МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Курский государственный университет»

кафедра программного обеспечения и администрирования информационных систем

-

Отчёт
по контрольной работе № 2

«Модульное тестирование»

по дисциплине

«Верификация программного обеспечения**»**

Выполнил: Студент группы 413
Мусонда Салиму

Проверил: к.т.н., доцент
кафедры ПОиАИС
Макаров К.С.

Курск, 2020

Цель: изучение механизмов создания и функционирования тестового окружения.

Задачи:

1) Изучить основные понятия Лекций 5-8;

2) Составить тест-требования и провести модульное тестирование (согласно варианту) одного из методов, представленных в разделе **Задания**.

В качестве примера тестируемого модуля используется класс AnalaizerClass.cs, исходный текст которого, а также программная библиотека функций CalcClass.dll, необходимая для компиляции, находятся в каталоге с данной контрольной работой.

В отчёте к лабораторной в качестве экспериментальных результатов представить тест-требования к тестируемому методу, текст программы и экранную форму разработанного тестового драйвера с результатами тестирования.

Задачи и цели модульного тестирования

Каждая сложная программная система состоит из отдельных частей – модулей, выполняющих ту или иную функцию в составе системы. Для того, чтобы удостовериться в корректной работе системы в целом, необходимо вначале протестировать каждый модуль системы в отдельности. В случае возникновения проблем это позволит проще выявить модули, вызвавшие проблему, и устранить соответствующие дефекты в них. Такое тестирование модулей по отдельности получило называние модульного тестирования (unit testing).

Для каждого модуля, подвергаемого тестированию, разрабатывается тестовое окружение, включающее в себя драйвер и заглушки, готовятся тест-требования и тест-планы, описывающие конкретные тестовые примеры.

Основная **цель модульного тестирования** – удостовериться в соответствии требованиям каждого отдельного модуля системы перед тем, как будет произведена его интеграция в состав системы.

При этом в ходе модульного тестирования решаются четыре основные **задачи**.

1 Поиск и документирование несоответствий требованиям – это классическая задача тестирования, включающая в себя не только разработку тестового окружения и тестовых примеров, но и выполнение тестов, протоколирование результатов выполнения, составление отчетов о проблемах.

2 Поддержка разработки и рефакторинга низкоуровневой архитектуры системы и межмодульного взаимодействия – эта задача больше свойственна «легким» методологиям типа XP, где применяется принцип тестирования перед разработкой (Test-driven development), при котором основным источником требований для программного модуля является тест, написанный до самого модуля. Однако, даже при классической схеме тестирования модульные тесты могут выявить проблемы в дизайне системы и нелогичные или запутанные механизмы работы с модулем.

3 Поддержка рефакторинга модулей – эта задача связана с поддержкой процесса изменения системы. Достаточно часто в ходе разработки требуется проводить рефакторинг модулей или их групп – оптимизацию или полную переделку программного кода с целью повышения его сопровождаемости, скорости работы или надежности. Модульные тесты при этом являются мощным инструментом Сщдля проверки того, что новый вариант программного кода работает в точности так же, как и старый.

4 Поддержка устранения дефектов и отладки — эта задача сопряжена с обратной связью, которую получают разработчики от тестировщиков в виде отчетов о проблемах. Подробные отчеты о проблемах, составленные на этапе модульного тестирования, позволяют локализовать и устранить многие дефекты в программной системе на ранних стадиях ее разработки или разработки ее новой функциональности.

В силу того, что модули, подвергаемые тестированию, обычно невелики по размеру, модульное тестирование считается наиболее простым (хотя и достаточно трудоемким) этапом тестирования системы. Однако, несмотря на внешнюю простоту, с модульным тестированием связано две **проблемы**.

1 Не существует единых принципов определения того, что в точности является отдельным модулем.

2 Различия в трактовке самого понятия модульного тестирования – понимается ли под ним обособленное тестирование модуля, работа которого поддерживается только тестовым окружением, или речь идет о проверке корректности работы модуля в составе уже разработанной системы. В последнее время термин «модульное тестирование» чаще используется во втором смысле, хотя в этом случае речь скорее идет об интеграционном тестировании.

Понятие модуля и его границ. Тестирование классов

Традиционное определение модуля с точки зрения его тестирования: «**модуль** – это компонент минимального размера, который может быть независимо протестирован в ходе верификации программной системы». В реальности часто возникают проблемы с тем, что считать модулем. Существует несколько подходов к данному вопросу:

- модуль – это часть программного кода, выполняющая одну функцию с точки зрения функциональных требований;
- модуль – это программный модуль, то есть минимальный компилируемый элемент программной системы;
- модуль – это задача в списке задач проекта (с точки зрения его менеджера);
- модуль – это участок кода, который может уместиться на одном экране или одном листе бумаги;

– модуль – это один класс или их множество с единым интерфейсом.

Обычно за тестируемый модуль принимается либо программный модуль (единица компиляции) в случае, если система разрабатывается на процедурном языке программирования, либо класс, если система разрабатывается на объектно-ориентированном языке.

В случае систем, написанных на процедурных языках, процесс тестирования модуля заключается в разработке для каждого модуля тестового драйвера, вызывающего функции модуля и собирающего результаты их работы, и набора заглушек, которые имитируют поведение функций, содержащихся в других модулях и не попадающих под тестирование данного модуля. При тестировании объектно-ориентированных систем существует ряд особенностей, прежде всего вызванных инкапсуляцией данных и методов в классах.

В случае объектно-ориентированных систем более мелкое деление классов и использование отдельных методов в качестве тестируемых модулей нецелесообразно в связи с тем, что для тестирования каждого метода потребуется разработка тестового окружения, сравнимого по сложности с уже написанным программным кодом класса. Кроме того, декомпозиция класса нарушает принцип инкапсуляции, согласно которому объекты каждого класса должны вести себя как единое целое с точки зрения других объектов.

Процесс тестирования классов как модулей иногда называют **компонентным тестированием**. В ходе такого тестирование проверяется взаимодействие методов внутри класса и правильность доступа методов к внутренним данным класса. При таком тестировании возможно обнаружение не только стандартных дефектов, связанных с выходами за границы диапазона или неверно реализованными требованиями, а также обнаружение специфических дефектов объектно-ориентированного программного обеспечения:

– дефектов **инкапсуляции**, в результате которых, например, сокрытые данные класса оказывается недоступными при помощи соответствующих публичных методов;

– дефектов **наследования**, при наличии которых схема наследования блокирует важные данные или методы от классов-потомков;

– дефектов **полиморфизма**, при которых полиморфное поведение класса оказывается распространенным не на все возможные классы;

– дефектов **инстанцирования**, при которых во вновь создаваемых объектах класса не устанавливаются корректные значения по умолчанию параметров и внутренних данных класса.

Модульное тестирование применительно к системе «Калькулятор»

Рассмотренный в предыдущей контрольной работе пример прост прежде всего за счет того, что нам не приходится создавать тестового окружения. Чтобы увидеть весь описанный механизм в действии, протестируем метод RunEstimate класса AnalaizerClass. Этот метод использует методы из класса CalcClass, в надежности которых мы не уверены. Заменим эти методы заглушкой, состоящей исключительно из функций стандартного класса Math. Для этого воспользуемся файлом My.dll и добавим его в проект.

Составление тест-требований было изучено в рамках контрольной работы № 1.

Продемонстрируем, как создать тестовое окружение и запустить метод. Проверяем операцию сложения на примере 2+2, то есть в стеке до начала выполнения самой операции (то есть после компиляции) находятся следующие элементы: «2», «2», «+»

Листинг программы, реализующей проверку данных тест-требований имеет следующий вид:

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Test3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonStart_Click(object sender, EventArgs e)
        {
            System.CodeDom.Compiler.CodeDomProvider prov =
System.CodeDom.Compiler.CodeDomProvider.CreateProvider("CSharp");
            System.CodeDom.Compiler.CompilerParameters cmpparam = new
System.CodeDom.Compiler.CompilerParameters();
            cmpparam.GenerateExecutable = false;
            cmpparam.IncludeDebugInformation = false;
            cmpparam.ReferencedAssemblies.Add(Application.StartupPath +
"\\CalcClass.dll");
            cmpparam.ReferencedAssemblies.Add("System.dll");
            cmpparam.ReferencedAssemblies.Add("System.Windows.Forms.dll");
            cmpparam.OutputAssembly = "My.dll";
            System.CodeDom.Compiler.CompilerResults res =
prov.CompileAssemblyFromFile(cmpparam, Application.StartupPath +
"\\AnalaizerClass.cs");
            if (res.Errors.Count != 0)
            {
                richTextBox1.Text += res.Errors[0].ToString();
            }
            else
            {
                System.IO.BinaryReader reader = new
System.IO.BinaryReader(new System.IO.FileStream(Application.StartupPath +
"\\My.dll",System.IO.FileMode.Open, System.IO.FileAccess.Read));
                Byte[] asmBytes = new Byte[reader.BaseStream.Length];
                reader.Read(asmBytes,0,(Int32)reader.BaseStream.Length);
                reader.Close();
                reader = null;
                System.Reflection.Assembly assm =
System.Reflection.Assembly.Load(asmBytes);
                Type[] types = assm.GetTypes();
                Type analaizer = types[0];
                System.Reflection.MethodInfo addinfo =
analaizer.GetMethod("RunEstimate");
                System.Reflection.FieldInfo fieldopz =
analaizer.GetField("opz");
                System.Collections.ArrayList ar = new
System.Collections.ArrayList();
                ar.Add("2");
                ar.Add("2");
                ar.Add("+");
                fieldopz.SetValue(null,ar);
                richTextBox1.Text += addinfo.Invoke(null,null).ToString();
                asmBytes = null;
            }
            prov.Dispose();
        }
```

```
    }
}
```

Задания

Составить тест-требования и провести модульное тестирование следующих методов:

2

// <summary>

/* Форматирует входное выражение, выставляя между операторами пробелы и удаляя лишние, а также отлавливает неопознанные операторы, следит за концом строки а также отлавливает ошибки на конце строки */

// </summary>

/* <returns>конечную строку или сообщение об ошибке, начинающиеся со спец. символа &</returns> */

public static string Format()

Экспериментальные результаты

Тест-требования:

Функция Format должна вставлять пробелы между символами, удалять лишние, фиксировать наличие несуществующих операторов, повтора

операторов (кроме унарного плюса и минуса) и превышение максимального числа символов.

Текст программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApp2
{

    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {


            richTextBox1.Text = "";


            System.CodeDom.Compiler.CodeDomProvider prov =
System.CodeDom.Compiler.CodeDomProvider.CreateProvider("CSharp");
            System.CodeDom.Compiler.CompilerParameters cmpparam = new
System.CodeDom.Compiler.CompilerParameters();
            cmpparam.GenerateExecutable = false;
            cmpparam.IncludeDebugInformation = false;
            cmpparam.ReferencedAssemblies.Add(Application.StartupPath +
"\\CalcClass.dll");
            cmpparam.ReferencedAssemblies.Add("System.dll");
            cmpparam.ReferencedAssemblies.Add("System.Windows.Forms.dll");
            cmpparam.OutputAssembly = "My.dll";
            System.CodeDom.Compiler.CompilerResults res =
prov.CompileAssemblyFromFile(cmpparam, Application.StartupPath + "\\AnalaizerClass.cs");
            if (res.Errors.Count != 0)
            {
                richTextBox1.Text += res.Errors[0].ToString();
            }
            else
            {
                System.IO.BinaryReader reader = new System.IO.BinaryReader(new
System.IO.FileStream(Application.StartupPath + "\\My.dll", System.IO.FileMode.Open,
System.IO.FileAccess.Read));
                Byte[] asmBytes = new Byte[reader.BaseStream.Length];
                reader.Read(asmBytes, 0, (Int32)reader.BaseStream.Length);
                reader.Close();
                reader = null;
                System.Reflection.Assembly assm =
System.Reflection.Assembly.Load(asmBytes);
                Type[] types = assm.GetTypes();
                Type analaizer = types[0];
```

```
System.Reflection.MethodInfo addinfo = analaizer.GetMethod("Format");

System.Reflection.FieldInfo fieldopz = analaizer.GetField("expression");


string h = "Пример 1:\n";


string u = "1 -    7   +4";

h += "Входные данные: " + u + "\n";


h += "Ожидаемые результаты: 1 - 7 + 4\n";


fieldopz.SetValue(null, u);



string k = (string)addinfo.Invoke(null, null);

h += "Результаты: " + k + "\n";

richTextBox1.Text += h;

if (k == "1 - 7 + 4 ")

    richTextBox1.Text += "Тест пройден\n";
else richTextBox1.Text += "Тест не пройден\n";




h = "Пример 2:\n";


u = "1 -    7   +4";

h += "Входные данные: " + u + "\n";


h += "Ожидаемые результаты: 1 - 7 + 4\n";


fieldopz.SetValue(null, u);



k = (string)addinfo.Invoke(null, null);

h += "Результаты: " + k + "\n";

richTextBox1.Text += h;

if (k == "1 - 7 + 4 ")

    richTextBox1.Text += "Тест пройден\n";
```

```
else richTextBox1.Text += "Тест не пройден\n";



h = "Пример 3:\n";


u = "    7*9mod3      ";

h += "Входные данные: " + u + "\n";


h += "Ожидаемые результаты: 1 - 7 + 4\n";


fieldopz.SetValue(null, u);



k = (string)addinfo.Invoke(null, null);

h += "Результаты: " + k + "\n";

richTextBox1.Text += h;

if (k == "7 * 9 mod 3 ")

    richTextBox1.Text += "Тест пройден\n";
else richTextBox1.Text += "Тест не пройден\n";



h = "Пример 4:\n";


u = " 2+(9*3)- ";

h += "Входные данные: " + u + "\n";


h += "Ожидаемые результаты: &Error 05\n";


fieldopz.SetValue(null, u);



k = (string)addinfo.Invoke(null, null);

h += "Результаты: " + k + "\n";

richTextBox1.Text += h;

if (k == "&Error 05")

    richTextBox1.Text += "Тест пройден\n";
else richTextBox1.Text += "Тест не пройден\n";
```

```
h = "Пример 5:\n";

u = " 15(3+7%3) ";
h += "Входные данные: " + u + "\n";

h += "Ожидаемые результаты: &Error 02 at 8\n";

fieldopz.SetValue(null, u);

k = (string)addinfo.Invoke(null, null);
h += "Результаты: " + k + "\n";
richTextBox1.Text += h;
if (k == "&Error 02 at 8")
    richTextBox1.Text += "Тест пройден\n";
else richTextBox1.Text += "Тест не пройден\n";

h = "Пример 6:\n";

u = " p7*(m9+7/3)-8mod 7 ";
h += "Входные данные: " + u + "\n";

h += "Ожидаемые результаты: p 7 * ( m 9 + 7 / 3 ) - 8 mod 7\n";

fieldopz.SetValue(null, u);

k = (string)addinfo.Invoke(null, null);
h += "Результаты: " + k + "\n";
richTextBox1.Text += h;
if (k == "p 7 * ( m 9 + 7 / 3 ) - 8 mod 7 ")
    richTextBox1.Text += "Тест пройден\n";
else richTextBox1.Text += "Тест не пройден\n";
```

```
h = "Пример 7:\n";
u = "";
for (int i = 0; i < 65535; i++)

    u += "1";


h += "Входные данные: " + u + "\n";



h += "Ожидаемые результаты:" + u + "\n";


fieldopz.SetValue(null, u);



k = (string)addinfo.Invoke(null, null);

h += "Результаты: " + k + "\n";

richTextBox1.Text += h;


if (k == u + " ")

    richTextBox1.Text += "Тест пройден\n";
else richTextBox1.Text += "Тест не пройден\n";







h = "Пример 8:\n";
u = "";
for (int i = 0; i < 65536; i++)

    u += "1";


h += "Входные данные: " + u + "\n";



h += "Ожидаемые результаты:" + u + " " + "\n";


fieldopz.SetValue(null, u);



k = (string)addinfo.Invoke(null, null);

h += "Результаты: " + k + "\n";
```

```
richTextBox1.Text += h;


if (k == u + " ")

    richTextBox1.Text += "Тест пройден\n";
else richTextBox1.Text += "Тест не пройден\n";







h = "Пример 9:\n";
u = "";
for (int i = 0; i < 65537; i++)

    u += "1";


h += "Входные данные: " + u + "\n";



h += "Ожидаемые результаты: Error 07" + "\n";


fieldopz.SetValue(null, u);




k = (string)addinfo.Invoke(null, null);

h += "Результаты: " + k + "\n";

richTextBox1.Text += h;


if (k == "&Error 07")

    richTextBox1.Text += "Тест пройден\n";
else richTextBox1.Text += "Тест не пройден\n";







h = "Пример 10:\n";
u = "";
for (int i = 0; i < 65536; i++)

    u += "1";


h += "Входные данные: " + u + "\n";



h += "Ожидаемые результаты:" + u + "\n";
```

```
fieldopz.SetValue(null, u);



k = (string)addinfo.Invoke(null, null);

h += "Результаты: " + k + "\n";

richTextBox1.Text += h;


if (k == u + " ")

    richTextBox1.Text += "Тест пройден\n";
else richTextBox1.Text += "Тест не пройден\n";







h = "Пример 11:\n";
u = "pp8";



h += "Входные данные: " + u + "\n";



h += "Ожидаемые результаты: p p 8" + "\n";


fieldopz.SetValue(null, u);



k = (string)addinfo.Invoke(null, null);

h += "Результаты: " + k + "\n";

richTextBox1.Text += h;


if (k == "p p 8")

    richTextBox1.Text += "Тест пройден\n";
else richTextBox1.Text += "Тест не пройден\n";







h = "Пример 12:\n";
u = "mm8";



h += "Входные данные: " + u + "\n";
```

```
                h += "Ожидаемые результаты: m m 8" + u + "\n";


                fieldopz.SetValue(null, u);



                k = (string)addinfo.Invoke(null, null);

                h += "Результаты: " + k + "\n";

                richTextBox1.Text += h;


                if (k == "m m 8")

                    richTextBox1.Text += "Тест пройден\n";
                else richTextBox1.Text += "Тест не пройден\n";









                asmBytes = null;









            }


            prov.Dispose();
        }

    }
}
```

Тестирование

Тестирование функции Format представлено на рисунке 1.

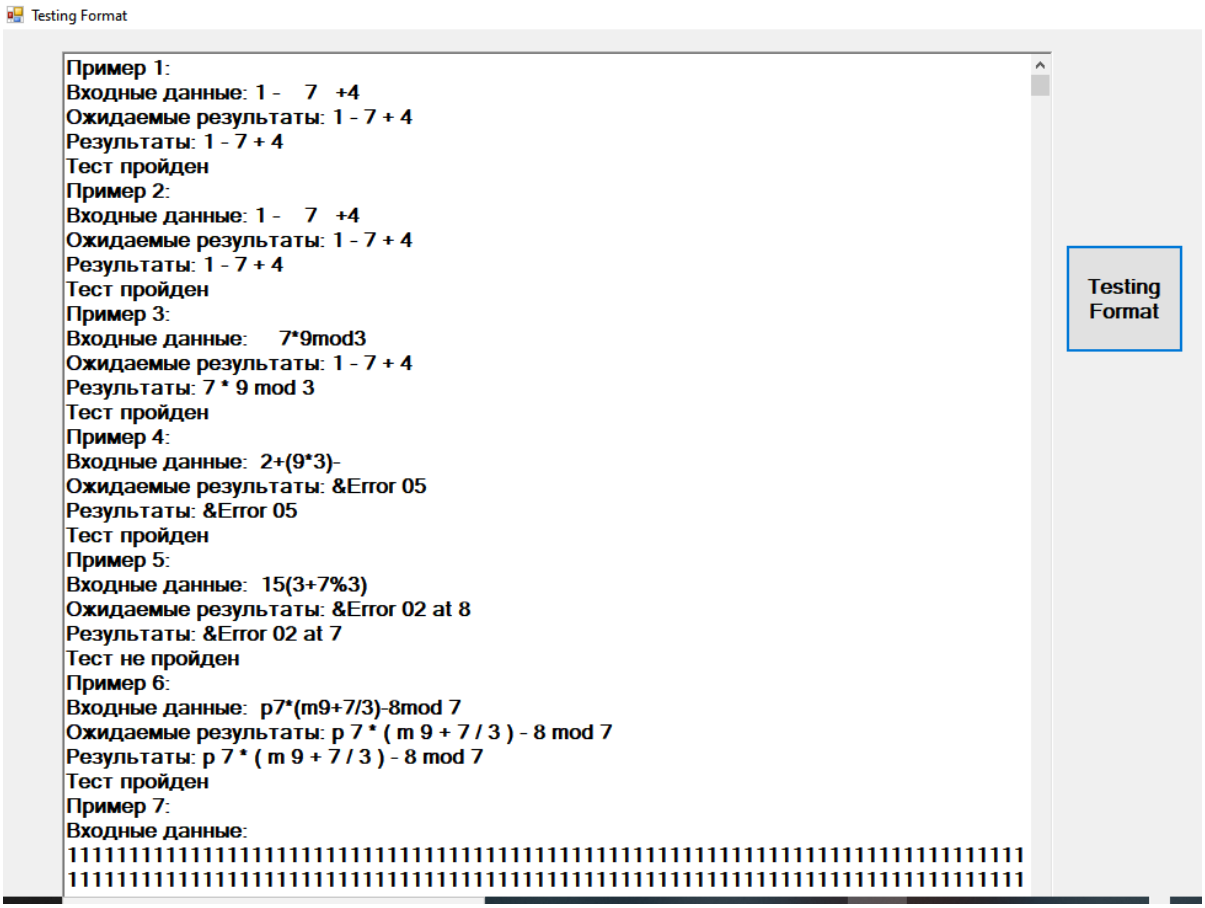

Рисунок 1 – Тестирование функции Format