```
1   using System;
2   using System.Collections.Generic;
3   using System.IO;
4   using SplashKitSDK;
5
6   namespace _5._2C_Not_Complete
7   {
8       class Drawing
9       {
10          //! Fields
11          private readonly List<Shape> _shapes;
12          private Color _background;
13
14
15          //! Constructors
16          //? Default constructor, should draw a white background when initialised.
17          public Drawing(Color background)
18          {
19              _background = background;
20              _shapes = new();
21          }
22
23          public Drawing()
24              : this(Color.White)
25          {
26
27          }
28
29
30          //! Properties
31          public Color Background
32          {
```

```
33              get { return _background; }
34              set { _background = value; }
35          }
36
37          //? Readonly
38          public int ShapeCount
39          {
40              get { return _shapes.Count; }
41          }
42
43          //? Readonly, adds a selected shape to the selectedShapes array
44          public List<Shape> SelectedShapes
45          {
46              get
47              {
48                  List<Shape> selectedShapes = new();
49
50                  foreach(Shape genericShape in _shapes)
51                  {
52                      if (genericShape.Selected)
53                      {
54                          selectedShapes.Add(genericShape);
55                      }
56                  }
57                  return selectedShapes;
58              }
59          }
60
61          //! Methods and Fields
62          public void AddShape(Shape genericShape)
63          {
64              _shapes.Add(genericShape);
65          }
66
67          public void RemoveShape(Shape genericShape)
68          {
69              _shapes.Remove(genericShape);
70          }
71
72          //? Turns selected to true if shape is at mouselocation
73          public void SelectShapesAt(Point2D mouseLocation)
74          {
75              foreach(Shape genericShape in _shapes)
76              {
77                  if (!genericShape.Selected)
78                  {
79                      genericShape.Selected = genericShape.IsAt(mouseLocation);
80                  }
81              }
82          }
83
84          //? Draw da shapes
```

```
85          public void Draw()
86          {
87              SplashKit.ClearScreen(Background);
88
89              foreach (Shape genericShape in _shapes)
90              {
91                  genericShape.Draw();
92              }
93          }
94
95          //! 5.2C Code Relating to saving and loading functionality
96          public void Save(string filename)
97          {
98              StreamWriter writer = new(filename);
99
100             try
101             {
102                 writer.WriteColor(Background);
103                 writer.WriteLine(ShapeCount);
104
105                 foreach (Shape genericShape in _shapes)
106                 {
107                     genericShape.SaveTo(writer);
108                 }
109             }
110             finally
111             {
112                 writer.Close();
113             }
114         }
115
116         public void Load(string filename)
117         {
118             StreamReader reader = new(filename); //TODO create exception to handle opening non-
119             try
120             {
121                 Shape genericShape;
122                 int count;
123                 string kind;
124
125                 Background = reader.ReadColor();
126                 count = reader.ReadInteger();
127
128                 _shapes.Clear();
129
130                 for (int i = 0; i < count; i++)
131                 {
132                     kind = reader.ReadLine();
133
134                     genericShape = kind switch
135                     {
136                         "Rectangle" => new MyRectangle(),
```

```
137                         "Circle" => new MyCircle(),
138                         "Line" => new MyLine(),
139                         _ => throw new Exception(kind + "is not a valid ShapeKind"),
140                     };
141
142                     genericShape.LoadFrom(reader);
143                     AddShape(genericShape);
144                 }
145             }
146
147             finally
148             {
149                 reader.Close();
150             }
151         }
152     }
153 }
154
```