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

namespace ShapeDrawer
{
    public class Drawing
    {
        //Step 3: Set variables
        private readonly List<Shape> _shapes;
        private Color _background; //Step 4
        private Shape _activeShape; // To store the last clicked shape
        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background; //Step 4
        }
        //Step 6:
        public Drawing() : this(Color.White)
        {

        }
        //Step 4
        public Color Background
        {
            get { return _background; }
            set { _background = value; }
        }
        //Step 7
        public int ShapeCount
        {
            get { return _shapes.Count; }
        }
        //Creativity
        public Shape ActiveShape
        {
            get { return _activeShape; }
        }
        //Step 8
        public void AddShape(Shape shape)
        {
            _shapes.Add(shape);
        }
        //Step 9:
        public void RemoveShape(Shape shape)
        {
```

```
            _shapes.Remove(shape);
        }
        //Step 11
        public void Draw()
        {
            SplashKit.ClearScreen(_background);
            foreach (Shape shape in _shapes)
            {
                shape.Draw();
            }
        }
        //Step 17
        //When right-clicked on the second to display border, the border of
          the first shape will not disappear.
        public void SelectShapesAt(Point2D pt)
        {
            bool ClickedOnAShape = false;
            _activeShape = null; // *** NEW: Reset active shape at the start
              of selection process ***
            foreach (Shape s in _shapes.Reverse<Shape>())
            {
                if (s.IsAt(pt))
                {
                    s.Selected = !s.Selected;
                    ClickedOnAShape = true;
                    if (s.Selected)
                    {
                        _activeShape = s;
                    }
                }
            }
            if (!ClickedOnAShape)
            {
                foreach (Shape s in _shapes)
                {
                    s.Selected = false;
                }
                _activeShape = null;
            }
        }
        //Step 18
        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> result = new List<Shape>();
                foreach (Shape s in _shapes)
                {
                    if (s.Selected)
```

```
                    {
                        result.Add(s);
                    }
                }
                return result;
            }
        }
    }
}
```