COS 20007
Task 4.1

Duc Thuan Tran
104330455

I. Code

1. Drawing.cs

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

 namespace MultipleShape
{
   public class Drawing
   {
      private readonly List<Shape> _shapes;
      private Color _background;

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

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

      public void Draw()
      {
         SplashKit.ClearScreen(_background);
         foreach (Shape s in _shapes)
         {
            s.Draw();
         }

      }

      public void SelectShapesAt(Point2D pt)
      {
```

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

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

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

        public void AddShape(Shape s)
        {
            _shapes.Add(s);
        }

        public void RemoveShape(Shape s)
        {
            _shapes.Remove(s);
        }

    }
}
```

2. Shape.cs

```
using System;
using SplashKitSDK;

namespace MultipleShape
```

```
{
    public abstract class Shape
    {
        private Color _color;
        private float _x;
        private float _y;
        private bool _selected;

        public Shape(Color color)
        {
            _color = color;
        }

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

        public abstract void Draw();
        public abstract void DrawOutline();
        public abstract bool IsAt(Point2D pt);
    }
}
```

3. MyCircle.cs

```
using System;
using SplashKitSDK;
```

```
namespace MultipleShape{
        public class MyCircle : Shape
        {
    private int _radius;

    public MyCircle(Color color, float x, float y, int radius) : base(color)
    {
      X = x;
      Y = y;
      _radius = radius;
                }

    public MyCircle() : this(Color.Blue, 0, 0, 50)
    {

    }

    public int Radius
    {
      get { return _radius; }
      set { _radius = value; }
    }

    public override void Draw()
    {
      SplashKit.FillCircle(Color, X, Y, _radius);
    }

    public override void DrawOutline()
    {

      SplashKit.FillCircle(Color.Black, X, Y, _radius + 2);
    }

    public override bool IsAt(Point2D pt)
    {
      double dX = pt.X - X;
      double dY = pt.Y - Y;
      double distance = System.Math.Sqrt(dX * dX + dY * dY);

      return distance <= _radius;
    }
  }
}
```

4. MyRectangle.cs

```
using System;
using SplashKitSDK;

namespace MultipleShape
{
        public class MyRectangle : Shape
        {
                private int _width;
                private int _height;

                public MyRectangle(Color color, float x, float y, int width, int height) :
base(color)
                {
          X = x;
          Y = y;
          _width = width;
          _height = height;
      }

      public MyRectangle() : this(Color.Green, 0, 0, 100, 100)
      {

      }

      public int Width
      {
        get { return _width; }
        set { _width = value; }
      }

      public int Height
                {
        get { return _height; }
        set { _height = value; }
      }

      public override void Draw()
      {
        SplashKit.FillRectangle(Color, X, Y, Width, Height);
      }

      public override void DrawOutline()
      {
        SplashKit.FillRectangle(Color.Black, X, Y, Width + 2, Height + 2);
      }
```

```
        public override bool IsAt(Point2D pt)
        {
            return pt.X >= X && pt.X <= X + Width && pt.Y >= Y && pt.Y <= Y + Height;
        }
    }
}
```

5. MyLine.cs

```
using System;
using SplashKitSDK;

namespace MultipleShape
{
    public class MyLine : Shape
    {
        private float _endX;
        private float _endY;
        private int _thickness;

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

        public MyLine() : this(Color.Black, 0, 0, 0, 0, 2)
        {

        }

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

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

```
        public int Thickness
        {
            get { return _thickness; }
            set { _thickness = value; }
        }

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

        public override void DrawOutline()
        {

            float radius = _thickness * 5f;
            SplashKit.FillCircle(Color.Red, X, Y, radius);
            SplashKit.FillCircle(Color.Red, _endX, _endY, radius);
        }

        public override bool IsAt(Point2D pt)
        {

            float minX = Math.Min(X, _endX) - _thickness / 2;
            float minY = Math.Min(Y, _endY) - _thickness / 2;
            float maxX = Math.Max(X, _endX) + _thickness / 2;
            float maxY = Math.Max(Y, _endY) + _thickness / 2;

            return pt.X >= minX && pt.X <= maxX && pt.Y >= minY && pt.Y <= maxY;
        }
    }
}
```

6. Program.cs

```
using System;
using SplashKitSDK;

namespace DrawingShape
{
    public class Program
    {
        public static void Main()
```

```
    {
        Window window = new Window("Multiple Shape", 800, 600);
        Drawing myDrawing = new Drawing();
        do
        {
            SplashKit.ProcessEvents();

            SplashKit.ClearScreen();

            if (SplashKit.MouseClicked(MouseButton.LeftButton))
            {
                Shape s = new Shape();
                s.X = SplashKit.MouseX();
                s.Y = SplashKit.MouseY();
                myDrawing.AddShape(s);
            }

            if (SplashKit.KeyTyped(KeyCode.SpaceKey))
            {
                myDrawing.Background = SplashKit.RandomRGBColor(255);
            }


            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);
    }
  }
}
```

II. Image

1. Program’s output