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 №6**

**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 тестированию методом серого ящика.

**2 Краткая теория**

Наиболее эффективная процедура тестирования заключается в том, чтобы разрабатывать тесты, используя стратегию черного ящика, а затем как необходимое условие – дополнительные тесты, используя методы белого ящика. Такой подход называют методом «серого ящика». Это подход, сочетающий элементы двух предыдущих подходов. С одной стороны, тестирование, ориентированное на пользователя, а значит, мы используем паттерны поведения пользователя, т.е. применяем методику «Черного ящика». А с другой стороны производим «информированное» тестирование, т.е. использование знаний о том, как устроена хотя бы часть тестируемого кода. Тестирование «черного ящика» может быть менее эффективным при поиске ошибок, связанных с потоком данных на уровне исходного кода. А тестирование «белого ящика» менее эффективно в нахождении высокоуровневых ошибок в операционной системе, а также ошибок совместимости. Тестирование же «серого ящика» используется для оценки проекта в рамках взаимодействия его индивидуальных компонентов. Тестирование «серого ящика» наиболее подходит для тестирования веб-приложений, потому что для него необходимы высокоуровневая разработка, операционное окружение и условия совместимости. Во время проведения анализа черного или белого ящика более сложно определить проблемы, связанные с непрерывным потоком данных. Специфические контекстные проблемы тестирования легче всего найти во время проверки серого ящика. В большинстве случаев целесообразнее использовать подход «Серого ящика» так как он позволяет достигать преимущества обоих способов тестирования.

**3 Задание**

1) Создать класс (в соответствии с вариантом задания из п.5), реализующий преобразование строки из одной системы счисления в другую. 2) Протестировать класс на основе метода серого ящика с использованием средств автоматизации. 3) Составить отчет о результатах проведенного тестирования.

**4 Выполнение работы**

Мой вариант – 4, исходное основание 10, целевое основание 12.

Листинг класса:

public class BaseConverter  
{  
 private const string *Digits* = "0123456789AB";  
  
 public string ConvertTo12Base(int number)  
 {  
 if (number < 0)  
 throw new ArgumentException("Число должно быть неотрицательным");  
  
 if (number == 0)  
 return "0";  
  
 StringBuilder result = new StringBuilder();  
  
 while (number > 0)  
 {  
 int remainder = number % 12;  
 result.Insert(0, *Digits*[remainder]);  
 number /= 12;  
 }  
  
 return result.ToString();  
 }  
}

Напишем тесты и проведем тестирование.

[TestFixture]  
public class BaseConverterTests  
{  
 private BaseConverter converter;  
  
 [SetUp]  
 public void Setup()  
 {  
 converter = new BaseConverter();  
 }  
  
 [Test]  
 [TestCase(0, "0")]  
 [TestCase(10, "A")]  
 [TestCase(11, "B")]  
 [TestCase(12, "10")]  
 [TestCase(14, "12")]  
 [TestCase(144, "100")]  
 [TestCase(173, "125")]  
 public void ConvertTo12Base\_ConvertsCorrectly(int input, string expected)  
 {  
 var result = converter.ConvertTo12Base(input);  
 Assert.That(result, Is.EqualTo(expected));  
 }  
  
 [Test]  
 public void ConvertTo12Base\_NegativeNumber\_ThrowsArgumentException()  
 {  
 Assert.Throws<ArgumentException>(() => converter.ConvertTo12Base(-5));  
 }  
  
 [Test]  
 public void ConvertTo12Base\_LargeNumber\_ReturnsExpectedFormat()  
 {  
 var result = converter.ConvertTo12Base(123456);  
 Assert.That(result, Does.Match("^[0-9AB]+$"));  
 }  
}

Запустим тесты

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 1 – результат запуска тестов