Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГТУ»)**

**Институт компьютерных систем и информационной безопасности**

**Кафедра информационных систем и программирования**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**Дисциплина: тестирование и отладка программного обеспечения**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Я.С Давков

Направление подготовки: 09.03.04 Программная инженерия

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Г. Волик

**1 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.

**2 Краткая теория**

При тестировании часто возникает ситуация, при которой сам тестовый код практически не отличается для различных тестовых методов, за исключением тестовых данных, передаваемых на вход тестируемого метода и результирующего значения. В таких случаях используется так называемое «Параметризованное тестирование», при котором наборы тестовых данных и результаты построчно группируются в таблицу, и каждая строка интерпретируется как отдельный тест. В среде Visual Studio также имеется возможность поддержки такого рода тестов, называемая «Тестирование, управляемое данными (DataDriven Unit Tests)», обладающая достаточно обширными возможностями. Тестирование, управляемое данными это тестирование, выполняемое для каждого варианта тестовых наборов. При этом источником данных могут быть значения, указанные в атрибуте TestCase или строки в таблице, указанной в качестве источника для теста в атрибуте TestCaseSource. Рассмотри оба варианта построения тестов. Для начала создадим демонстрационный проект содержащий классутилиту с методом, выполняющим сложения двух чисел (пример 5.1).

**3 Задание**

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

**4 Выполнение работы**

Создадим новый тестовый класс для DDT, будем использовать метод с TestCaseSource.

Листинг тестового класса:

[TestFixture]  
public class AreaCheckerTests  
{  
 private AreaChecker checker;  
  
 [SetUp]  
 public void Setup()  
 {  
 checker = new AreaChecker(4.0);  
 }  
  
 private static IEnumerable<TestCaseData> SemicircleTestCases  
 {  
 get  
 {  
 yield return new TestCaseData(0, 0, 1).SetName("Начало координат");  
 yield return new TestCaseData(0, 4, 1).SetName("Верхняя точка полукруга");  
 yield return new TestCaseData(4, 0, 1).SetName("Правая граница полукруга");  
 yield return new TestCaseData(-4, 0, 1).SetName("Левая граница полукруга");  
 yield return new TestCaseData(2.82842712, 2.82842712, 1).SetName("Диагональ 45° Точно");  
 yield return new TestCaseData(2, 2, 1).SetName("Внутри полукруга I квадрант");  
 yield return new TestCaseData(-2, 2, 1).SetName("Внутри полукруга II квадрант");  
 yield return new TestCaseData(3, 0, 1).SetName("На оси X внутри радиуса");  
 yield return new TestCaseData(-3, 0, 1).SetName("На оси X слева");  
 yield return new TestCaseData(0, 3, 1).SetName("На оси Y внутри радиуса");  
 }  
 }  
  
 [Test, TestCaseSource(nameof(SemicircleTestCases))]  
 public void TestPoint\_PointInSemicircle\_ReturnsOne(double x, double y, int expected)  
 {  
 int result = checker.TestPoint(x, y);  
 Assert.That(result, Is.EqualTo(expected));  
 }  
  
 private static IEnumerable<TestCaseData> TriangleTestCases  
 {  
 get  
 {  
 yield return new TestCaseData(0, -4, 2).SetName("Вершина треугольника ось Y");  
 yield return new TestCaseData(0, -2, 2).SetName("Правый катет");  
 yield return new TestCaseData(-2, -2, 2).SetName("Гипотенуза");  
 yield return new TestCaseData(-2, 0, 1).SetName("Верхний катет");  
 yield return new TestCaseData(-1, -1, 2).SetName("Внутри треугольника");  
 yield return new TestCaseData(-3, 0, 1).SetName("На верхнем катете");  
 yield return new TestCaseData(-0.5, -0.5, 2).SetName("Близко к началу координат");  
 }  
 }  
  
 [Test, TestCaseSource(nameof(TriangleTestCases))]  
 public void TestPoint\_PointInTriangle\_ReturnsTwo(double x, double y, int expected)  
 {  
 int result = checker.TestPoint(x, y);  
 Assert.That(result, Is.EqualTo(expected));  
 }  
  
 private static IEnumerable<TestCaseData> OutsideTestCases  
 {  
 get  
 {  
 yield return new TestCaseData(5, 0, 0).SetName("I квадрант вне");  
 yield return new TestCaseData(-5, 5, 0).SetName("II квадрант вне");  
 yield return new TestCaseData(3, 4, 0).SetName("Вне полукруга сверху");  
 yield return new TestCaseData(-5, -1, 0).SetName("III квадрант вне треугольника");  
 yield return new TestCaseData(4.1, 0, 0).SetName("За границей справа");  
 yield return new TestCaseData(2, -2, 0).SetName("IV квадрант положительный X");  
 }  
 }  
  
 [Test, TestCaseSource(nameof(OutsideTestCases))]  
 public void TestPoint\_PointOutsideAreas\_ReturnsZero(double x, double y, int expected)  
 {  
 int result = checker.TestPoint(x, y);  
 Assert.That(result, Is.EqualTo(expected));  
 }  
  
 [Test]  
 public void Constructor\_NegativeRadius\_ThrowsArgumentException()  
 {  
 Assert.Throws<ArgumentException>(() => new AreaChecker(-1));  
 }  
  
 [Test]  
 public void Constructor\_ZeroRadius\_ThrowsArgumentException()  
 {  
 Assert.Throws<ArgumentException>(() => new AreaChecker(0));  
 }  
}

Запустим dotCover и проверим прцент покрытия кода тестами.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 1 – результат dotCover

Теперь проанализируем покрытие кода тестами на основе задания из лабораторной работы 4

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 – результат dotCover