# Федеральное государственное автономное образовательное учреждение высшего образования «СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

## Институт Космических и информационных технологий институт Кафедра «Информатика» кафедра

### ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №6

<u>Генерация блочных тестов через "контрактное программирование"</u>

Преподаватель		А.С.Кузнецов
	подпись, дата	инициалы, фамилия
Студент <u>КИ18-17/16 031830504</u>		Е.В. Железкин
номер группы, зачетной книжки	подпись, дата	инициалы, фамилия

#### 1 Цель работы

Изучить методологию генерации блочных тестов через "контрактное программирование".

#### 2 Задача работы

Для классов, спроектированных при выполнении любой из предыдущих работ, составить контрактную спецификацию. Согласно полученной спецификации протестировать систему на ранее написанных тестах. Написать отчет в форматах PDF, ODT или MarkDown в свободной форме, содержащий описание выполненных действий и полученные результаты.

#### Вариант 5

Битовый вектор (реализовать в виде массива, хранящего значения truefalse), при этом должны быть реализованы и протестированы следующие побитовые (поэлементные) логические операции:

- Сравнение
- И
- ИЛИ
- HE
- Импликация

#### 2.1 Инструкция по запуску

Необходимо установить .Net SDK:

• Страница загрузки для Windows:

https://dotnet.microsoft.com/download/dotnet/5.0

Для Linux:

```
$ sudo apt-get update; \
$ sudo apt-get install -y apt-transport-https && \
$ sudo apt-get update && \
$ sudo apt-get install -y dotnet-sdk-5.0
```

Далее на любой из двух систем выполнить в папке проекта:

\$ dotnet test

\$ dotnet test -v n (для более подробного вывода)

#### 3 Ход работы

Контрактный подход не поддерживается в sdk, установленных на моем пк, поэтому сначала пришлось откатить до .Net 4.8. Однако после некоторых манипуляций, статические методы класса *Contract* всё же начали выполняться в моем проекте. (Рисунки 1-3)

```
① Note

Code contracts aren't supported in .NET 5+ (including .NET Core versions). Consider using Nullable reference types instead.
```

Рисунок 1 – Оповещении в документации .net

```
Contract.Requires(firstArray.Length == 0);

if (firstArray.Length throw new Argument throw new Argument parameter BitArray firstArray

for (var i = 0; i < file first array of bits to compute is skipped. Compiler will not generate method invocation because the method is conditional, or it is partial method without implementation is skipped. Compiler will not generate method invocation because the method is conditional, or it is partial method without implementation is skipped. Compiler will not generate method invocation because the method is conditional, or it is partial method without implementation is skipped. Compiler will not generate method invocation because the method is conditional, or it is partial method without implementation is partial method without im
```

Рисунок 2 – Устаревший подход в экосистеме .net



Рисунок 3 – Установка пакета «Code Contracts for .Net»

Добавим необходимые предусловия в методы и уберём излишние проверки. (Рисунок 4)

```
public static BitArray ParseFromString(string input)
{
    Contract.Requires<NullReferenceException>(condition: input != null, userMessage: "input != null");
    Contract.Requires<NullReferenceException>(condition: input.Length > 0, userMessage: "input.Length > 0");

// if (input == null || input.Length == 0)
    // {
        throw new ArgumentException("Empty argument!");
        // }
```

Рисунок 4 – Добавление предусловий для тестируемых методов

Однако немного позже было обнаружено, что поведение методов отличается от ожидаемого, так как по умолчанию проект всё равно выполнялся на версии sdk .Net 5.0 с параметрами обратной совместимости (Рисунок 5), хотя в параметрах проекта был явно указан нужный sdk (Рисунок 6).

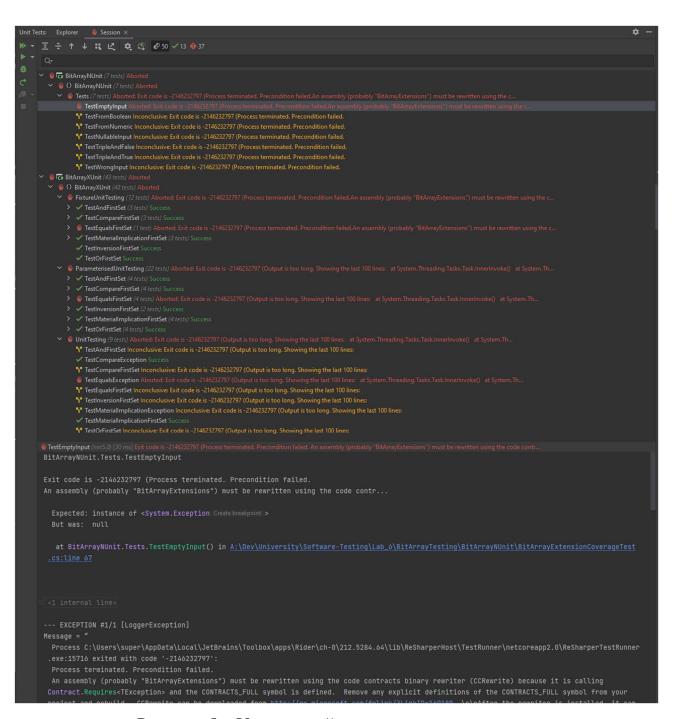


Рисунок 5 – Устаревший подход в экосистеме .net

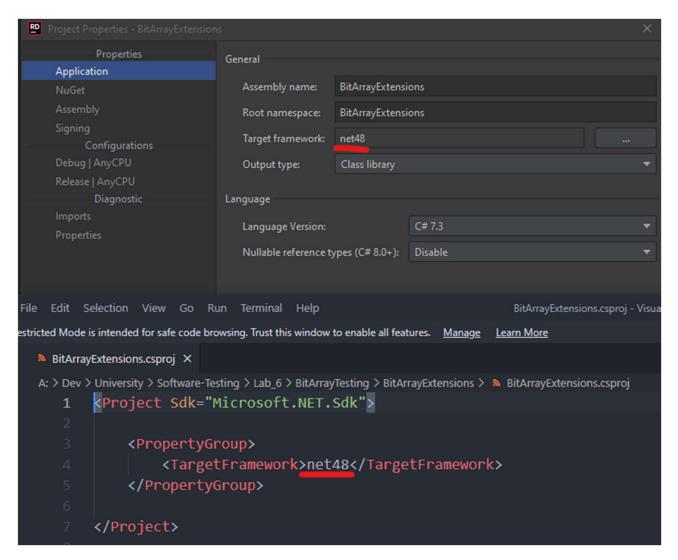


Рисунок 6 – Явное указание sdk в проекте

Поэтому было принято решение реализовать необходимые методы по их сигнатуре, описанной в базовой библиотеке (Рисунки 7-9).

Рисунок 7 – Явное указание sdk в проекте

Рисунок 8 – Реализация необходимой функциональности

```
Contract.Requires<NullReferenceException>(condition: firstArray.Length > 0, userMessage: "firstArray != null");

CustomContract.Requires<NullReferenceException>(condition: firstArray.Length > 0, userMessage: "firstArray != null");

// Contract.Require
// Contract.Require
in class CustomContract
// Contract.Require

TException is NullReferenceException

in class CustomContract

TException is NullReferenceException

in class CustomContract

in
```

Рисунок 9 – Кастомный вызов предусловия

Успешное прохождение тестов, в том числе и с пустыми входными параметрами. (рисунки 10-11)



Рисунок 10 – Итоговое тестирование методов

```
[Test]
public void TestWrongInput()
{
          Assert.Catch(code() => BitArrayExtension.ParseFromString(input "wrong input string"));
}

[Test]
public void TestNullableInput()
{
          Assert.Catch(code() => BitArrayExtension.ParseFromString(input null));
}

[Test]
public void TestEmptyInput()
{
          Assert.Catch(code() => BitArrayExtension.ParseFromString(input ""));
}
```

Рисунок 10 – Методы, вызывающие предусловия

#### 4 Вывод

В ходе данной лабораторной работы была изучена методология генерации блочных тестов через "контрактное программирование".