Министерство науки и высшего образования Российской Федерации
Севастопольский государственный университет
Кафедра ИС

Отчет
по лабораторной работе №3
«Исследование способов модульного тестирования программного обеспечения»
по дисциплине
«ТЕСТИРОВАНИЕ ПО»

Выполнил студент группы ИС/б-17-2-о
Горбенко К. Н.
Проверил
Тлуховская Н.П.

Севастополь
2019

1 ЦЕЛЬ РАБОТЫ

Исследовать основные подходы к модульному тестированию программного обеспечения. Приобрести практические навыки составления модульных тестов для объектно-ориентированных программ.

2 ПОСТАНОВКА ЗАДАЧИ

Выполнить модульное тестиование одного из классов, созданных в ходе ЛР № 1. Для этого:

1. Создать спецификацию тестового случая для одного из методов выбранного класса.

2. Реализовать тестируемый класс и необходимое тестовое окружение на языке C#.

3. Выполнить тестирование с выводом результатов на экран и сохранением в log-файл.

4. Проанализировать результаты тестирования, сделать вывод.

3 ХОД РАБОТЫ

Для модульного тестирования выберем класс ExtensionMethods:

```
1 public static class ExtensionMethods
2 {
3     public static IEnumerable<int[]> GetColumns(this int[,] array)
4     {
5         if (array == null)
6         {
7             throw new ArgumentNullException($"{nameof(array)} instance was null
                  ");
8         }
9
10        for (var j = 0; j < array.GetLength(1); j++)
11        {
12            var column = new int[array.GetLength(0)];
13
14            for (var i = 0; i < array.GetLength(0); i++)
15                column[i] = array[i, j];
16
17            yield return column;
18        }
```

```
19     }
20 }
```

Для реализации тестов создадим класс, представляющий один тестовый случай:

```
1 public class Test
2 {
3     public string Name                  { get; set; }
4     public int[,] Input                 { get; set; }
5     public IEnumerable<int[]> Expected { get; set; }
6 }
```

Теперь создадим тестовое окружение:

```
1 public class TestEngine
2 {
3     private ILogger           Logger    { get; set; }
4     private IEnumerable<Test> TestSuite { get; set; }
5
6     public TestEngine(ILogger logger, IEnumerable<Test> testSuite)
7     {
8         Logger    = logger    ?? throw new ArgumentNullException("Logger
              instance was null");
9         TestSuite = testSuite ?? throw new ArgumentNullException("Test suite
              was null");
10    }
11
12    public void TestEquality()
13    {
14        foreach (var test in TestSuite)
15        {
16            try
17            {
18                var actual = test.Input.GetColumns();
19                var result = actual.SequenceEqual(test.Expected, new
                      Int32ArrayEqualityComparer());
20
21                Logger.Log($"Test {test.Name}," +
22                        $"result:{result}" +
23                        $"{(result ? "" : $",expected: {PrintArrayCollection(
                              test.Expected)} actual:{PrintArrayCollection(actual
                              )}")}");
24            }
25            catch (Exception e)
26            {
27                Logger.Log($"Test {test.Name}," +
28                        $"result:{false}," +
29                        $"Exception: {e.Message}");
```

```
30              }
31          }
32      }
33
34      private string PrintArrayCollection(IEnumerable<int[]> collection)
35      {
36          var stringBuilder = new StringBuilder("{ ");
37
38          foreach (var array in collection)
39          {
40              stringBuilder.Append("[ ");
41              foreach (var item in array)
42              {
43                  stringBuilder.Append($"{item} ");
44              }
45              stringBuilder.Append("] ");
46          }
47          stringBuilder.Append("}");
48          return stringBuilder.ToString();
49      }
50  }
```

В этом классе:

1. Содержится поле, содержащее экземпляр файлового логгера.

2. Содержится коллекция экземпляров тестов, представляющая набор тестовых случаев данного класса.

3. Содержится метод, выполняющий непосредственно тестирование согласно спецификации, предоставленной набором тестовых случаев. При этом сравнивается ожидаемое значение метода с текущим.

4. При провале теста для текущего тестового случая выводится ожидаемое значение результата выполнения метода и текущее значение для создания возможности проследить ошибки.

Класс файл-логгера:

```
1  public interface ILogger
2  {
3      void Log(string message);
4  }
5
6  public class FileLogger : ILogger
7  {
8      private readonly StreamWriter streamWriter;
9
10     public FileLogger(StreamWriter writer)
```

```
11      {
12          streamWriter = writer ?? throw new ArgumentNullException("StreamWriter
                instance was null");
13      }
14
15      public void Log(string message)
16      {
17          if (message != null && !string.IsNullOrWhiteSpace(message))
18          {
19              streamWriter.WriteLine(message);
20              streamWriter.Flush();
21          }
22      }
23  }
```

Этот класс не позволяет записывать пустые сообщения в файл.

Основная программа, запускающая тестовое окружение:

```
 1  public static class Program
 2  {
 3      static void Main(string[] args)
 4      {
 5          var fileLogger = new FileLogger(new StreamWriter(path, append:true));
 6          var testEngine = new TestEngine(fileLogger, TestSuite);
 7          testEngine.TestEquality();
 8      }
 9
10      private static string path = "file.txt";
11
12      private static IEnumerable<Test> TestSuite = new[]
13      {
14          new Test {
15              Name     = "Single item",
16              Input    = new[,] { { 3 } },
17              Expected = new[] { new[] { 3 } }.AsEnumerable()
18          },
19          new Test {
20              Name = "Multiple items in square matrix",
21              Input = new[,]
22              {
23                  { 15, 10, 36 },
24                  { 23,  0, 14 },
25                  { 62, 15, 35 }
26              },
27              Expected = new[]
28              {
29                  new[] { 15, 23, 62 },
30                  new[] { 10,  0, 15 },
```

```
31                  new[] { 36, 14, 35 }
32              }.AsEnumerable()
33          },
34          new Test {
35              Name = "Multiple items in non-square matrix",
36              Input = new[,]
37              {
38                  { 13,  2, 14, 16,  46 },
39                  {  0, 54, -2,  9, -14 },
40                  { 62,  0, 21, 15,   2 },
41                  { 64,  0, 11,  2, 523 }
42              },
43              Expected = new[]
44              {
45                  new[] { 13,   0, 62,  64 },
46                  new[] {  2,  54, -2,   9 },
47                  new[] { 14,  -2, 21,  11 },
48                  new[] { 16,   9, 15,   2 },
49                  new[] { 46, -14,  2, 523 }
50              }.AsEnumerable()
51          },
52      };
53 }
```

В этом классе происходит создание экземпляров логгера и тестового окружения и запуск тестового окружения. При этом, класс соддержит коллекцию тестов, которую передает в тестовое окружение.

Результат работы программы(содержимое файла). В программе специально один из тестовых примеров провален для демонстрации:

```
1 Test Single item,result:True
2 Test Multiple items in square matrix,result:True
3 Test Multiple items in non-square matrix,result:False,expected: { [ 13 0 62 64
    ] [ 2 54 -2 9 ] [ 14 -2 21 11 ] [ 16 9 15 2 ] [ 46 -14 2 523 ] } actual:{ [
     13 0 62 64 ] [ 2 54 0 0 ] [ 14 -2 21 11 ] [ 16 9 15 2 ] [ 46 -14 2 523 ] }
```

4 ВЫВОДЫ

В ходе лабораторной работы было проведено модульное тестирование класса, включающее в себя создание тестового окружения. Результат тестирования был записан в файл. Преимуществом данного метода является возможность автоматизации процесса тестирования ПО. Недостатком является сложность реализации, при которой для каждого класса необходимо создавать тестовое окружение.

Кроме того, при таком подходе существуют особенные сложности тестирования исключительных случаев.