

# Selected files

## 6 printable files

Drawing.cs  
ExetnsionMethods.cs  
MyLine.cs  
MyRectangle.cs  
Program.cs  
Shape.cs

### Drawing.cs

```
1  using System;
2  using System.IO;
3  using System.Text;
4  using SplashKitSDK;
5  using System.Collections.Generic;
6
7
8  namespace Drawing_Program__Saving_and_Loading
9  {
10     public class Drawing
11     {
12         private readonly List<Shape> _shapes;
13         private Color _background;
14
15         public Drawing(Color background)
16         {
17             _background = background;
18             _shapes = new List<Shape>();
19         }
20         public Drawing() : this(Color.White)
21         {
22
23         }
24         public Color Background
25         {
26             get { return _background; }
27             set { _background = value; }
28         }
29
30         public void Draw()
31         {
32             SplashKit.ClearScreen(_background);
33             foreach (Shape s in _shapes)
34             {
35                 s.Draw();
36             }
37
38         }
39
40         public void SelectShapesAt(Point2D pt)
41         {
42             foreach (Shape s in _shapes)
43             {
44                 if (s.IsAt(pt))
45                     s.Selected = true;
```

```
46         else
47             s.Selected = false;
48     }
49 }
50
51 public List<Shape> SelectedShapes
52 {
53     get
54     {
55         List<Shape> _selectedShapes = new List<Shape>();
56         foreach(Shape s in _shapes)
57         {
58             if (s.Selected)
59                 _selectedShapes.Add(s);
60         }
61         return _selectedShapes;
62     }
63 }
64
65 public int ShapeCount
66 {
67     get { return _shapes.Count; }
68 }
69
70 public void AddShape(Shape s)
71 {
72     _shapes.Add(s);
73 }
74
75 public void RemoveShape(Shape s)
76 {
77     _shapes.Remove(s);
78 }
79
80 public void Save(string filename)
81 {
82     StreamWriter writer = new StreamWriter(filename);
83     writer.WriteColor(_background);
84     writer.WriteLine(ShapeCount);
85
86     foreach (Shape s in _shapes)
87     {
88         s.SaveTo(writer);
89     }
90
91     writer.Close();
92 }
93
94 public void Load(string filename)
95 {
96     StreamReader reader = new StreamReader(filename);
97     _background = reader.ReadColor();
98     int count = reader.ReadInteger();
99
100     _shapes.Clear();
101     try
102     {
103         Shape s;
104
105         for (int i = 0; i < count; i++)
```

```

106         {
107             string kind = reader.ReadLine();
108
109             switch (kind)
110             {
111                 case "Rectangle":
112                     s = new MyRectangle();
113                     break;
114                 case "Circle":
115                     s = new MyCircle();
116                     break;
117                 case "Line":
118                     s = new MyLine();
119                     break;
120                 default:
121                     throw new InvalidDataException("Unknown shape kind: "
+ kind);
122             }
123             s.LoadFrom(reader);
124             AddShape(s);
125         }
126     }
127     finally
128     {
129         reader.Close();
130     }
131 }
132 }
133 }
134
135

```

## ExetnsionMethods.cs

```

1  using System;
2  using System.IO;
3  using SplashKitSDK;
4
5  namespace Drawing_Program__Saving_and>Loading
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 }

```

```
26 |     }  
27 | }  
28 |  
29 |
```

## MyLine.cs

```
1  using System;  
2  using SplashKitSDK;  
3  
4  namespace Drawing_Program__Saving_and_Loading  
5  {  
6      public class MyLine : Shape  
7      {  
8          private float _endX;  
9          private float _endY;  
10         private int _thickness;  
11  
12         public MyLine(Color color, float endX, float endY, int thickness) :  
base(color)  
13         {  
14             _endX = endX;  
15             _endY = endY;  
16             _thickness = thickness;  
17         }  
18  
19         public MyLine() : this(Color.Black, 0, 0, 0)  
20         {  
21         }  
22     }  
23  
24     public float EndX  
25     {  
26         get { return _endX; }  
27         set { _endX = value; }  
28     }  
29  
30     public float EndY  
31     {  
32         get { return _endY; }  
33         set { _endY = value; }  
34     }  
35  
36     public int Thickness  
37     {  
38         get { return _thickness; }  
39         set { _thickness = value; }  
40     }  
41  
42     public override void Draw()  
43     {  
44         if (Selected)  
45             DrawOutline();  
46         SplashKit.DrawLine(Color, X, Y, _endX, _endY);  
47     }  
48  
49     public override void DrawOutline()  
50     {  
51
```

```
52     }
53
54     public override bool IsAt(Point2D pt)
55     {
56
57         float minX = Math.Min(X, _endX) - _thickness / 2;
58         float minY = Math.Min(Y, _endY) - _thickness / 2;
59         float maxX = Math.Max(X, _endX) + _thickness / 2;
60         float maxY = Math.Max(Y, _endY) + _thickness / 2;
61
62
63         return pt.X >= minX && pt.X <= maxX && pt.Y >= minY && pt.Y <= maxY;
64     }
65
66     public override void SaveTo(StreamWriter writer)
67     {
68         writer.WriteLine("Line");
69         base.SaveTo(writer);
70         writer.WriteLine(EndX);
71         writer.WriteLine(EndY);
72     }
73
74     public override void LoadFrom(StreamReader reader)
75     {
76         base.LoadFrom(reader);
77         EndX = reader.ReadInteger();
78         EndY = reader.ReadInteger();
79     }
80 }
81 }
82
83
84
```

## MyRectangle.cs

```
1  using System;
2  using SplashScreen;
3
4  namespace Drawing_Program__Saving_and>Loading
5  {
6      public class MyRectangle : Shape
7      {
8          private int _width;
9          private int _height;
10
11      public MyRectangle(Color color, float x, float y, int width, int height)
12      : base(color)
13      {
14          X = x;
15          Y = y;
16          _width = width;
17          _height = height;
18      }
19
20      public MyRectangle() : this(Color.Green, 0, 0, 100, 100)
21      {
22      }
23
24
```

```

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 Draw()
37     {
38         SplashKit.FillRectangle(Color, X, Y, Width, Height);
39     }
40
41     public override void DrawOutline()
42     {
43         SplashKit.FillRectangle(Color.Black, X, Y, Width + 2, Height + 2);
44     }
45
46     public override bool IsAt(Point2D pt)
47     {
48         return pt.X >= X && pt.X <= X + Width && pt.Y >= Y && pt.Y <= Y +
Height;
49     }
50
51     public override void SaveTo(StreamWriter writer)
52     {
53         writer.WriteLine("Rectangle");
54         base.SaveTo(writer);
55         writer.WriteLine(Width);
56         writer.WriteLine(Height);
57     }
58
59     public override void LoadFrom(StreamReader reader)
60     {
61         base.LoadFrom(reader);
62         Width = reader.ReadInteger();
63         Height = reader.ReadInteger();
64     }
65 }
66 }
67
68

```

### Program.cs

```

1  using System;
2  using SplashKitSDK;
3
4  namespace Drawing_Program__Saving_and_Loading
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("Multiple Shape", 800, 600);
18         Drawing myDrawing = new Drawing();
19
20         ShapeKind kindToAdd = ShapeKind.Circle;
21         do
22         {
23             SplashKit.ProcessEvents();
24
25             SplashKit.ClearScreen();
26
27             if (SplashKit.MouseClicked(MouseButton.LeftButton))
28             {
29                 Shape newShape;
30                 switch (kindToAdd)
31                 {
32                     case ShapeKind.Circle:
33                         newShape = new MyCircle();
34                         newShape.X = SplashKit.MouseX();
35                         newShape.Y = SplashKit.MouseY();
36                         myDrawing.AddShape(newShape);
37                         break;
38
39                     case ShapeKind.Rectangle:
40                         newShape = new MyRectangle();
41                         newShape.X = SplashKit.MouseX();
42                         newShape.Y = SplashKit.MouseY();
43                         myDrawing.AddShape(newShape);
44                         break;
45
46                     case ShapeKind.Line:
47                         newShape = new MyLine();
48                         newShape.X = SplashKit.MouseX();
49                         newShape.Y = SplashKit.MouseY();
50                         myDrawing.AddShape(newShape);
51                         break;
52                 }
53             }
54
55             if (SplashKit.KeyDown(KeyCode.RKey))
56             {
57                 kindToAdd = ShapeKind.Rectangle; // Press 'R' to draw
58                 rectangles
59             }
60
61             if (SplashKit.KeyDown(KeyCode.CKey))
62             {
63                 kindToAdd = ShapeKind.Circle; // Press 'C' to draw circles
64             }
65
66             if (SplashKit.KeyDown(KeyCode.LKey))
67             {
68                 kindToAdd = ShapeKind.Line; // Press 'L' to draw lines
69             }
69         }
70     }
```

```

69
70         if (SplashKit.KeyTyped(KeyCode.SpaceKey))
71         {
72             myDrawing.Background = SplashKit.RandomRGBColor(255);
73         }
74
75         if (SplashKit.MouseClicked(MouseButton.RightButton))
76         {
77             myDrawing.SelectShapesAt(SplashKit.MousePosition());
78         }
79
80         if (SplashKit.KeyDown(KeyCode.DeleteKey) ||
81 SplashKit.KeyDown(KeyCode.BackspaceKey))
82         {
83             foreach (Shape s in myDrawing.SelectedShapes)
84             {
85                 myDrawing.RemoveShape(s);
86             }
87
88             if (SplashKit.KeyDown(KeyCode.SKey))
89             {
90                 string filePath = "/Users/thuanduc/Documents/thuan's
91 folder/work/COS20007/week 5/5.3/TextDrawing.txt";
92                 myDrawing.Save(filePath); // press "S" to save the Drawing
93             }
94             if (SplashKit.KeyDown(KeyCode.OKey))
95             {
96                 try
97                 {
98                     string filePath = "/Users/thuanduc/Documents/thuan's
99 folder/work/COS20007/week 5/5.3/TextDrawing.txt";
100                     myDrawing.Load(filePath); // press "O" to loading the
101 test Drawing file.
102                 } catch (Exception e)
103                 {
104                     Console.Error.WriteLine("Error loading file: {0}",
105 e.Message);
106                 }
107             }
108             myDrawing.Draw();
109
110             SplashKit.RefreshScreen();
111         } while (!window.CloseRequested);
112     }
113 }

```

## Shape.cs

```

1 using System;
2 using System.IO;
3 using SplashKitSDK;
4
5 namespace Drawing_Program__Saving_and>Loading
6 {
7     public abstract class Shape
8     {
9         private Color _color;

```



```
10 private float _x;
11 private float _y;
12 private bool _selected;
13
14 public Shape(Color color)
15 {
16     _color = color;
17 }
18
19 public Color Color
20 {
21     get { return _color; }
22     set { _color = value; }
23 }
24
25 public float X
26 {
27     get { return _x; }
28     set { _x = value; }
29 }
30
31 public float Y
32 {
33     get { return _y; }
34     set { _y = value; }
35 }
36
37 public bool Selected
38 {
39     get { return _selected; }
40     set { _selected = value; }
41 }
42
43 public abstract void Draw();
44 public abstract void DrawOutline();
45 public abstract bool IsAt(Point2D pt);
46
47 public virtual void SaveTo(StreamWriter writer)
48 {
49     writer.WriteColor(_color);
50     writer.WriteLine(_x);
51     writer.WriteLine(_y);
52 }
53
54 public virtual void LoadFrom(StreamReader reader)
55 {
56     Color = reader.ReadColor();
57     X = reader.ReadInteger();
58     Y = reader.ReadInteger();
59 }
60
61 }
62 }
63
64
```