```
 1  using System.Collections.Generic;
 2  using SplashKitSDK;
 3
 4  namespace ShapeDrawer
 5  {
 6      internal class Drawing
 7      {
 8          private readonly List<Shape> _shapes;
 9          private Color _background;
10
11          public Drawing(Color background)
12          {
13              _shapes = new List<Shape>();
14              _background = background;
15          }
16
17          public Drawing() : this(Color.White) { }
18
19          public Color Background
20          {
21              get { return _background; }
22              set { _background = value; }
23          }
24
25          public int ShapeCount
26          {
27              get { return _shapes.Count; }
28          }
29
30          public void AddShape(Shape s)
31          {
32              _shapes.Add(s);
33          }
34
35          public void RemoveShape(Shape s)
36          {
37              _shapes.Remove(s);
38          }
39
40          public void Draw()
41          {
42              SplashKit.ClearScreen(_background);
43              foreach (Shape s in _shapes)
44              {
45                  s.Draw();
46              }
47          }
48
49          public void SelectShapesAt(Point2D pt)
```

```
50          {
51              foreach (Shape s in _shapes)
52              {
53                  s.Selected = s.IsAt(pt);
54              }
55          }
56
57          public List<Shape> SelectedShapes
58          {
59              get
60              {
61                  List<Shape> result = new List<Shape>();
62                  foreach (Shape s in _shapes)
63                  {
64                      if (s.Selected)
65                      {
66                          result.Add(s);
67                      }
68                  }
69                  return result;
70              }
71          }
72      }
73  }
74
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6  using SplashKitSDK;
 7
 8  namespace ShapeDrawer
 9  {
10      public class MyCircle : Shape
11      {
12          private int _radius;
13          public int Radius
14          {
15              get { return _radius; }
16              set { _radius = value; }
17          }
18          public MyCircle() : this(Color.Chocolate,184)
19          {
20
21          }
22
23          public MyCircle(Color color, int _radius) : base(color)
24          {
25              Radius = _radius;
26          }
27          public override void Draw()
28          {
29              if (Selected)
30             {
31                  DrawOutline(5);
32              }
33              SplashKit.FillCircle(Color, X, Y, _radius);
34          }
35          public override void DrawOutline(int extra)
36          {
37              {
38                  SplashKit.DrawCircle(Color.Black,X, Y, _radius + 2);
39              }
40          }
41
42          public override bool IsAt(Point2D pt)
43          {
44              if (pt.X >= _radius && pt.Y >= _radius )
45              {
46                  return true;
47              }
48              else
49              {
```

```
50                  return false;
51              }
52          }
53      }
54  }
55
```

```
1  using System;
2  using SplashKitSDK;
3
4  namespace ShapeDrawer
5  {
6      public class MyLine : Shape
7      {
8          private float _endX;
9          private float _endY;
10
11         public MyLine() : this (Color.Chocolate,5,5,10,10)
12         {
13
14         }
15
16         public MyLine(Color color, float startX, float startY, float endX,
             float endY) : base(color)
17         {
18             X = startX;
19             Y = startY;
20             _endX = endX;
21             _endY = endY;
22         }
23
24         public float EndX
25         {
26             get { return _endX; }
27             set { _endX = value; }
28         }
29
30         public float EndY
31         {
32             get { return _endY; }
33             set { _endY = value; }
34         }
35
36         public override void Draw()
37         {
38             SplashKit.DrawLine(Color, X, Y, _endX, _endY);
39             if (Selected) DrawOutline(2);
40         }
41
42         public override void DrawOutline(int extra)
43         {
44             SplashKit.FillCircle(Color.Black, X, Y, extra);
45             SplashKit.FillCircle(Color.Black, _endX, _endY, extra);
46         }
47
48         public override bool IsAt(Point2D pt)
```

```
49          {
50              return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y,
                  _endX, _endY));
51          }
52      }
53  }
54
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  //using System.Drawing;
 4  using System.Linq;
 5  using System.Text;
 6  using System.Threading.Tasks;
 7  using SplashKitSDK;
 8
 9  namespace ShapeDrawer
10  {
11      public class MyRectangle : Shape
12      {
13          private int _width;
14          private int _height;
15          public int Width
16          {
17              get { return _width; }
18              set { _width = value; }
19          }
20          public int Height
21          {
22              get { return _height; }
23              set { _height = value; }
24          }
25
26          public MyRectangle(Color color,float x,float y,int width,int
              height) : base(color)
27          {
28              _width = width;
29              _height = height;
30              X = x;
31              Y = y;
32          }
33          public MyRectangle() : this(Color.Chocolate, 0.0f, 0.0f, 184, 184)
34          {
35
36          }
37          public override void Draw()
38          {
39              SplashKit.FillRectangle(Color, X, Y, _width, _height);
40              if (Selected)
41              {
42                  int extra = 9;
43                  DrawOutline(extra);
44              }
45          }
46
47
48          public override void DrawOutline(int extra)
```

```
49          {
50              SplashKit.DrawRectangle(Color.Black, X - extra, Y - extra,
                  Width + 2 * extra, Height + 2 * extra);
51          }
52
53          public override bool IsAt(Point2D pt)
54          {
55              if (pt.X >= X && pt.Y >= Y && pt.X <= X + _width && pt.Y <=
56              Y + _height)
57              {
58                  return true;
59              }
60              else
61              {
62                  return false;
63              }
64          }
65      }
66
67  }
68
69
```

```
 1  using SplashKitSDK;
 2
 3  namespace ShapeDrawer
 4  {
 5      public class Program
 6      {
 7          private enum ShapeKind
 8          {
 9              Rectangle,
10              Circle,
11              Line
12          }
13
14          public static void Main()
15          {
16              Drawing myDrawing = new Drawing();
17              Window window = new Window("Shape Drawer", 800, 600);
18              ShapeKind kindToAdd = ShapeKind.Circle;
19
20
21              string studentID = "104762184";
22              int lastDigit = int.Parse(studentID[^1].ToString());
23              int numberOfLinesToAdd = lastDigit == 0 ? 5 : lastDigit;
24              int linesAdded = 0;
25
26              do
27              {
28                  SplashKit.ProcessEvents();
29                  SplashKit.ClearScreen();
30
31                  if (SplashKit.KeyDown(KeyCode.RKey)) kindToAdd =
                      ShapeKind.Rectangle;
32                  if (SplashKit.KeyDown(KeyCode.CKey)) kindToAdd =
                      ShapeKind.Circle;
33                  if (SplashKit.KeyDown(KeyCode.LKey)) kindToAdd =
                      ShapeKind.Line;
34
35                  if (SplashKit.MouseClicked(MouseButton.LeftButton))
36                  {
37                      Shape newShape;
38
39                      switch (kindToAdd)
40                      {
41                          case ShapeKind.Circle:
42                              newShape = new MyCircle();
43                              break;
44                          case ShapeKind.Line:
45
46                              if (linesAdded < numberOfLinesToAdd)
```

```
47                              {
48                                  newShape = new MyLine();
49                                  newShape.X = SplashKit.MouseX();
50                                  newShape.Y = SplashKit.MouseY();
51                                  myDrawing.AddShape(newShape);
52                                  linesAdded++;
53                              }
54                              continue;
55                          default:
56                              newShape = new MyRectangle();
57                              break;
58                      }
59
60                      newShape.X = SplashKit.MouseX();
61                      newShape.Y = SplashKit.MouseY();
62                      myDrawing.AddShape(newShape);
63                  }
64
65
66                  if (SplashKit.MouseClicked(MouseButton.RightButton) ||
67                      (kindToAdd != ShapeKind.Line && linesAdded > 0))
68                  {
69                      linesAdded = 0;
70                  }
71
72                  if (SplashKit.KeyDown(KeyCode.SpaceKey))
73                  {
74                      myDrawing.Background = Color.Random();
75                  }
76
77                  if (SplashKit.MouseClicked(MouseButton.RightButton))
78                  {
79                      myDrawing.SelectShapesAt(SplashKit.MousePosition());
80                  }
81
82                  if (SplashKit.KeyDown(KeyCode.DeleteKey) ||                ↵
                      SplashKit.KeyDown(KeyCode.BackspaceKey))
83                  {
84                      foreach (Shape s in myDrawing.SelectedShapes)
85                      {
86                          myDrawing.RemoveShape(s);
87                      }
88                  }
89
90                  myDrawing.Draw();
91                  SplashKit.RefreshScreen();
92              } while (!window.CloseRequested);
93          }
94      }
```

```
95  }
96
```

```
 1  using SplashKitSDK;
 2
 3
 4  namespace ShapeDrawer
 5  {
 6      public abstract class Shape
 7      {
 8
 9          private Color _color;
10          private float _x;
11          private float _y;
12          private bool _selected;
13
14
15          public Shape(Color color)
16          {
17              _x = 0.0f;
18              _y = 0.0f;
19              _color = Color.Chocolate;
20          }
21          public Shape() : this(Color.Yellow)
22          {
23
24          }
25
26
27          public Color Color
28          {
29              get { return _color; }
30              set { _color = value; }
31          }
32          public float X
33          {
34              get
35              {
36                  return _x;
37              }
38              set { _x = value; }
39
40          }
41          public float Y
42          {
43              get
44              {
45                  return _y;
46              }
47              set { _y = value; }
48
49          }
```

```
50
51
52          public abstract void Draw();
53
54
55
56          public abstract bool IsAt(Point2D pt);
57
58
59
60          public bool Selected
61          {
62              get { return _selected; }
63              set { _selected = value; }
64
65          }
66          public abstract void DrawOutline(int extra);
67
68
69
70
71
72      }
73  }
74
75
```