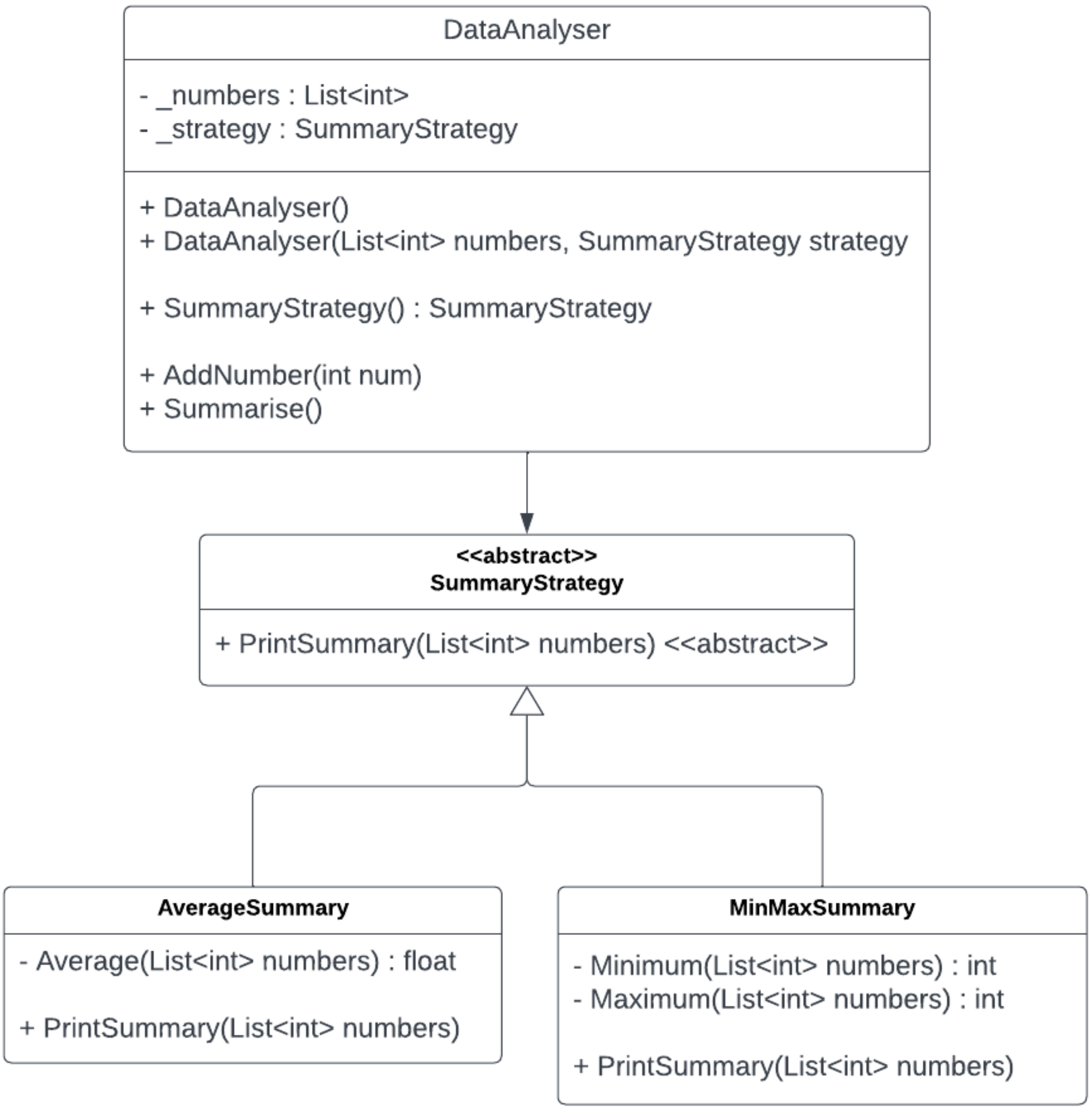
Swinburne University of Technology

COS20007 Object Oriented Programming

T1 - Semester Test

PDF generated at 21:31 on Thursday 27th April, 2023


DataAnalyser
- _numbers : List<int>
- _strategy : SummaryStrategy
+ DataAnalyser()
+ DataAnalyser(List<int> numbers, SummaryStrategy strategy
+ SummaryStrategy() : SummaryStrategy
+ AddNumber(int num)
+ Summarise()
<<abstract>>
SummaryStrategy
+ PrintSummary(List<int> numbers) <<abstract>>
AverageSummary
- Average(List<int> numbers) : float
+ PrintSummary(List<int> numbers)
MinMaxSummary
- Minimum(List<int> numbers) : int
- Maximum(List<int> numbers) : int
+ PrintSummary(List<int> numbers)

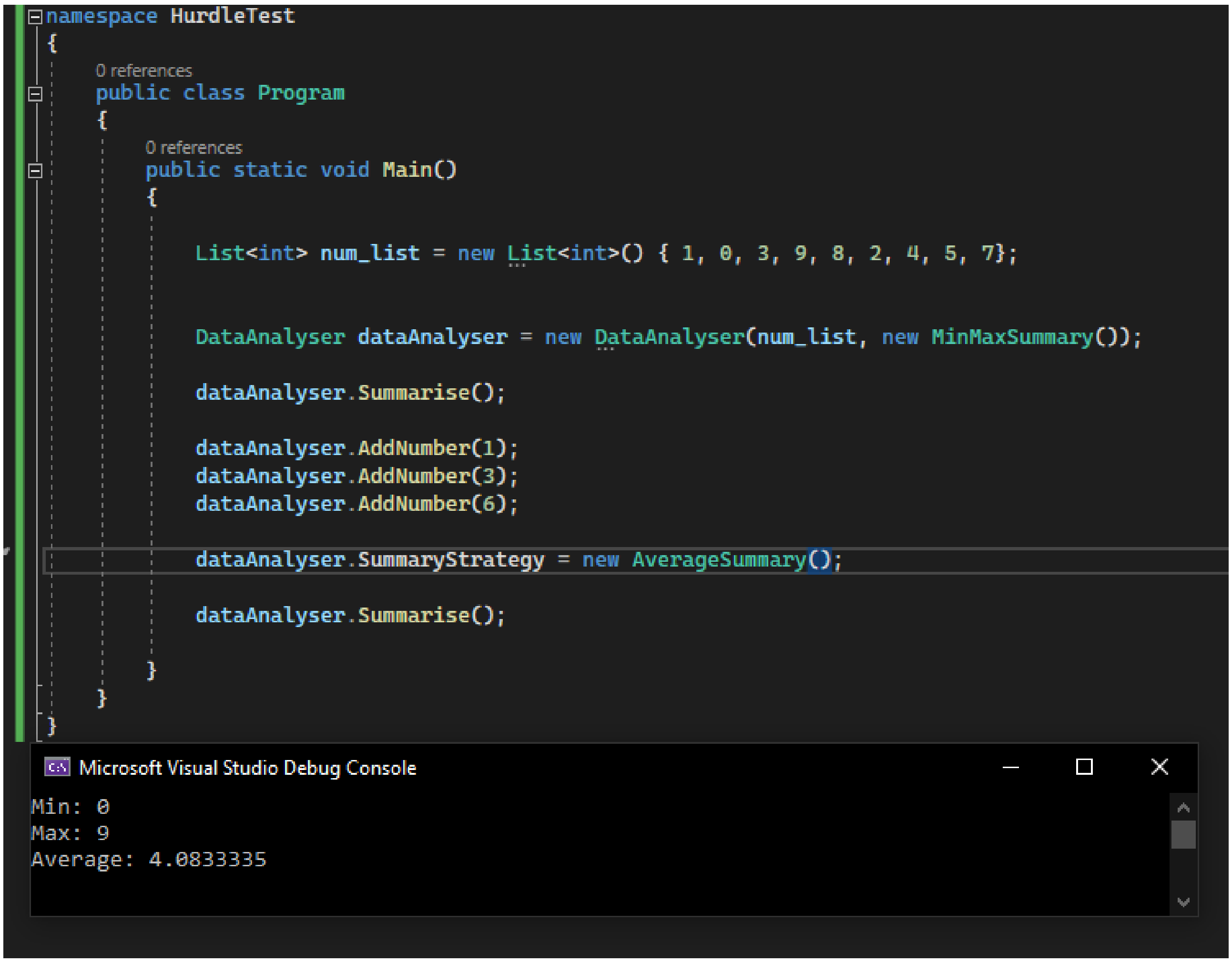
```
using System;

namespace HurdleTest
{
    public class Program
    {
        public static void Main()
        {

            List<int> num_list = new List<int>() { 1, 0, 3, 9, 8, 2, 4, 5, 7};


            DataAnalyser dataAnalyser = new DataAnalyser(num_list, new
 ↪  MinMaxSummary());

            dataAnalyser.Summarise();

            dataAnalyser.AddNumber(1);
            dataAnalyser.AddNumber(3);
            dataAnalyser.AddNumber(6);

            dataAnalyser.SummaryStrategy = new AverageSummary();

            dataAnalyser.Summarise();

        }
    }
}
```

```
using System;

namespace HurdleTest
{
    public class DataAnalyser
    {

        private List<int> _numbers;

        private SummaryStrategy _strategy;


        public DataAnalyser(List<int> numbers, SummaryStrategy strategy) {

            _numbers = numbers;
            _strategy = strategy;

        }

        public DataAnalyser() : this(new List<int> { }, new AverageSummary()) { }


        public SummaryStrategy SummaryStrategy { get { return _strategy; } set {
        ↪  _strategy = value; } }


        public void AddNumber(int num) { _numbers.Add(num); }

        public void Summarise() { _strategy.PrintSummary(_numbers); }

    }
}
```

```
using System;

namespace HurdleTest
{
    public abstract class SummaryStrategy
    {

        public abstract void PrintSummary(List<int> numbers);

    }
}
```

```csharp
using System;

namespace HurdleTest
{
    public class MinMaxSummary : SummaryStrategy
    {

        private int Maximum(List<int> numbers) {

            int max = numbers[0];

            foreach (int value in numbers)
            {
                if (value > max) { max = value; }
            }

            return max;

        }

        private int Minimum(List<int> numbers) {

            int min = numbers[0];

            foreach (int value in numbers)
            {
                if (value < min) { min = value; }
            }

            return min;

        }

        public override void PrintSummary(List<int> numbers)
        {

            Console.WriteLine($"Min: {Minimum(numbers)}");
            Console.WriteLine($"Max: {Maximum(numbers)}");

        }

    }
}
```

```
using System;

namespace HurdleTest
{
    public class AverageSummary : SummaryStrategy
    {

        private float Average(List<int> numbers) {

            float sum = 0;

            foreach (int value in numbers) { sum += value; }

            return (sum / (float)numbers.Count);

        }

        public override void PrintSummary(List<int> numbers) {

            Console.WriteLine($"Average: {Average(numbers)}");

        }

    }
}
```



```
namespace HurdleTest
{
    0 references
    public class Program
    {
        0 references
        public static void Main()
        {

            List<int> num_list = new List<int>() { 1, 0, 3, 9, 8, 2, 4, 5, 7};


            DataAnalyser dataAnalyser = new DataAnalyser(num_list, new MinMaxSummary());

            dataAnalyser.Summarise();

            dataAnalyser.AddNumber(1);
            dataAnalyser.AddNumber(3);
            dataAnalyser.AddNumber(6);

            dataAnalyser.SummaryStrategy = new AverageSummary();

            dataAnalyser.Summarise();

        }
    }
}
```

Microsoft Visual Studio Debug Console

```
Min: 0
Max: 9
Average: 4.0833335
```

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

Polymorphism allows objects to take on many forms it involves the implementation of a parent class to manipulate the type that the child classes' possible classifications, this flexibility allows objects of multiple classes to be held and passed under common parameters.

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

Abstraction removes and generalizes features of a program by allowing several components of the program to override the given feature. Abstraction cuts down on the need to repeat code as commands from a parent class can be used in the override class.

For example, take a program that is required to draw various shapes onto a window, these shapes would require their own classes since shapes have different properties such as length, width, radius, etc. The shapes would also require other common properties and functions such as position and colour, therefore a new general shape class would be appropriate as it can hold all the common variables and methods while allowing the properties that are specific to a certain shape to be held in their respective classes. This abstraction also allows all these methods to be held under a common name making it simpler to write and read the code.

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.

In task 1, two variables were made to hold an instance of each of the summary methods, if 50 different summary methods were required, then the program would require 50 new variables to hold an instance of each of them.

The changes made to the original code allows compatibility with only one variable holding an instance of the required summary instance.