

SWINBURNE UNIVERSITY OF TECHNOLOGY

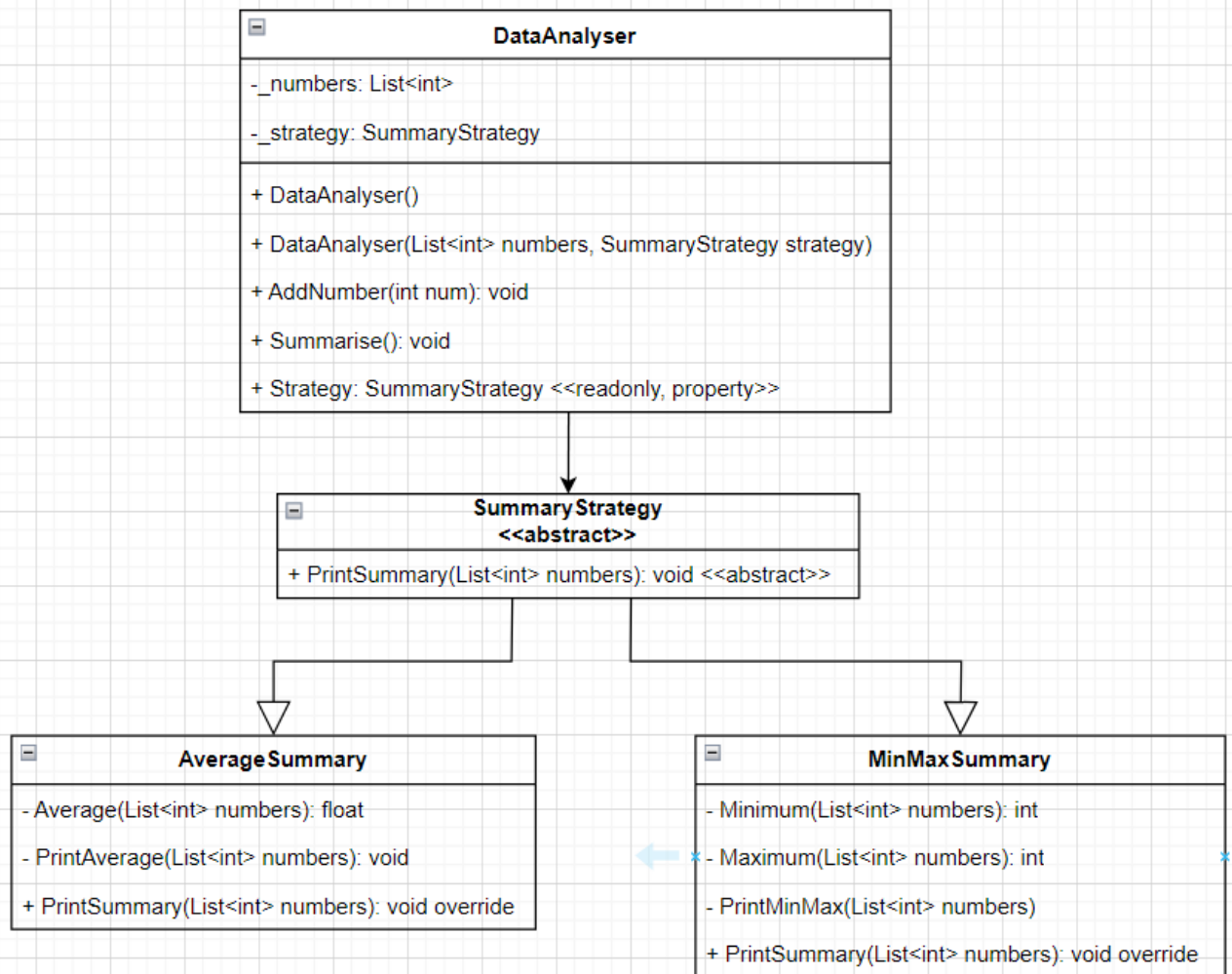
COS20007 OBJECT ORIENTED PROGRAMMING

---

## Semester test

---

PDF generated at 02:13 on Saturday 11<sup>th</sup> November, 2023



```
1 // See https://aka.ms/new-console-template for more information
2 using SemesterTest;
3
4 ///          Initialising number list          ///
5 List<int> nums = new List<int>(new int[] { 1, 2, 3, 4});
6
7 /// Initialising strategy type as MinMax and DataAnalyser object ///
8 SummaryStrategy strategy = new MinMaxSummary();
9 DataAnalyser analyser = new DataAnalyser(nums, strategy);
10
11 /// Calling Summarise function with MinMax strategy ///
12 Console.WriteLine("MinMax Summary Strategy:");
13 analyser.Summarise();
14
15 /// Manually adding additional numbers into the list ///
16 analyser.AddNumber(7);
17 analyser.AddNumber(9);
18 analyser.AddNumber(12);
19
20 /// Change strategy type from MinMax to Average ///
21 strategy = new AverageSummary();
22 analyser = new DataAnalyser(nums, strategy);
23
24 /// Calling Summarise function with Average strategy ///
25 Console.WriteLine("Average Summary Strategy:");
26 analyser.Summarise();
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SemesterTest
8  {
9      public abstract class SummaryStrategy
10     {
11         public abstract void PrintSummary(List<int> numbers);
12     }
13 }
```

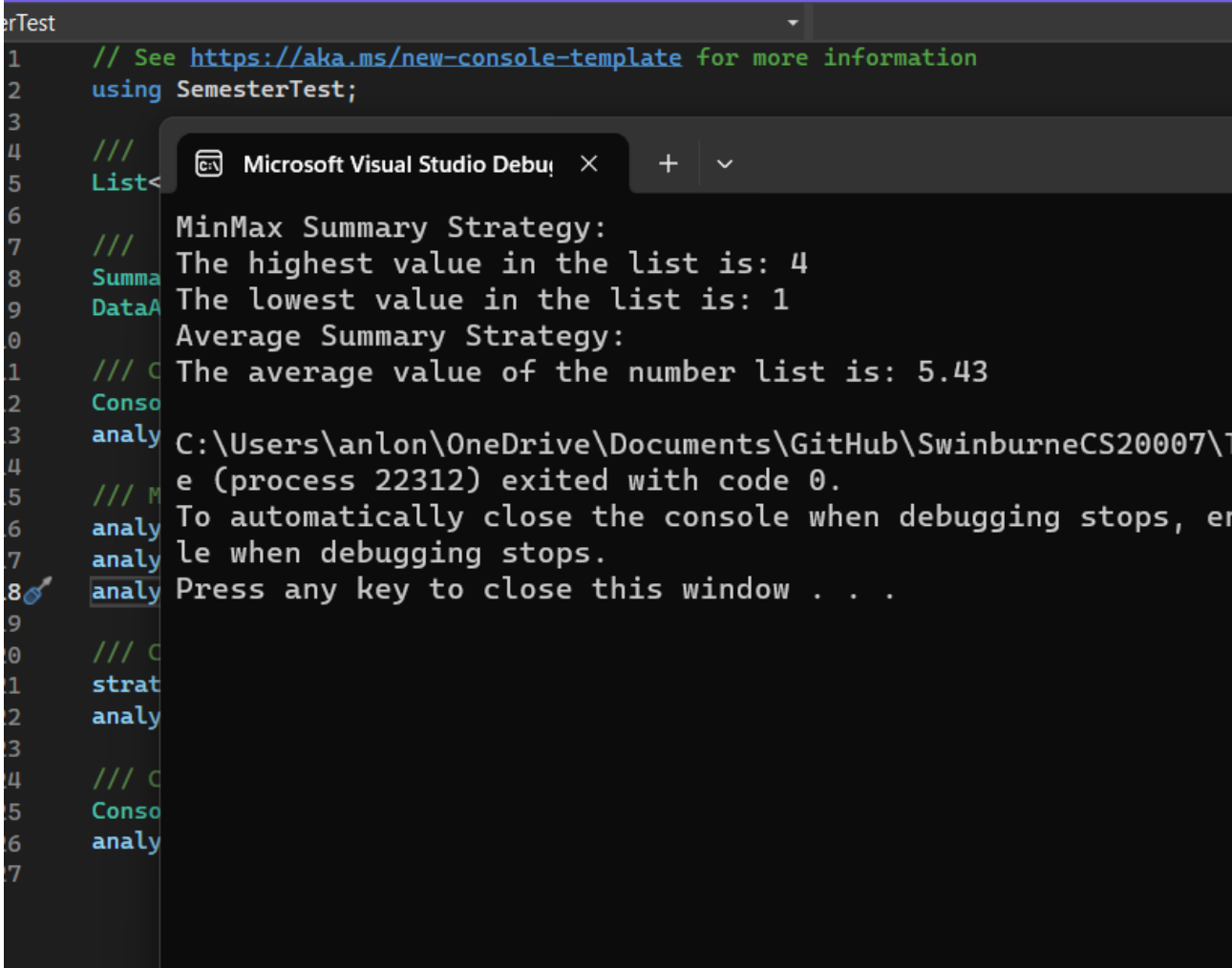
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SemesterTest
8  {
9      public class MinMaxSummary : SummaryStrategy
10     {
11         /// <summary>
12         /// Algorithm that caluclate the lowest value in List<int>numbers
13         /// </summary>
14         /// <param name="numbers"></param>
15         /// <returns></returns>
16         private int Minimum(List<int> numbers)
17         {
18             int lowest = numbers[0];
19             for (int i = 0; i < numbers.Count; i++)
20             {
21                 if (numbers[i] < lowest)
22                     lowest = numbers[i];
23             }
24             return lowest;
25         }
26
27         /// <summary>
28         /// Algorithm that caluclate the highest value in List<int>numbers
29         /// </summary>
30         /// <param name="numbers"></param>
31         /// <returns></returns>
32         private int Maximum(List<int> numbers)
33         {
34             int highest = numbers[0];
35             for (int i = 0; i < numbers.Count; i++)
36             {
37                 if (numbers[i] > highest)
38                     highest = numbers[i];
39             }
40             return highest;
41         }
42
43         /// <summary>
44         /// Responsible for displaying the output in Program.cs
45         /// </summary>
46         /// <param name="numbers"></param>
47         private void PrintMinMax(List<int> numbers)
48         {
49             int max = Maximum(numbers);
50             int min = Minimum(numbers);
51
52             Console.WriteLine($"The highest value in the list is: {max}");
53             Console.WriteLine($"The lowest value in the list is: {min}");
```

```
54         }
55
56         /// <summary>
57         /// Called from DataAnalyser class through the abstract class SummaryStrategy
58         /// </summary>
59         /// <param name="numbers"></param>
60         public override void PrintSummary(List<int> numbers)
61         {
62             PrintMinMax(numbers);
63         }
64     }
65 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SemesterTest
8  {
9      public class AverageSummary : SummaryStrategy
10     {
11         /// <summary>
12         /// Algorithm that calculate the average value in List<int>numbers
13         /// </summary>
14         /// <param name="numbers"></param>
15         /// <returns></returns>
16         private float Average(List<int> numbers)
17         {
18             float sum = 0;
19
20             for(int i = 0; i < numbers.Count; i++)
21             {
22                 sum += numbers[i];
23             }
24
25             return sum / numbers.Count;
26         }
27
28         /// <summary>
29         /// Responsible for displaying output in Program.cs
30         /// </summary>
31         /// <param name="numbers"></param>
32         private void PrintAverage(List<int> numbers)
33         {
34             float average = Average(numbers);
35             Console.WriteLine($"The average value of the number list is:
36 ↵ {average:.0#}");
37         }
38
39         /// <summary>
40         /// Called from DataAnalyser class through the abstract class SummaryStrategy
41         /// </summary>
42         /// <param name="numbers"></param>
43         public override void PrintSummary(List<int> numbers)
44         {
45             PrintAverage(numbers);
46         }
47     }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SemesterTest
8  {
9      public class DataAnalyser
10     {
11         private List<int> _numbers;
12         private SummaryStrategy _strategy;
13
14         public DataAnalyser() : this(new List<int>(), new AverageSummary())
15         {
16
17         }
18         public DataAnalyser(List<int> numbers, SummaryStrategy strategy)
19         {
20             _numbers = numbers;
21             _strategy = strategy;
22         }
23
24         /// <summary>
25         /// Allow user to manually add any number into List<int> _numbers
26         /// </summary>
27         /// <param name="number"></param>
28         public void AddNumber(int number)
29         {
30             _numbers.Add(number);
31         }
32
33         /// <summary>
34         /// Global function used in Program.cs to access hidden functions
35         /// </summary>
36         public void Summarise()
37         {
38             _strategy.PrintSummary(_numbers);
39         }
40
41         public SummaryStrategy Strategy
42         {
43             get { return _strategy; }
44         }
45     }
46 }
```





The screenshot shows a Visual Studio Code editor with a C# file named `SemesterTest.cs`. The code defines a `SemesterTest` class with methods for calculating min/max, sum, and average of a list. The console output shows the results of these calculations for a specific list.

```
1 // See https://aka.ms/new-console-template for more information
2 using SemesterTest;
3
4 ///
5 List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
6
7 /// MinMax Summary Strategy:
8 Summa The highest value in the list is: 10
9 DataA The lowest value in the list is: 1
10
11 /// Average Summary Strategy:
12 Conso The average value of the number list is: 5.43
13 analy C:\Users\anlon\OneDrive\Documents\GitHub\SwinburneCS20007\SemesterTest\bin\Debug\net6.0\SemesterTest.exe (process 22312) exited with code 0.
14
15 /// To automatically close the console when debugging stops, enable the 'AutoCloseConsole' property in the launch configuration.
16 analy Press any key to close this window . . .
17
18 ///
19 strat
20 analy
21
22 ///
23 Conso
24 analy
```

## **1. Describe the principle of polymorphism and how it was used in Task 1**

Polymorphism is a concept of reusing code and expanding the program without impacting the original code. It allows developers to create various classes by recycling the same interface, which saves a lot of time and easy to debug. This concept was used in Task 1 to decrease the amount of repetitive code and the amount of memory space used to run the program.

## **2. Using an example, explain the principle of abstraction. In your answer, refer to how classes in OO programs are designed.**

Abstraction is one of the four key concepts in OOP, with the sole purpose of hiding unnecessary information from the user and only show the important attributes of the program. For example, instead of calling the functions such as Average in Average Summary class and Minimum and Maximum in MinMax strategy. This put program at risk of revealing the internal classes and data of the program. Instead of waiting for those two possible “if” conditions in Summarise function, DataAnalyser class would use the abstract class to call for the PrintSummary function that both MinMaxSummary and AverageSummary classes contain only once. This hides the necessary functions from both classes to perform calculations and return the summarised results.

## **3. What was the issue with the original design in Task 1? Consider what would happen if we had 50 different summary approaches to choose from instead of just 2.**

The main issue with the original design in Task 1 was the possibility of creating repetitive code that would be redundant for the program, affecting its performance and efficiency. It is evident in the original design where it requires two variables for containing each type of summary methods to access the functions that are responsible for calculation. If we had 50 different summary approaches, the DataAnalyser class would contain 50 different variables for each type of summary strategy, as well as 50 “if” conditions within the Summarise function. Therefore, programs that implement the original design in Task 1 would inefficient, taking up a huge amount of memory space, and potentially have extremely poor performance.