COS20007
Object Oriented Programming

Thomas Horsley

Contents

List of Figures

1 UML Class Diagrams

Class diagrams are designed to clearly depict what an object knows (the objects variables, constants and custom data types) and what it can do (the methods within the object and relationships between the objects data). A class diagram consists of the class title, the attributes held within the class and the methods the class can call.

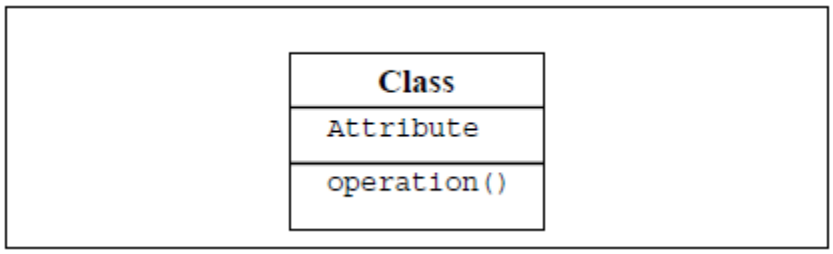


Figure 1: General class diagram structure

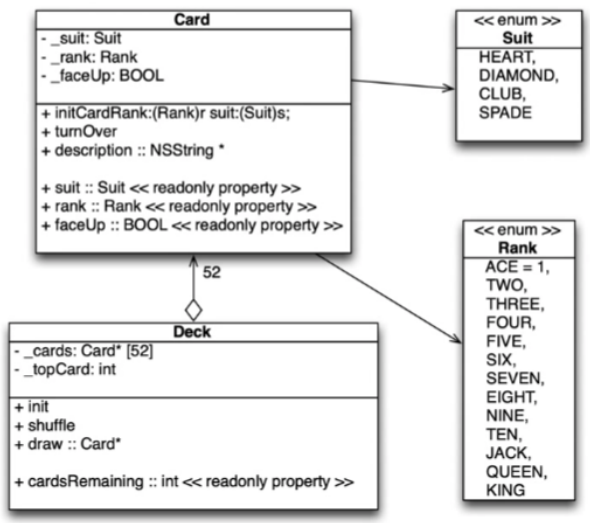
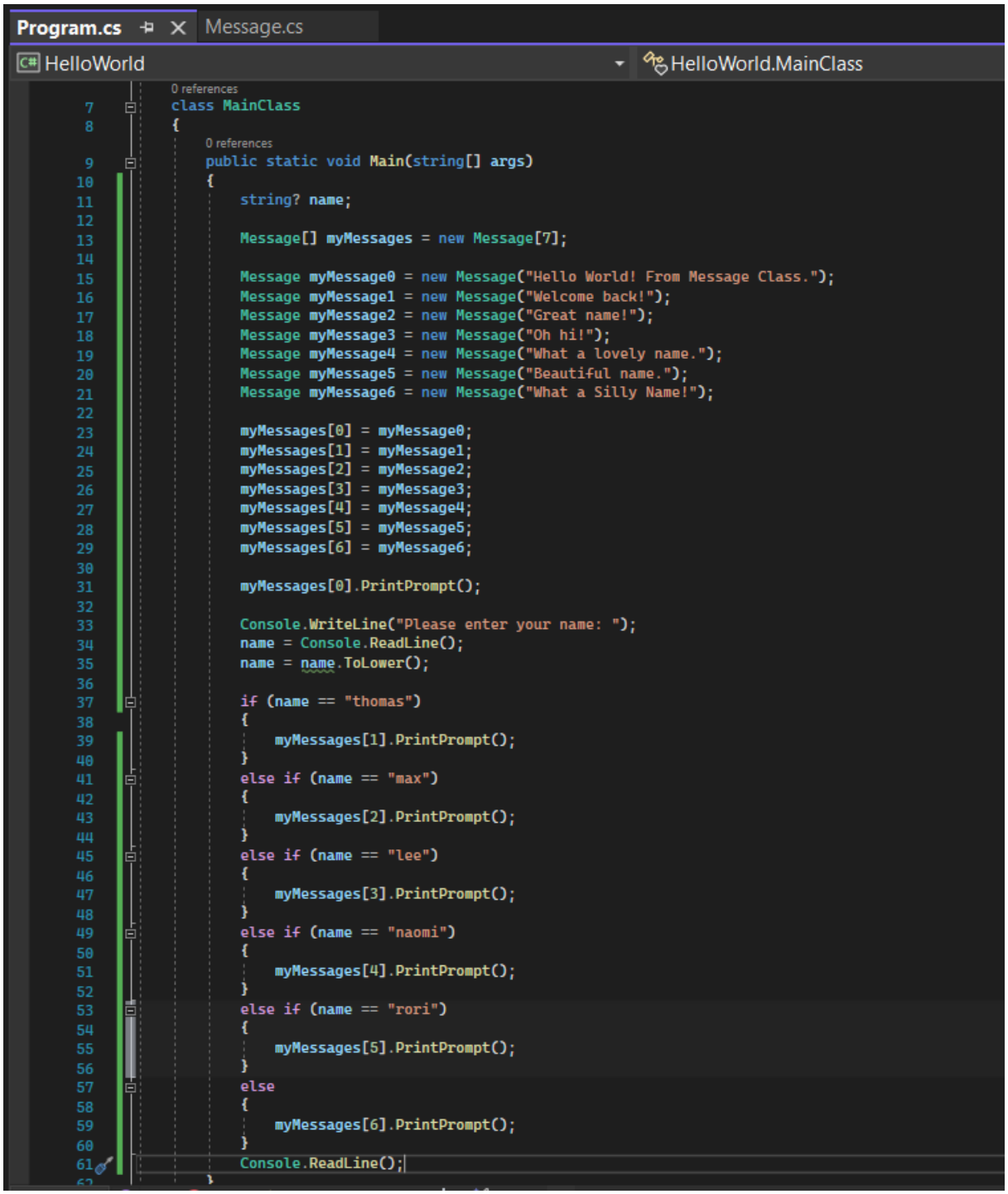


Figure 2: Example of a set of UML Class Diagrams for a deck of playing cards

The access modifier (the '+' and '−' found before each field) indicates whether the field is private (denoted '−') or public (denoted '+').

Operations are written first by explicitly denoting their access modifier and name, followed by the parameters passed into the operation, followed by the return value.

2 Portfolio Task 1.2P - OOP Hello World



```
Program.cs   Message.cs
HelloWorld                                  HelloWorld.MainClass
                0 references
7               class MainClass
8               {
                    0 references
9                   public static void Main(string[] args)
10                  {
11                      string? name;
12
13                      Message[] myMessages = new Message[7];
14
15                      Message myMessage0 = new Message("Hello World! From Message Class.");
16                      Message myMessage1 = new Message("Welcome back!");
17                      Message myMessage2 = new Message("Great name!");
18                      Message myMessage3 = new Message("Oh hi!");
19                      Message myMessage4 = new Message("What a lovely name.");
20                      Message myMessage5 = new Message("Beautiful name.");
21                      Message myMessage6 = new Message("What a Silly Name!");
22
23                      myMessages[0] = myMessage0;
24                      myMessages[1] = myMessage1;
25                      myMessages[2] = myMessage2;
26                      myMessages[3] = myMessage3;
27                      myMessages[4] = myMessage4;
28                      myMessages[5] = myMessage5;
29                      myMessages[6] = myMessage6;
30
31                      myMessages[0].PrintPrompt();
32
33                      Console.WriteLine("Please enter your name: ");
34                      name = Console.ReadLine();
35                      name = name.ToLower();
36
37                      if (name == "thomas")
38                      {
39                          myMessages[1].PrintPrompt();
40                      }
41                      else if (name == "max")
42                      {
43                          myMessages[2].PrintPrompt();
44                      }
45                      else if (name == "lee")
46                      {
47                          myMessages[3].PrintPrompt();
48                      }
49                      else if (name == "naomi")
50                      {
51                          myMessages[4].PrintPrompt();
52                      }
53                      else if (name == "rori")
54                      {
55                          myMessages[5].PrintPrompt();
56                      }
57                      else
58                      {
59                          myMessages[6].PrintPrompt();
60                      }
61                      Console.ReadLine();
                    }
```

Figure 3: *Program.cs* is the entry point for the Hello World program and houses most of the functionality

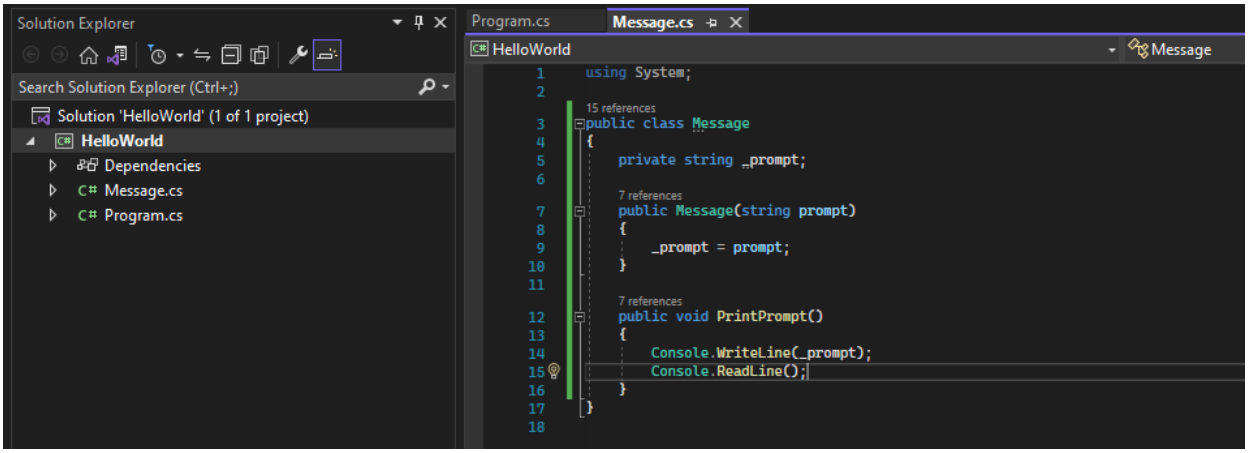
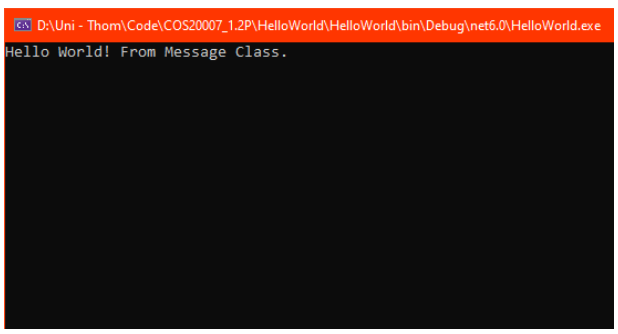
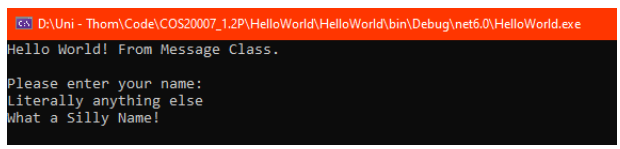


Figure 4: Code encapsulating the Message class

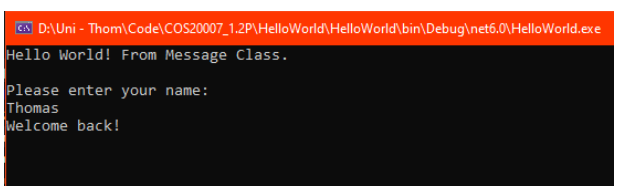


(a) Hello World!

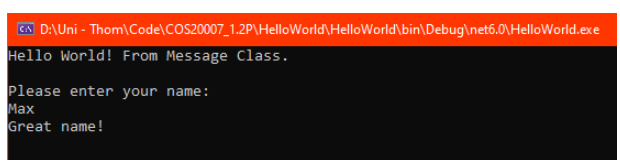


(b) Test 1 - Silly name tester

Figure 5: **1.2P** Hello World Solution and Silly Name Tester



(a) Test 2 - Thomas's Prompt Test

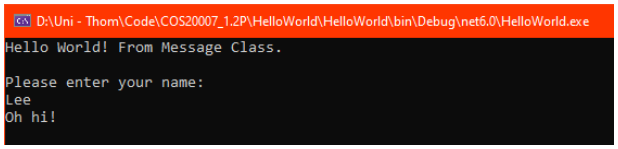


(b) Test 3 - Max's Prompt

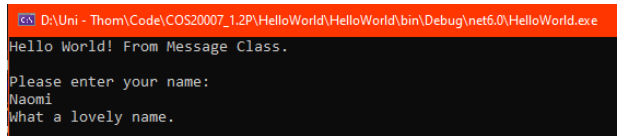
Figure 6: Thomas and Max prompt tests

NOTE:

My solution and images can also be found at https://github.com/KingSchlock/COS20007



(a) Test 4 - Lee's Prompt Test



(b) Test 5 - Naomi's Prompt

Figure 7: Lee and Naomi prompt tests

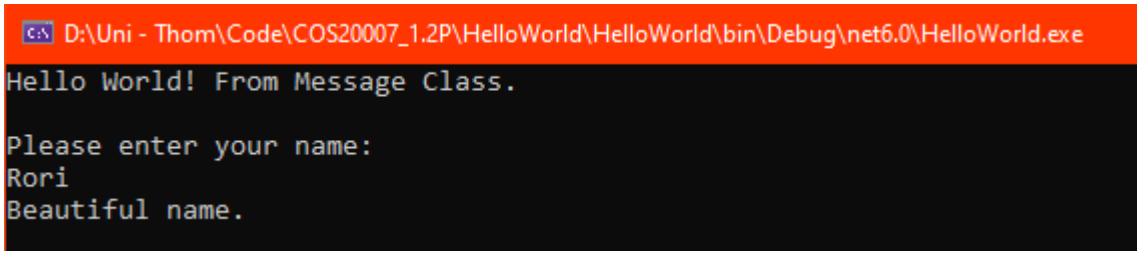


Figure 8: Test 6 - Rori's Prompt

3 Portfolio Task 2.1P - Counter Class

```
Counter.cs    Program.cs
CounterClass                                  CounterClass.CounterProgram
1   using System;
2
3   namespace CounterClass
4   {
        2 references
5       internal class CounterProgram
6       {
            2 references
7           private static void PrintCounters(Counter[] myCounters)
8           {
9               foreach (Counter counter in myCounters)
10              {
11                  Console.WriteLine("{0} is {1}", counter.NameCounter, counter.Tick);
12              }
13              Console.ReadLine();
14          }
15
            0 references
16          static void Main(string[] args)
17          {
18              Counter[] myCounters = new Counter[3];
19
20              myCounters[0] = new Counter("Counter 1");
21              myCounters[1] = new Counter("Counter 2");
22              myCounters[2] = myCounters[0];
23
24              for(int i = 0; i < 10; i++)
25              {
26                  myCounters[0].IncrementCounter();
27              }
28
29              for (int i = 0; i < 15; i++)
30              {
31                  myCounters[1].IncrementCounter();
32              }
33
34              CounterProgram.PrintCounters(myCounters);
35              myCounters[2].ResetCounter();
36
37              CounterProgram.PrintCounters(myCounters);
38          }
39      }
40  }
90 %    No issues found
```

Figure 9: *Program.cs* is the entry point for the Counter Class program.

NOTE:

My solution and images can also be found at https://github.com/KingSchlock/COS20007

```
Counter.cs   Program.cs
CounterClass
using System;

namespace CounterClass
{
    7 references
    public class Counter
    {
        private string _name;
        private int _count;

        2 references
        public Counter(string name)
        {
            _name = name;
            _count = 0;
        }

        1 reference
        public string NameCounter
        {
            get
            {
                return _name;
            }
            set
            {
                _name = value;
            }
        }

        1 reference
        public int Tick
        {
            get
            {
                return _count;
            }
        }

```

Figure 10: *Counter.cs* containing it's fields and property operations

```
        2 references
        public void IncrementCounter()
        {
            _count++;
        }

        1 reference
        public void ResetCounter()
        {
            _count = 0;
        }

        0 references
        public void PrintCounter()
        {
            Console.WriteLine($"Name is {_name}, Counter is {_count}");
        }
    }
}

```

Figure 11: *Counter.cs* increment, reset and print operations

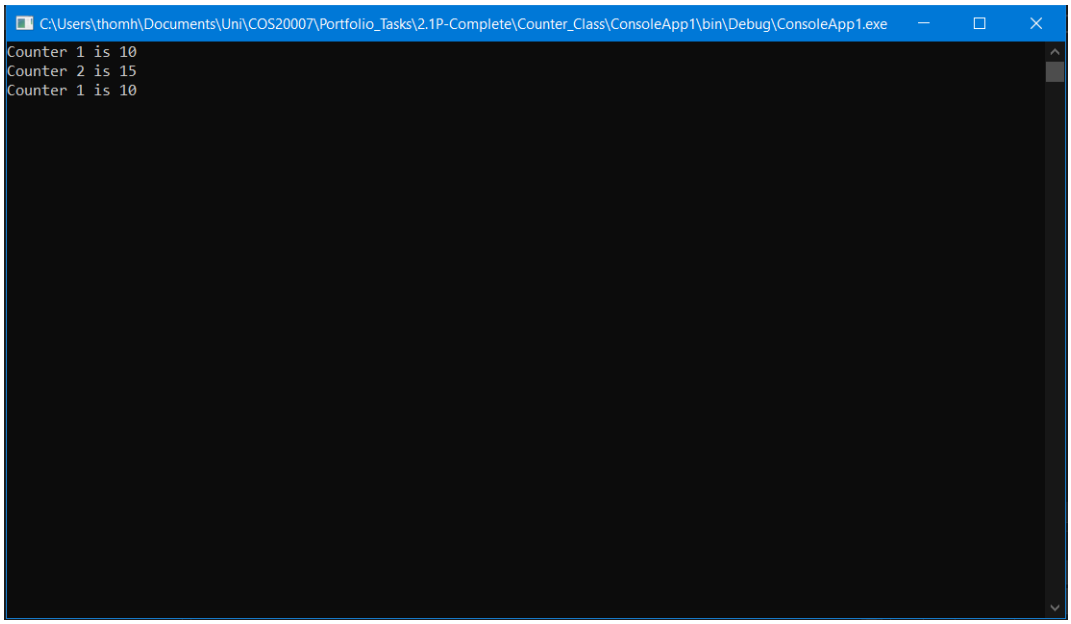


Figure 12: Counter Class first print

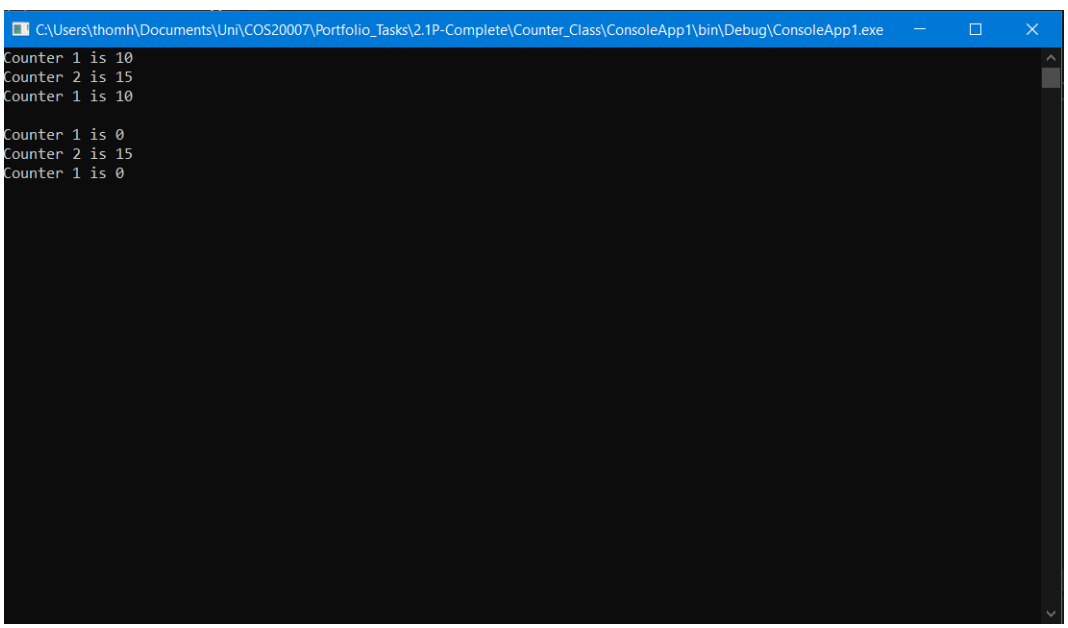


Figure 13: Counter Class second print

4 Portfolio Task 2.2P - Shape Drawer

Shape.cs | Program.cs

ShapeDrawer | ShapeDrawer.Program

```
using SplashKitSDK;

namespace ShapeDrawer
{
    0 references
    public class Program
    {
        0 references
        public static void Main()
        {
            Window window = new Window("Shape Drawer: Thomas Horsley - 103071494", 800, 600);
            Shape myShape = new Shape(Color.Green, 0, 0, 100, 100);

            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();

                Point2D pt = SplashKit.MousePosition();

                // Set x to be mouseX and y to be MouseY if mouse clicked is true
                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    myShape.X = SplashKit.MouseX();
                    myShape.Y = SplashKit.MouseY();
                }

                if (myShape.IsAt(pt) && SplashKit.KeyTyped(KeyCode.SpaceKey))
                {
                    myShape.Color = Color.RandomRGB(255);
                }

                myShape.IsAt(pt); //for debugging, remove after

                myShape.Draw();

                SplashKit.RefreshScreen();
            } while (!window.CloseRequested);
        }
    }
}
```

Figure 14: *Program.cs* is the entry point for the ShapeDrawer program.

NOTE:

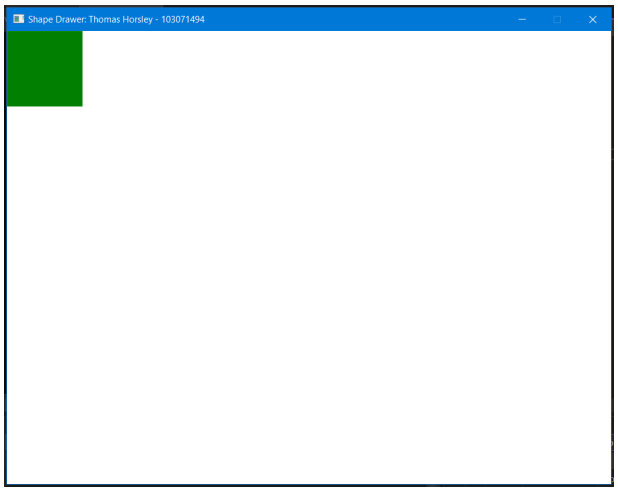
My solution and images can also be found at https://github.com/KingSchlock/COS20007

```
Shape.cs*   Program.cs
ShapeDrawer                                   ShapeDrawer.Shape
1   using SplashKitSDK;
2
3   namespace ShapeDrawer
4   {
        3 references
5       public class Shape
6       {
7           private Color _color;
8           private float _x;
9           private float _y;
10          private int _width;
11          private int _height;
12
            1 reference
13          public Shape(Color color, float x, float y, int width, int height)
14          {
15              _color = color;
16              _x = x;
17              _y = y;
18              _width = width;
19              _height = height;
20          }
21
            1 reference
22          public Color Color
23          {
24              get { return _color; }
25              set { _color = value; }
26          }
27
            1 reference
28          public float X...
33
            1 reference
34          public float Y...
39
            0 references
40          public int Width...
45
            0 references
46          public int Height...
51
```

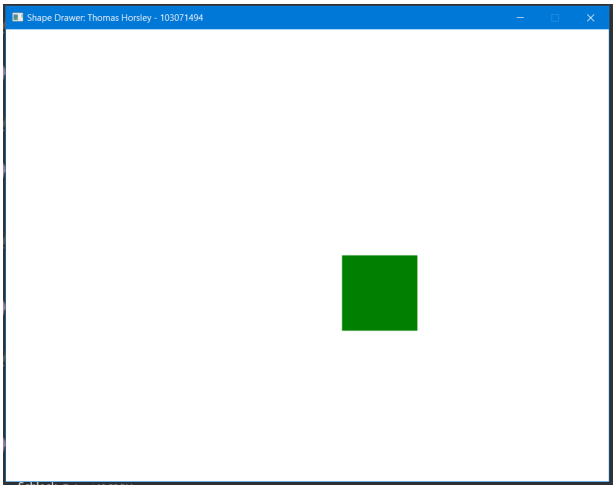
Figure 15: *Shape.cs* fields and property operations

```
52          public void Draw()
53          {
54              SplashKit.FillRectangle(_color, _x, _y, _width, _height);
55          }
56
            2 references
57          public bool IsAt(Point2D pt)
58          {
59              /* if point.X is within x and  x + width and point.Y is withiny and y + height
60               * pt will be within the bounds of our rectangle
61               */
62
63              if (_x < pt.X && pt.X < (_x + _width) && _y < pt.Y && pt.Y < (_y+ _height))
64              {
65                  return true;
66              }
67              else
68              {
69                  return false;
70              }
71          }
72      }
73  }
```

Figure 16: *Shape.cs* draw and IsAt operations

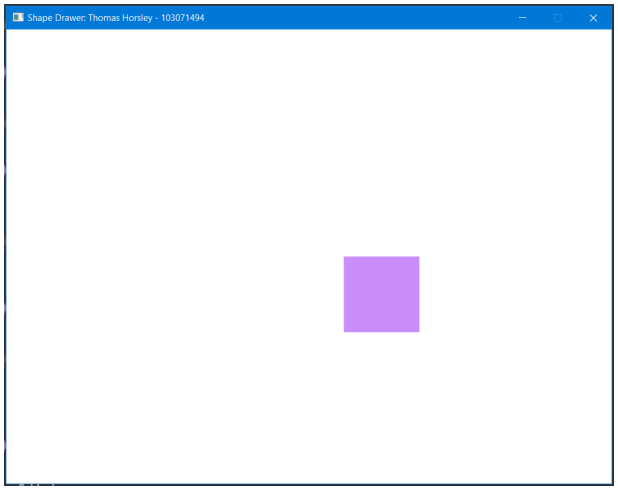


(a) initial shape state from myShape.Draw();

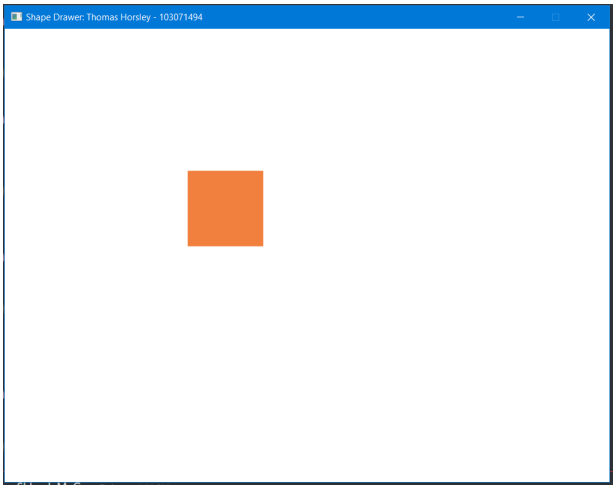


(b) setting position(x, y) to be Mouse.X and Mouse.Y respectively

Figure 17: Shape drawer functionality examples



(a) Shape drawer color change functionality



(b) Shape drawer further examples

Figure 18: Shape drawer color change functionality

5 Portfolio Task 2.3P