OOP: Task 3.3P

Program.cs:

```
using System;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Program
    {
        public static void Main()
        {
            Window window = new Window("Shape Drawer - Tien", 800, 600);
            Drawing myDrawing = new Drawing();
            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();
                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape myShape = new Shape();
                    myShape.X = SplashKit.MouseX();
                    myShape.Y = SplashKit.MouseY();
                    myDrawing.AddShape(myShape);
                }
                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    Point2D selected = SplashKit.MousePosition();
                    myDrawing.SelectShapesAt(selected);
                }
                if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                {
                    myDrawing.Background = SplashKit.RandomColor();
                }
                if (SplashKit.KeyTyped(KeyCode.DeleteKey) || SplashKit.KeyTyped(KeyCode.BackspaceKey))
                {
```

```
                foreach (Shape s in myDrawing.SelectedShapes)
                {
                    myDrawing.RemoveShape(s);
                }
            }
            myDrawing.Draw();
            SplashKit.RefreshScreen();
        } while (!window.CloseRequested);
    }
  }
}
```

Shape.cs:

```
using System;
using SplashKitSDK;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ShapeDrawer
{
    public class Shape
    {
        private Color _color;
        private float _x, _y;
        private int _width, _height;
        private bool _selected;
        public Shape()
        {
            _color = Color.Chocolate;
            _x = 0.0f;
            _y = 0.0f;
            _width = 148;
            _height = 148;
        }
```

```
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 int Width
```

```
    {
        get
        {
            return _width;
        }
        set
        {
            _width = value;
        }
    }

    public int Height
    {
        get
        {
            return _height;
        }
        set
        {
            _height = value;
        }
    }
    public bool Selected
    {
        get
        {
            return _selected;
        }
        set
        {
            _selected = value;
        }
    }
    public void Draw()
    {
        SplashKit.FillRectangle(_color, _x, _y, _width, _height);
        if (_selected)
```

```
        {
            DrawOutline();
        }
    }

    public bool IsAt(Point2D pt)
    {
        if ((pt.X >= _x) && (pt.X <= _x + _width) && (pt.Y >= _y) && (pt.Y <= _y + _height))
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    //my ID: 104700948 => last digit: 8 => value = 13
    public void DrawOutline()
    {
        int value = 13;
        SplashKit.DrawRectangle(Color.Black, _x - value, _y - value , _width + value*2, _height + value*2);
    }
  }
}
```

Drawing.cs:

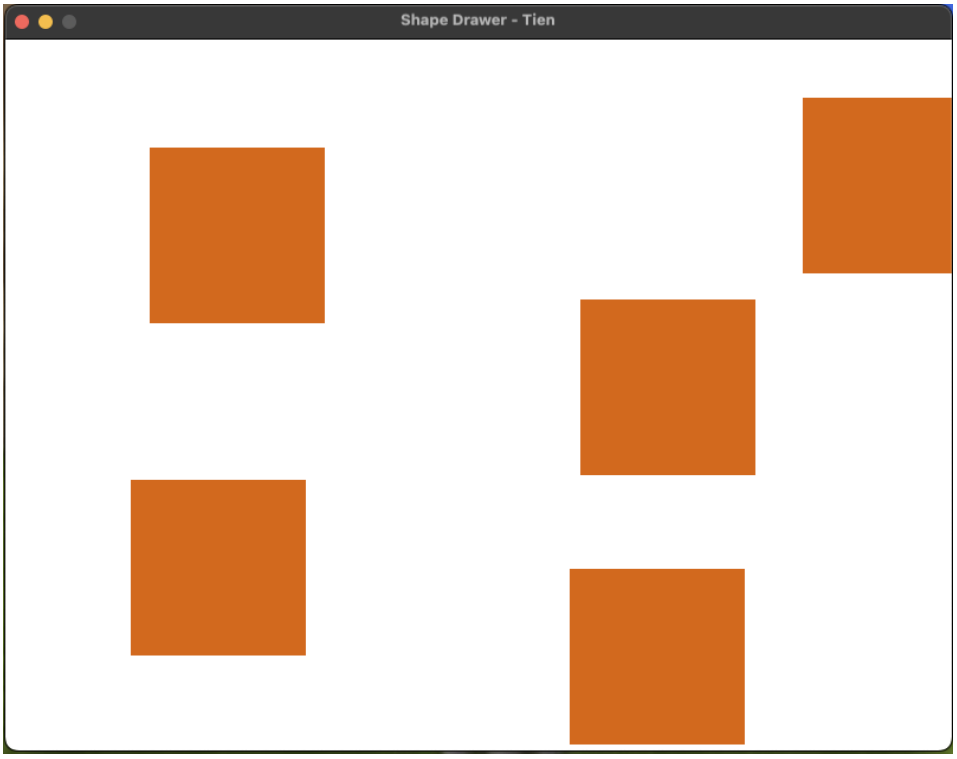
```
using System;
using SplashKitSDK;
namespace ShapeDrawer
{
  public class Drawing
  {
    private readonly List<Shape> _shapes;
    private Color _background;
```

```
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;
    }
}
public void AddShape(Shape s)
{
    _shapes.Add(s);
}
public void RemoveShape(Shape s)
{
    _ = _shapes.Remove(s);
}
public void Draw()
```

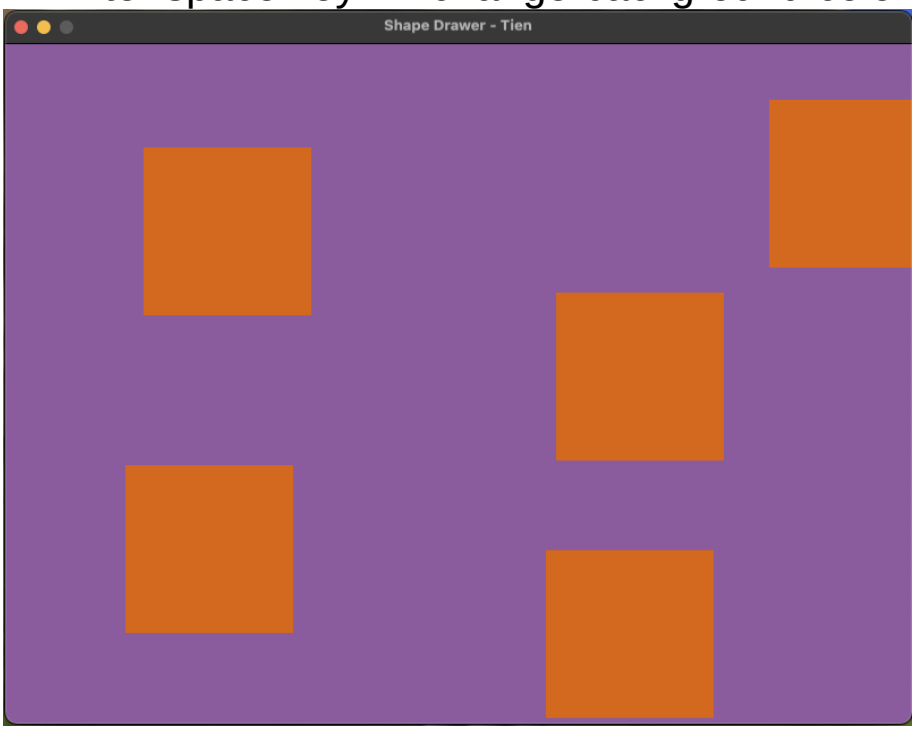
```
    {
        SplashKit.ClearScreen(_background);
        for (int i=0; i<_shapes.Count; i++)
        {
            _shapes[i].Draw();
        }
    }
    public void SelectShapesAt(Point2D pt)
    {
        foreach (Shape s in _shapes)
        {
            s.Selected = s.IsAt(pt);
        }
    }
    public List<Shape> SelectedShapes
    {
        get
        {
            List<Shape> result = new List<Shape>();

            foreach (Shape s in _shapes)
            {
                if (s.Selected)
                {
                    result.Add(s);
                }
            }return result;
        }
    }
  }
}
```

Output:

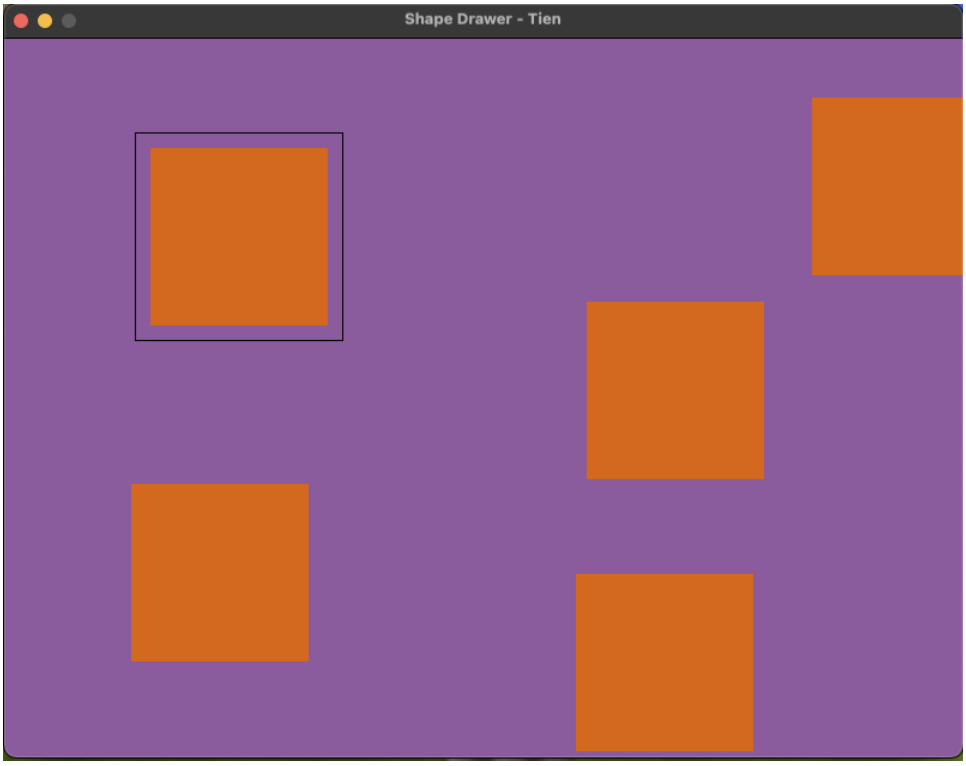
1. Left click to add shapes on different positions



2. Enter space key => change background color



3. Right click on a specific shape => occur another black rectangle



4. Press the delete key => that shape will be removed