Simon Dunkley 1628623

Pass Task 11 – Shape Drawer

Related Learning Outcomes

ULO1 – Explain the OO Principles

This exercise demonstrated the ability to draw many shapes through new code that could manage a collection of shapes. The class created could manage and draw multiple shapes with the ability to add, remove, select and draw new shapes.
Learnt about aggregation showing the relationship between the different classes.
Learnt how to use lists to store the objects. The Task demonstrated using multiple constructors with different parameters to get different results such as how default settings work.

This exercise demonstrated the ability to inherit common features from a parent class.
This exercise demonstrated subtype polymorphism which means that you can use subtypes of objects as a kind of the parent class. Polymorphism is used to avoid duplication of code. This exercise taught me that I can use types within other types with enumeration. The exercise showed using abstract to override the method.

Shape requires an abstract class so that the children/subclasses could be used for unique implementation for each method and fields. Making the shape class Abstract resulted in it being incomplete requiring the children to fill in the virtual or abstract methods with their override methods for the different types of shapes. An example of this would be for the circle, you need to actually draw a circle so all shapes don't need to draw a circle you use the abstract method in the shape class to show that it exists for the children but they have to fill out the information inside the methods. Making the Shape class abstract has the benefit of not repeating all the Properties, methods and fields that can be used across all the different types of shapes.

ULO2 – Use OO Language and Library

Demonstrated class, method, field and constructor declaration. We used the library in accessing System.Collections.Generic. Learnt about class libraries that include number of classes that can be used to create objects in the program. The task specifically taught about managing collections of objects. The collection object maintains a number of objects for you. The list type of class can add, remove and fetch objects from the list for you.
The task demonstrated using virtual to override the methods for specific subclasses such as to draw a circle for the circle class instead of drawing a rectangle.

ULO3 – Design, Develop and Test using an IDE

The code was developed using Xamarin Studio to build and run the program, as well as integrated debugging features to step and inspect values. Learnt how to use the IDE to add methods. Demonstrated use of NUnit to test and debug the game to make sure everything was working correctly and how to add a method based on the Test class.

ULO4 – Communicate using UML Diagrams

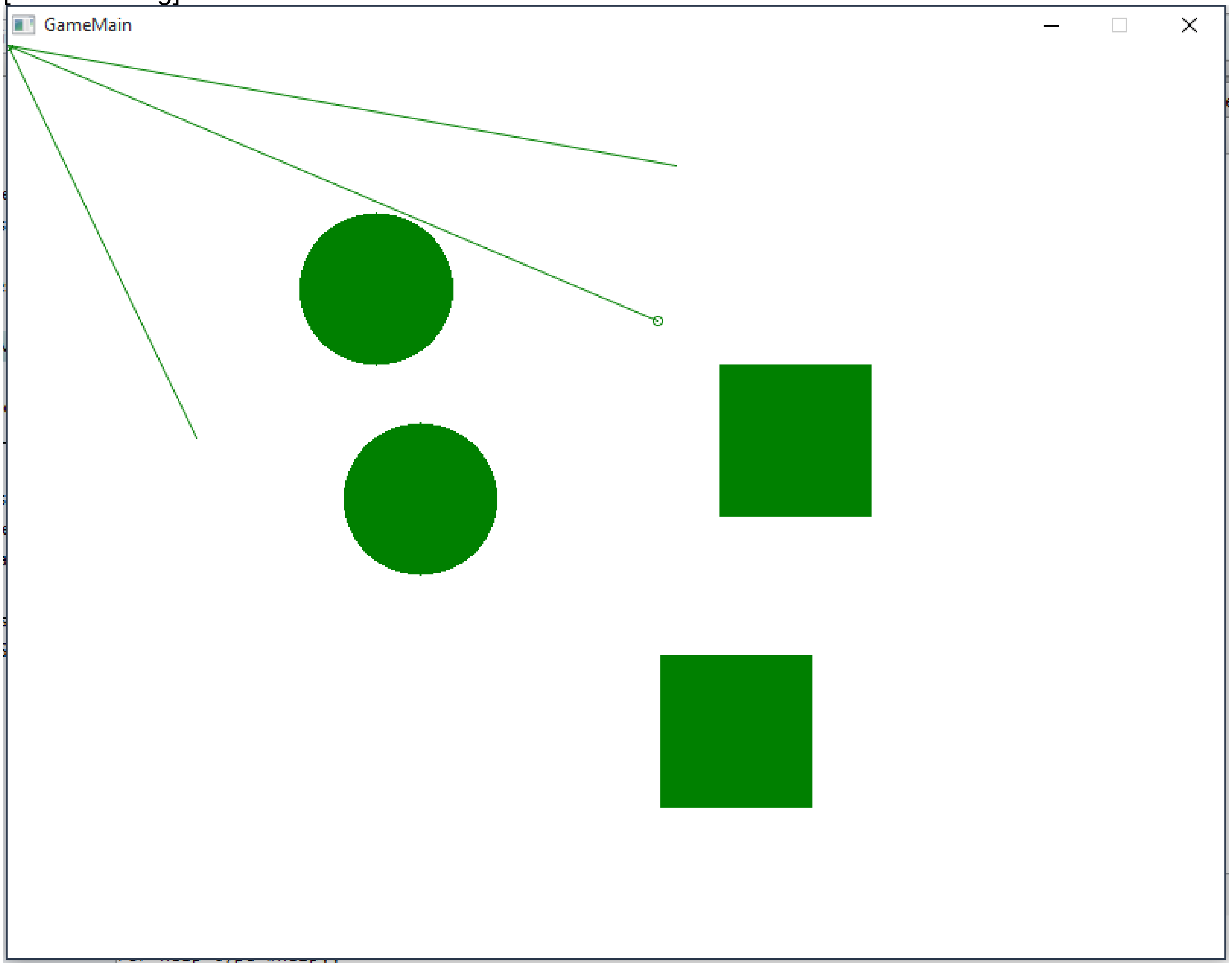
I learned how to interpret a UML class diagram and write the related code.

ULO5 – Describe Elements of Good OO Design

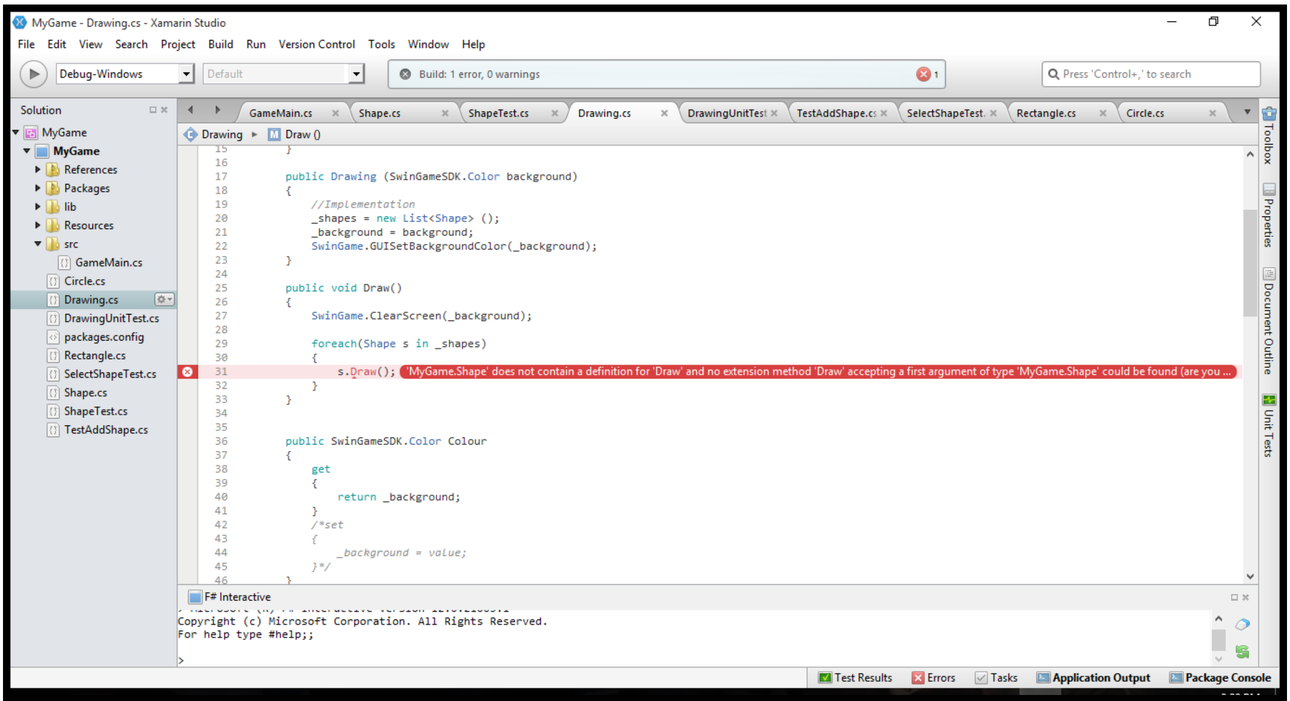
The exercise demonstrated correct use of C# coding conventions.

Screenshots

[code running]



[use of IDE]



MyGame - Drawing.cs - Xamarin Studio

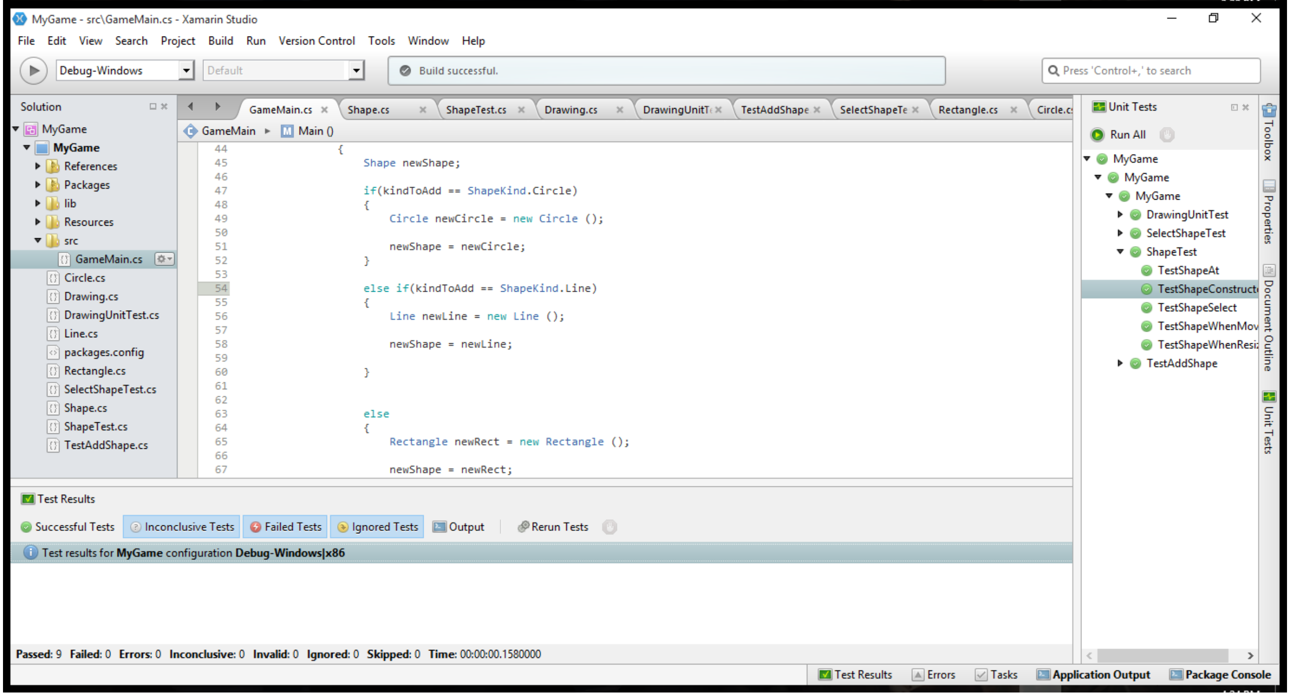
Build: 1 error, 0 warnings

```
        }

        public Drawing (SwinGameSDK.Color background)
        {
            //Implementation
            _shapes = new List<Shape> ();
            _background = background;
            SwinGame.GUISetBackgroundColor(_background);
        }

        public void Draw()
        {
            SwinGame.ClearScreen(_background);

            foreach(Shape s in _shapes)
            {
                s.Draw();
            }
        }


        public SwinGameSDK.Color Colour
        {
            get
            {
                return _background;
            }
            /*set
            {
                _background = value;
            }*/
        }
```

'MyGame.Shape' does not contain a definition for 'Draw' and no extension method 'Draw' accepting a first argument of type 'MyGame.Shape' could be found (are you ...

F# Interactive

```
Copyright (c) Microsoft Corporation. All Rights Reserved.
For help type #help;;

>
```



MyGame - src\GameMain.cs - Xamarin Studio

Build successful.

```
                {
                    Shape newShape;

                    if(kindToAdd == ShapeKind.Circle)
                    {
                        Circle newCircle = new Circle ();

                        newShape = newCircle;
                    }

                    else if(kindToAdd == ShapeKind.Line)
                    {
                        Line newLine = new Line ();

                        newShape = newLine;

                    }


                    else
                    {
                        Rectangle newRect = new Rectangle ();

                        newShape = newRect;
```

Test results for **MyGame** configuration **Debug-Windows|x86**

Passed: 9 **Failed:** 0 **Errors:** 0 **Inconclusive:** 0 **Invalid:** 0 **Ignored:** 0 **Skipped:** 0 **Time:** 00:00:00.1580000