```
using System.Collections.Generic;
using SplashKitSDK;

namespace ShapeDrawer
{
    internal class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;

        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;
        }

        public Drawing() : this(Color.White) { }

        public Color Background
        {
            get { return _background; }
            set { _background = value; }
        }

        public int ShapeCount
        {
            get { return _shapes.Count; }
        }

        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();
            }
        }

        public void SelectShapesAt(Point2D pt)
        {
            foreach (Shape s in _shapes)
            {
                s.Selected = s.IsAt(pt);
            }
        }

        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> result = new List<Shape>();
                foreach (Shape s in _shapes)
                {
                    if (s.Selected)
                    {
                        result.Add(s);
                    }
                }
                return result;
            }
        }
    }
}

```

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

namespace ShapeDrawer
{
    public class MyCircle : Shape
    {
        private int _radius;
        public int Radius
        {
            get { return _radius; }
            set { _radius = value; }
        }
        public MyCircle() : this(Color.Chocolate,184)
        {

        }

        public MyCircle(Color color, int _radius) : base(color)
        {
            Radius = _radius;
        }
        public override void Draw()
        {
            if (Selected)
           {
                DrawOutline(5);
            }
            SplashKit.FillCircle(Color, X, Y, _radius);
        }
        public override void DrawOutline(int extra)
        {
            {
                SplashKit.DrawCircle(Color.Black,X, Y, _radius + 2);
            }
        }

        public override bool IsAt(Point2D pt)
        {
            if (pt.X >= _radius && pt.Y >= _radius )
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}

```

```
using System;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class MyLine : Shape
    {
        private float _endX;
        private float _endY;

        public MyLine() : this (Color.Chocolate,5,5,10,10)
        {

        }

        public MyLine(Color color, float startX, float startY, float endX, 
          float endY) : base(color)
        {
            X = startX;
            Y = startY;
            _endX = endX;
            _endY = endY;
        }

        public float EndX
        {
            get { return _endX; }
            set { _endX = value; }
        }

        public float EndY
        {
            get { return _endY; }
            set { _endY = value; }
        }

        public override void Draw()
        {
            SplashKit.DrawLine(Color, X, Y, _endX, _endY);
            if (Selected) DrawOutline(2);
        }

        public override void DrawOutline(int extra)
        {
            SplashKit.FillCircle(Color.Black, X, Y, extra);
            SplashKit.FillCircle(Color.Black, _endX, _endY, extra);
        }

        public override bool IsAt(Point2D pt)
        {
            return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y, _endX, _endY));
        }
    }
}

```

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

namespace ShapeDrawer
{
    public class MyRectangle : Shape
    {
        private int _width;
        private int _height;
        public int Width
        {
            get { return _width; }
            set { _width = value; }
        }
        public int Height
        {
            get { return _height; }
            set { _height = value; }
        }

        public MyRectangle(Color color,float x,float y,int width,int
          height) : base(color)
        {
            _width = width;
            _height = height;
            X = x;
            Y = y;
        }
        public MyRectangle() : this(Color.Chocolate, 0.0f, 0.0f, 184, 184)
        {

        }
        public override void Draw()
        {
            SplashKit.FillRectangle(Color, X, Y, _width, _height);
            if (Selected)
            {
                int extra = 9;
                DrawOutline(extra);
            }
        }


        public override void DrawOutline(int extra)
        {
            SplashKit.DrawRectangle(Color.Black, X - extra, Y - extra,
              Width + 2 * extra, Height + 2 * extra);
        }

        public override bool IsAt(Point2D pt)
        {
            if (pt.X >= X && pt.Y >= Y && pt.X <= X + _width && pt.Y <=
            Y + _height)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }

}


```

```csharp
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Circle,
            Line
        }

        public static void Main()
        {
            Drawing myDrawing = new Drawing();
            Window window = new Window("Shape Drawer", 800, 600);
            ShapeKind kindToAdd = ShapeKind.Circle;


            string studentID = "104762184";
            int lastDigit = int.Parse(studentID[^1].ToString());
            int numberOfLinesToAdd = lastDigit == 0 ? 5 : lastDigit;
            int linesAdded = 0;

            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();

                if (SplashKit.KeyDown(KeyCode.RKey)) kindToAdd = ShapeKind.Rectangle;
                if (SplashKit.KeyDown(KeyCode.CKey)) kindToAdd = ShapeKind.Circle;
                if (SplashKit.KeyDown(KeyCode.LKey)) kindToAdd = ShapeKind.Line;

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape newShape;

                    switch (kindToAdd)
                    {
                        case ShapeKind.Circle:
                            newShape = new MyCircle();
                            break;
                        case ShapeKind.Line:

                            if (linesAdded < numberOfLinesToAdd)
                            {
                                newShape = new MyLine();
                                newShape.X = SplashKit.MouseX();
                                newShape.Y = SplashKit.MouseY();
                                myDrawing.AddShape(newShape);
                                linesAdded++;
                            }
                            continue;
                        default:
                            newShape = new MyRectangle();
                            break;
                    }

                    newShape.X = SplashKit.MouseX();
                    newShape.Y = SplashKit.MouseY();
                    myDrawing.AddShape(newShape);
                }


                if (SplashKit.MouseClicked(MouseButton.RightButton) ||
                    (kindToAdd != ShapeKind.Line && linesAdded > 0))
                {
                    linesAdded = 0;
                }

                if (SplashKit.KeyDown(KeyCode.SpaceKey))
                {
                    myDrawing.Background = Color.Random();
                }

                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    myDrawing.SelectShapesAt(SplashKit.MousePosition());
                }

                if (SplashKit.KeyDown(KeyCode.DeleteKey) ||
                  SplashKit.KeyDown(KeyCode.BackspaceKey))
                {
                    foreach (Shape s in myDrawing.SelectedShapes)
                    {
                        myDrawing.RemoveShape(s);
                    }
                }

                myDrawing.Draw();
                SplashKit.RefreshScreen();
            } while (!window.CloseRequested);
        }
    }
```

```
}

```

```
using SplashKitSDK;


namespace ShapeDrawer
{
    public abstract class Shape
    {

        private Color _color;
        private float _x;
        private float _y;
        private bool _selected;


        public Shape(Color color)
        {
            _x = 0.0f;
            _y = 0.0f;
            _color = Color.Chocolate;
        }
        public Shape() : this(Color.Yellow)
        {

        }


        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 bool IsAt(Point2D pt);



        public bool Selected
        {
            get { return _selected; }
            set { _selected = value; }

        }
        public abstract void DrawOutline(int extra);






    }
}


```