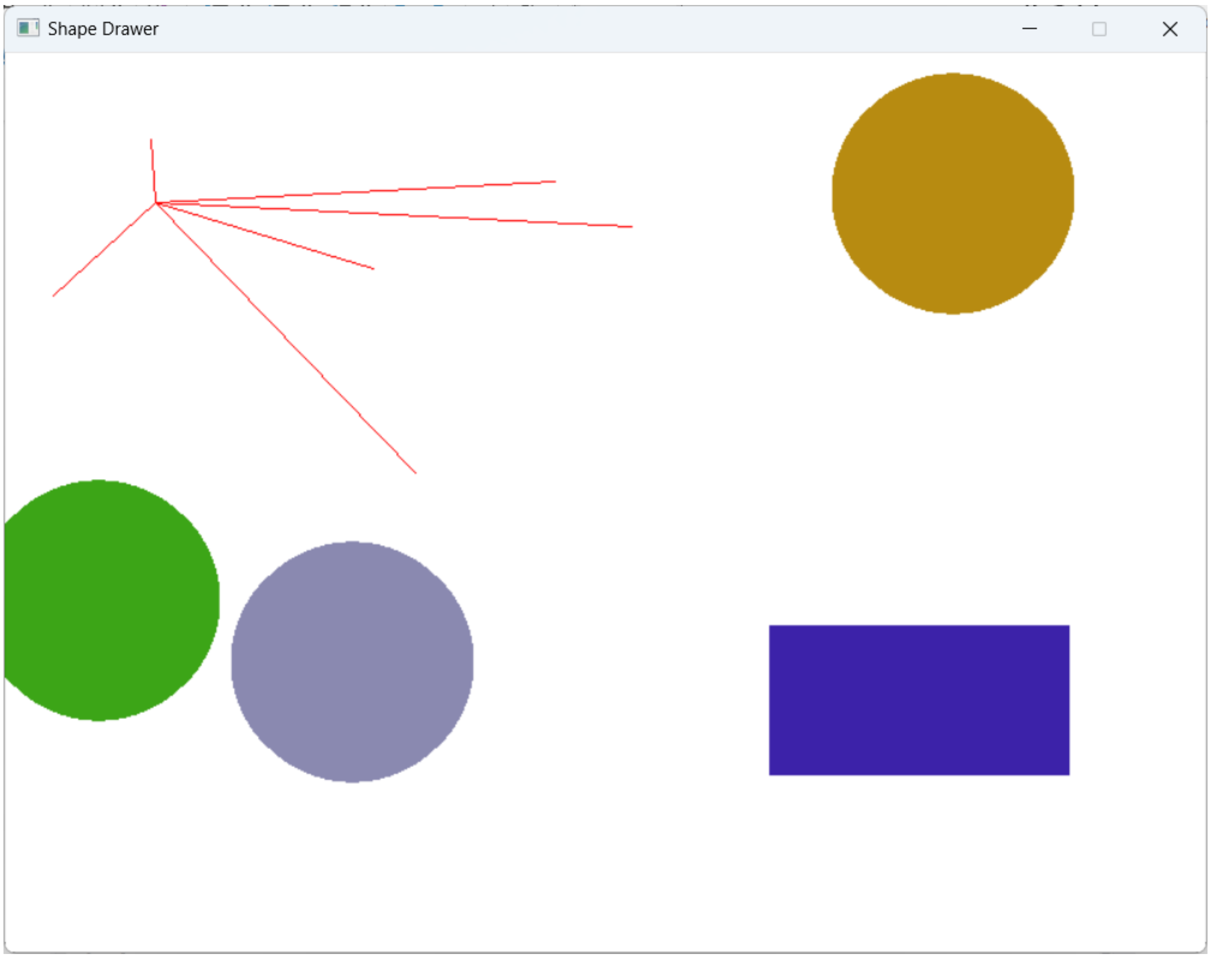


```
1
1
1
10
Line
1
0
0
97
58
100
100
Line
1
0
0
245
144
100
100
Line
1
0
0
366
86
100
100
Line
1
```

Error List - Current Document (TestDrawing.txt)

0 Errors | 0 of 1 Warning | 0 of 16 Messages

Save



load

Extensionmethod.cs:

```
using System;
using System.IO;
using SplashKitSDK;

namespace ShapeDrawer
{
    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);
        }
    }
}
```

Drawing.cs:

```
using System;
using System.Linq;
using System.Collections.Generic;
using SplashKitSDK;
using System.IO;
namespace ShapeDrawer
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;
        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;
        }
        public Color Background
        {
            get { return _background; }
            set { _background = value; }
        }
        public Drawing() : this(Color.White) { }
        public int ShapeCount
        {
            get { return _shapes.Count; }
        }
```

```
public List<Shape> SelectedShapes()
{
    List<Shape> _selectedShapes = new List<Shape>();
    foreach (Shape s in _shapes)
    {
        if (s.Selected)
        {
            _selectedShapes.Add(s);
        }
    }
    return _selectedShapes;
}
public void AddShape(Shape s)
{
    _shapes.Add(s);
}
public void RemoveSelectedShapes()
{
    foreach (Shape s in _shapes.ToList())
    {
        if (s.Selected)
        {
            _shapes.Remove(s);
        }
    }
}
public void Save(string filename)
{

    using (StreamWriter writer = new StreamWriter(filename))
    {
        writer.WriteColor(_background);
        writer.WriteLine(ShapeCount);
        foreach (Shape s in _shapes)
        {
            s.SaveTo(writer);
        }
    }
}
public void Load(string filename)
{

    using (StreamReader reader = new StreamReader(filename))
    {
        Shape s;
        string kind;
        Background = reader.ReadColor();
        int count = reader.ReadInteger();
        _shapes.Clear();
        for (int i = 0; i < count; i++)
        {
            kind = reader.ReadLine();
            switch (kind)
            {
                case "Rectangle":
                    s = new MyRectangle();
                    break;
                case "Circle":
                    s = new MyCircle();
                    break;
                case "Line":
                    s = new MyLine();
                break;
            default:
                throw new InvalidDataException("Unknown shape kind: " + kind);
        }
        s.LoadFrom(reader);
        AddShape(s);
    }
  }
}
public void Draw()
{
  SplashKit.ClearScreen(_background);
  foreach (Shape s in _shapes)
  {
    s.Draw();
  }
}
public void ChangingShapeColor()
{
  foreach (Shape s in _shapes)
  {
    if (s.Selected)
    {
      s.Color = Color.RandomRGB(255);
    }
  }
}
public void SelectedShapeAt(Point2D pt)
{
  foreach (Shape s in _shapes)
  {
```

```
                if (s.IsAt(pt))
                {
                    s.Selected = true;
                }
                else
                {
                    s.Selected = false;
                }
            }
        }
    }
}
```

Mycircle.cs:

```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ShapeDrawer
{
    public class MyCircle : Shape
    {
        private int _radius;
        public MyCircle(Color clr, int radius) : base(clr) //default constructor for circles
        {
            _radius = radius;
        }
        public MyCircle() : this(Color.RandomRGB(255), 80) { } //default constructor for creating circles
        public int Radius
        {
```

```
    get
    {
        return _radius;
    }
    set
    {
        _radius = value;
    }
}
public override void Draw()
{
    if (Selected)
    {
        DrawOutline();
    }
    SplashKit.FillCircle(Color, X, Y, _radius);
}
public override void DrawOutline()
{
    SplashKit.DrawCircle(Color.Black, X, Y, _radius + 1);
}
public override bool IsAt(Point2D pt) //ensure shape is selected
{
    return SplashKit.PointInCircle(pt, SplashKit.CircleAt(X, Y, _radius));
}
public override void SaveTo(StreamWriter writer) //save method
{
    writer.WriteLine("Circle");
    base.SaveTo(writer);
    writer.WriteLine(Radius);
}
```

```
        public override void LoadFrom(StreamReader reader) //load method
        {
            base.LoadFrom(reader);
            Radius = reader.ReadInteger();
        }
    }
}
```

Myline.cs:

```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ShapeDrawer
{
    public class MyLine : Shape
    {
        private float _endX;
        private float _endY;
        public MyLine(Color color, float startX, float startY, float endX, float endY) : base(color)
        {
            X = startX; // Starting point
            Y = startY;
            _endX = endX; // Ending point
            _endY = endY;
        }

        public MyLine() : this(Color.Red, 0, 0, 100, 100) // Default line start and end coordinates
        {
        }
```

```
public float EndX
{
   get { return _endX; }
   set { _endX = value; }
}
public float EndY
{
   get { return _endY; }
   set { _endY = value; }
}
// Override Draw method
public override void Draw()
{
   SplashKit.DrawLine(Color, X, Y, _endX, _endY); // Drawing the shape line
   if (Selected)
   {
      DrawOutline(); // Draw outline if selected
   }
}

public override void DrawOutline()
{
   SplashKit.DrawCircle(Color.Black, X, Y, 5); // making small circles at the starting point
   SplashKit.DrawCircle(Color.Black, _endX, _endY, 5); //making small circles at the ending point
}

public override bool IsAt(Point2D pt)
{
   return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y, _endX, _endY));
}
public override void SaveTo(StreamWriter writer) //save method
```

```
        {
            writer.WriteLine("Line");
            base.SaveTo(writer);
            writer.WriteLine(_endX);
            writer.WriteLine(_endY);
        }
        public override void LoadFrom(StreamReader reader) //load method
        {
            base.LoadFrom(reader);
            _endX = reader.ReadInteger();
            _endY = reader.ReadInteger();
        }
    }
}
```

Myrectangle.cs:

```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ShapeDrawer
{
    public class MyRectangle : Shape
    {
        private int _width;
        private int _height;
        public MyRectangle(Color clr, float x, float y, int width, int height) : base(clr) //constructor for
intialising values
        {
            X = x;
            Y = y;
            Width = width;
            Height = height;
        }
        public MyRectangle() : this(Color.RandomRGB(255), 0, 0, 200, 100) { } //constructor provding
default shapes
        public int Width
        {
            get
            {
                return _width;
            }
            set
            {
                _width = value;
            }
        }
        public int Height
        {
            get
            {
                return _height;
            }
            set
            {
                _height = value;
            }
        }
        public override void Draw()
        {
            if (Selected)
```

```
            {
                DrawOutline();
            }
            SplashKit.FillRectangle(Color, X, Y, Width, Height);
        }
        public override void DrawOutline()
        {
            SplashKit.DrawRectangle(Color.Black, X - 2, Y - 2, _width + 4, _height + 4);
        }
        public override bool IsAt(Point2D p)
        {
            return SplashKit.PointInRectangle(p, SplashKit.RectangleFrom(X, Y, Width, Height));
        }
        public override void SaveTo(StreamWriter writer)
        {
            writer.WriteLine("Rectangle");
            base.SaveTo(writer);
            writer.WriteLine(Width);
            writer.WriteLine(Height);
        }
        public override void LoadFrom(StreamReader reader)
        {
            base.LoadFrom(reader);
            Width = reader.ReadInteger();
            Height = reader.ReadInteger();
        }
    }
}
```

Program.cs:

```
using SplashKitSDK;
namespace ShapeDrawer
```

```
{
  public class Program
  {
    private enum Shapekind
    {
      Rectangle,
      Circle,
      Line
    }
    public static void Main()
    {
      Window window = new Window("Shape Drawer", 800, 600);
      Drawing myDrawing = new Drawing();
      Shapekind KindToAdd = Shapekind.Circle;
      do
      {
        SplashKit.ProcessEvents();
        SplashKit.ClearScreen();
        if (SplashKit.KeyTyped(KeyCode.RKey))
        {
          KindToAdd = Shapekind.Rectangle;
        }
        if (SplashKit.KeyTyped(KeyCode.LKey))
        {
          KindToAdd = Shapekind.Line;
        }
        if (SplashKit.KeyTyped(KeyCode.CKey))
        {
          KindToAdd = Shapekind.Circle;
        }
        if (SplashKit.MouseClicked(MouseButton.LeftButton))
```

```
{
    Shape newShape;
    switch (KindToAdd)
    {
        case Shapekind.Circle:
            newShape = new MyCircle();
            newShape.X = SplashKit.MouseX();
            newShape.Y = SplashKit.MouseY();
            break;
        case Shapekind.Line:
            newShape = new MyLine();
            newShape.X = SplashKit.MouseX();
            newShape.Y = SplashKit.MouseY();
            break;
        default:
            newShape = new MyRectangle();
            newShape.X = SplashKit.MouseX();
            newShape.Y = SplashKit.MouseY();
            break;
    }
    myDrawing.AddShape(newShape);
}
if (SplashKit.KeyTyped(KeyCode.SpaceKey))
{
    myDrawing.Background = Color.RandomRGB(255);
}
if (SplashKit.MouseClicked(MouseButton.RightButton))
{
    myDrawing.SelectedShapeAt(SplashKit.MousePosition());
}
myDrawing.Draw();
```

```
            if (SplashKit.KeyTyped(KeyCode.BackspaceKey))
            {
                myDrawing.RemoveSelectedShapes();
            }
            SplashKit.RefreshScreen();
            //press S to save
            if (SplashKit.KeyDown(KeyCode.SKey))
            {
                myDrawing.Save("C:\\Users\\joshu\\OneDrive\\Desktop\\COS20007 OOP\\ShapeDrawer
- 5.3 demo\\TestDrawing.txt");
            }
            //press O to Load
            if (SplashKit.KeyTyped(KeyCode.OKey))
            {
                try
                {
                    myDrawing.Load("C:\\Users\\joshu\\OneDrive\\Desktop\\COS20007
OOP\\ShapeDrawer - 5.3 demo\\TestDrawing.txt");
                }
                catch (Exception e)
                {
                    Console.Error.WriteLine("Error loading file: {0}", e.Message);
                }
            }
        }
        while (!window.CloseRequested);
        window.Close();


    }
  }
}
```

Shape.cs:

```
using SplashKitSDK;
using System.IO;
namespace ShapeDrawer
{
    public abstract class Shape
    {
        private Color _color;
        private float _x, _y;
        public bool Selected
        {
            get;
            set;
        }
        public Shape() : this(Color.Blue)
        { }
        public Shape(Color color)
        {
            _color = color;
            _x = 0;
            _y = 0;
            Selected = false;
        }
        public Color Color
        {
            get
            {
                return _color;
            }
            set
            {
                _color = value;
            }
        }
        public float X
        {
            get { return _x; }
            set { _x = value; }
        }
        public float Y
        {
            get { return _y; }
            set { _y = value; }
        }
        public abstract void Draw();
        public abstract void DrawOutline();
        public abstract bool IsAt(Point2D p);
        public virtual void SaveTo(StreamWriter writer)
        {
            writer.WriteColor(_color);
            writer.WriteLine(X);
            writer.WriteLine(Y);
        }
        public virtual void LoadFrom(StreamReader reader)
        {
            Color = reader.ReadColor();
            X = reader.ReadInteger();
            Y = reader.ReadInteger();
        }
    }
}
```