МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе №6

на тему: «Разработка собственного тестового драйвера»

по дисциплине: «Качество и тестирование программного обеспечения»

Вариант 7

Выполнила: Карпикова С.П.

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71ПГ

Проверили: Ужаринский А.Ю., Конюхова О.В.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2020 г.

**Задание:** необходимо разработать программу тестового драйвера для тестирования программных модулей. На вход данная программа получает путь к исходному коду тестируемого модуля. Также драйвер на вход получает файл с описанием тестов, которые нужно выполнить. Описание теста включает в себя название функции, параметры, которые должны быть переданы в функцию и ожидаемый результат. Структура файла с тестами может быть произвольной. Результатом работы тестового драйвера должен быть отчёт о результатах тестирования. Отчёт должен содержать следующую информацию: номер теста, полученный результат, ожидаемый результат, результат прохождения теста (success или faile). Если результат прохождения теста faile, то нужно вывести причину ошибки. Драйвер тестирования должен быть максимально универсальным. Для языков программирования, позволяющих выполнять код в runtime нужно чтобы драйвер работал с разными модулями без перекомпиляции. Если это не возможно, то сделать так, чтобы для тестирования нового программного кода требовалось минимум изменений в коде драйвера тестирования (в идеале в отдельном заголовочном файле изменить путь к файлу и возможно список вызовов методов).

**Ход работы**

В результате выполнения работы был разработан универсальный тестовый драйвер с пользовательским и интерфейсом на языке программирования c#.

Для того, чтобы воспользоваться тестовым драйвером, его проект необходимо добавить в решение в MS Visual Studio. Для этого необходимо кликнуть правой кнопкой мыши на Решение -> Добавить-> Существующий проект и выбрать проект тестового драйвера.

На рисунке 1 показано главное окно программы. Оно содержит 3 области: область создания теста, дерево тестов, результаты выполнения тестов.

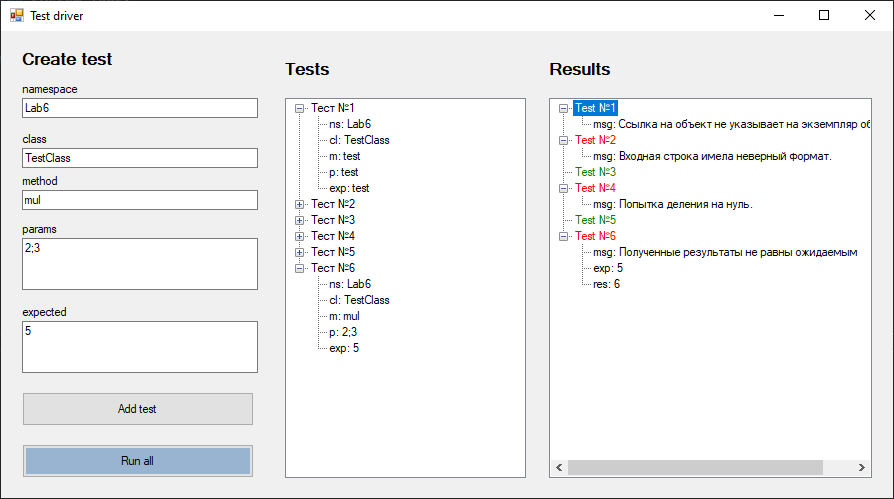


Рисунок 1 – Тестовый драйвер

В дереве тестов можно получить подробную информацию о созданном тесте (название класса, методов, параметров, ожидаемого результата и т.д.). В дереве результатов выполнения зеленым цветом подсвечены успешно пройденные тесты. Красным цветом подсвечены не пройденные тесты. В деталях не пройденного теста можно получить информацию о типе ошибки или исключения.

**Программа на C#:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab6

{

public class SimpleTest {

public SimpleTest(string \_namespace, string \_class, string \_method, string \_params, string \_expected) {

Namespace = \_namespace;

Class = \_class;

Method = \_method;

Params = \_params;

Expected = \_expected;

}

public string Namespace { get; set; }

public string Class { get; set; }

public string Method { get; set; }

public string Params { get; set; }

public string Expected { get; set; }

}

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Reflection;

using EnvDTE80;

using EnvDTE;

using System.Collections;

using System.IO;

using System.Runtime.InteropServices;

using Microsoft.Win32;

using VSLangProj;

using Microsoft.Build;

namespace Lab6

{

public partial class Form1 : Form

{

public List<SimpleTest> allTests;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

allTests = new List<SimpleTest>();

}

private void btnAddTestClick(object sender, EventArgs e)

{

var test = new SimpleTest(Namespace.Text, Class.Text, Method.Text, Params.Text, Expected.Text);

allTests.Add(test);

TreeNode testNode = new TreeNode("Тест №" + allTests.Count.ToString());

testNode.Nodes.Add(new TreeNode("ns: " + test.Namespace));

testNode.Nodes.Add(new TreeNode("cl: " + test.Class));

testNode.Nodes.Add(new TreeNode("m: " + test.Method));

testNode.Nodes.Add(new TreeNode("p: " + test.Params));

testNode.Nodes.Add(new TreeNode("exp: " + test.Expected));

tests.Nodes.Add(testNode);

}

private void btnRunTestsClick(object sender, EventArgs e)

{

results.Nodes.Clear();

var testNum = 1;

foreach (var test in allTests) {

TreeNode testNode = new TreeNode("Test №" + testNum.ToString());

try

{

var myObj = Activator.CreateInstance(Type.GetType(test.Namespace + "." + test.Class));

MethodInfo m = myObj.GetType().GetMethod(test.Method);

MethodBase Mymethodbase = myObj.GetType().GetMethod(test.Method);

ParameterInfo[] Myarray = Mymethodbase.GetParameters();

string[] methodParams = test.Params.Split(';');

object[] methodParamsWithTypes = new object[methodParams.Length];

foreach (ParameterInfo Myparam in Myarray)

{

methodParamsWithTypes[Myparam.Position] = Convert.ChangeType(methodParams[Myparam.Position], Myparam.ParameterType);

}

var res = m.Invoke(myObj, test.Params.Length > 0 ? methodParamsWithTypes : null);

var expected = Convert.ChangeType(test.Expected, m.ReturnType);

if (res.Equals(expected))

{

testNode.ForeColor = Color.Green;

results.Nodes.Add(testNode);

}

else

{

testNode.ForeColor = Color.Red;

testNode.Nodes.Add(new TreeNode("msg: Полученные результаты не равны ожидаемым"));

testNode.Nodes.Add(new TreeNode("exp: " + test.Expected));

testNode.Nodes.Add(new TreeNode("res: " + res.ToString()));

results.Nodes.Add(testNode);

}

}

catch (Exception ex)

{

testNode.ForeColor = Color.Red;

if (ex.InnerException != null)

{

testNode.Nodes.Add(new TreeNode("msg: " + ex.InnerException.Message));

}

else {

testNode.Nodes.Add(new TreeNode("msg: " + ex.Message));

}

results.Nodes.Add(testNode);

}

testNum++;

}

}

}

}