Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий

Кафедра ИС

# ОТЧЕТ

по лабораторной работе №3

ИССЛЕДОВАНИЕ СПОСОБОВ МОДУЛЬНОГО ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Выполнил:

ст. гр. ИС/б-21-2-о

Мовенко К. М.

Проверил:

Собченко М.В.

Севастополь

2023

# ЦЕЛЬ РАБОТЫ

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

# ЗАДАНИЕ

* 1. Выбрать в качестве тестируемого один из классов, спроектированных в лабораторных работах №1-2;
  2. Составить спецификацию тестового случая для одного из методов выбранного класса;
  3. Реализовать тестируемый класс и необходимое тестовое окружение на языке С#;
  4. Выполнить тестирование с выводом результатов на экран и сохранением в log-файл;
  5. Проанализировать результаты тестирования, сделать выводы;

# ХОД РАБОТЫ

Поскольку при написании программы, использованной в лабораторных работах №1-2, не был использован объектно-ориентированный подход, было принято решение написать новый класс для последующего тестирования – класс StringTransformer со статическим методом Transform, принимающим на вход строку и меняющий все её символы, идущие после символа ‘#’, на символ ‘@’.

public class StringTransformer

{

// Метод, заменяющий все символы, идущие после #, на @

public static string Transform(string str)

{

string transformedStr = str;

bool foundHash = false;

for (int i = 0; i < transformedStr.Length; i++)

{

if (transformedStr[i] == '#')

{

foundHash = true;

}

else if (foundHash)

{

transformedStr = transformedStr.Substring(0, i) + '@' + transformedStr.Substring(i + 1);

}

}

return transformedStr;

}

}

Далее был расписан механизм тестирования класса. Тестовый драйвер содержит список кортежей, представляющих собой пары из тестируемой строки и ожидаемого результата. Для указанного метода класса было рассмотрено пять случаев:

1. Срока не пуста, символ ‘#’ расположен в середине;
2. Строка не пуста, символ ‘#’ является первым;
3. Строка не пуста, символ ‘#’ является последним;
4. Строка не пуста, символ ‘#’ отсутствует;
5. Строка пуста;

Далее идёт цикл, вызывающий метод Transform для каждой тестируемой строки и сравнивающий возвращённый результат с ожидаемым. Для записи результатов тестирования в log-файл был написан метод WriteLog.

public class Program

{

// Сохранение результатов тестирования в log-файл

private static void WriteLog(string message, string filename)

{

using (StreamWriter file = new StreamWriter(filename, true))

{

file.WriteLine(message);

}

}

public static void Main()

{

// Наборы тестовых примеров и ожидаемых результатов

string[][] testCases = new string[][]

{

new string[] {"Hello, #world!", "Hello, #@@@@@@"},

new string[] {"#dvjnsialw", "#@@@@@@@@@"},

new string[] {"dvjnsialw#", "dvjnsialw#"},

new string[] {"no hashes", "no hashes"},

new string[] {"", ""}

};

// Тестирование функции на примерах

for (int i = 0; i < testCases.Length; i++)

{

string[] testCase = testCases[i];

string str = testCase[0];

string expected = testCase[1];

string result = StringTransformer.Transform(str);

string message = (result == expected) ? $"Тест {i + 1} пройден успешно" : $"Тест {i + 1} не пройден";

Console.WriteLine(message);

WriteLog(message, "test.log");

}

}

}

После запуска программы был проверен log-файл, содержащий результаты тестирования. Судя по его содержимому, метод transform выдал верный результат для всех тестируемых примеров (Рисунок 1). На основании тестирования можно судить о верной работе класса StringTransform.

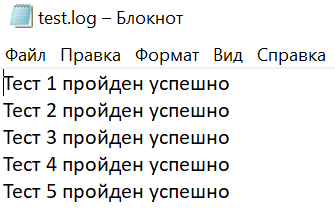


Рисунок 1 – Вывод результатов тестирования в файл.

# ВЫВОД

В ходе выполнения лабораторной работы был изучен модульный подход к тестированию ПО. Были получены навыки составления модульных тестов для программных модулей на примере классов объектно-ориентированной программы.