**18-КБ-ПР2 Буниф М.**

**Отчёт по лабораторной работе №4**

1. **Наименование и цель работы**

**Наименование:** Тестирование, управляемое данными (Data-Driven Unit Tests) и Анализ покрытия кода (Code Coverage)

**Цель работы:** Изучить подход к автоматизации процесса тестирования с помощью средств среды разработки Microsoft Visual Studio, а также понятие покрытия кода (Code Coverage) тестами и научиться использовать на практике средства автоматизации определения покрытия.

1. **Задание на лабораторную работу**

1) Используя класс из задания «Лабораторной работы №2. Тестирование методом черноного ящика», реализующий проверку принадлежности точки к области и тестовые наборы из той же работы создать проект тестирования (или добавить еще один тестовый класс в уже используемый тестовый проект) с набором тестов управляемых данными.

2) Запустить проект тестирования и проверить результаты работы.

3) Проанализировать покрытие кода тестами.

4) На основе тестов здания «Лабораторной работы №3. Тестирование методом белого ящика» проанализировать покрытие кода тестами.

5) Составить отчет о результатах проведенного тестирования.

**Текст программы на алгоритмическом языке**

**Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab2

{

public class CircleClass

{

public bool Coordinate(double x, double y, double R)

{

if (x <= 0 && y >= 0 && x >= (R \* (-1)) && y <= R && Math.Sqrt((x\*x) + (y\*y)) <= R ||

x >= 0 && y <= 0 && x >= R/2 && y >= (-2 \* x) ||

(R - x < R/2) && y <= ((2\*x) - (2\*R)))

return true;

else return false;

}

}

}

//lab3

namespace Lab3

{

public class StrClass

{

public string Split(string s, char c1, char c2)//1

{

var str = s.ToCharArray();//2

string ready = "";//3

for (int i = 0;//4

i < str.Length;//5

i++)//6

{

if (str[i] == c1)//7

str[i] = c2;//8

}

for (int i = 0;//9

i < str.Length;//10

i++)//11

ready += str[i];//12

return ready;//13

}

}

}

**Теsting classes**

using Lab2;

using NUnit.Framework;

namespace CircleTests

{

[TestFixture]

public class Class1

{

//AREA 1

//start point

[TestCase(0, 0, 5)]

//quarter II

[TestCase(-3, 1, 5)]

//quarter IV

[TestCase(3, -2, 5)]

//border btw quarters

[TestCase(0, 5, 5)]//btw I-II

[TestCase(-5, 0, 5)]//btw II-III

[TestCase(5, 0, 5)]//btw IV-I

public void TestPointArea1(double x, double y, double R)

{

bool expected = true;

CircleClass c = new CircleClass();

bool actual = c.Coordinate(x, y, R);

Assert.AreEqual(expected, actual);

}

//AREA 2

//quarter I

[TestCase(4, 1, 5)]

//quarter III

[TestCase(-4, -1, 5)]

[TestCase(0, -5, 5)]//btw III-IV

public void TestPointArea2(double x, double y, double R)

{

bool expected = false;

CircleClass c = new CircleClass();

bool actual = c.Coordinate(x, y, R);

Assert.AreEqual(expected, actual);

}

}

}

using Lab3;

using NUnit.Framework;

namespace StrTests

{

[TestFixture]

public class Class1

{

[Test()]

public void PossibleToSplit()

{

StrClass s = new StrClass();

string s1 = "The.Savage.Tiger.";

string expected = "The\nSavage\nTiger\n";

string actual = s.Split(s1, '.', '\n');

Assert.AreEqual(actual, expected);

}

[Test()]

public void ImpossibleToSplit()

{

StrClass s = new StrClass();

string s1 = "The.Savage.Tiger.";

string expected = "The.Savage.Tiger.";

string actual = s.Split(s1, ' ', '\n');

Assert.AreEqual(actual, expected);

}

}

}

**3. Результат проделанной работы**

