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

namespace ShapeDrawing
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;
        public List<Shape> Shapes
        {
            get { return _shapes; }
        }

        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 int ShapeCount
        {
            get
            {
                return _shapes.Count;
            }
        }
        //Change method SelectShapesAt to select multiple shapes to delete at once
        public void SelectShapesAt(Point2D pt)
        {
            foreach (Shape s in _shapes)
            {
                if (s.IsAt(pt))
                {
                    s.Selected = !s.Selected; // Toggle selection
                }
            }
        }
        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> result = new List<Shape>();
                foreach (Shape s in _shapes)
                {
                    if (s.Selected)
                    {
                        result.Add(s);
                    }
                }
                return result;
            }
        }
        public void AddShape(Shape s)
        {
            _shapes.Add(s);
        }
        public void RemoveShape(Shape s)
        {
            _shapes.Remove(s);
        }
        public void Draw()
        {
            SplashKit.ClearScreen(_background);
            foreach (Shape s in _shapes)
            {
                s.Draw();
            }
        }
        //Step 4 of task 15.3C
        //Save file
        public void Save(string filename)
        {
            StreamWriter writer = new(filename);

            try
            {
                writer.WriteColor(Background);
                writer.WriteLine(ShapeCount);

                foreach (Shape shape in _shapes)
                {
                    shape.SaveTo(writer);
                }
            }
            finally
            {
                writer.Close();
            }
        }
        //Load file (Step 12 of task 15.3C)
        public void Load(string filename)
        {
            Shape s;
            int count;
            string kind;

            StreamReader reader = new(filename);
            try
            {
                Background = reader.ReadColor();
                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);
                }
            }
            finally
            {
                reader.Close();
            }
        }
    }
}
```