Лабораторная работа №5

Тестирование ПО методом «стеклянного ящика»

1 Цель работы

1.1 Тестирование программного кода по методологии белого ящика;

2 Литература

2.1 Фленов М. Е. Библия C#. — 5-е изд., перераб. и доп. / М. Е. Фленов — СПб.: БХВ-Петербург, 2022. — 464 с.

2.2 Куликов, С. С. Тестирование программного обеспечения. Базовый курс : практ. пособие. / С. С. Куликов. — Минск: Четыре четверти, 2020. — 294 с

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Создайте проект консольного приложения C# и скопируйте в него код из приложения п.9.1;

5.2 Составить тест-кейсы для созданного приложения по шаблону из приложения п.9.2;

5.3 Составить отчет по проделанной работе.

6 Порядок выполнения работы

6.1 Повторить теоретический материал п. 3.1;

6.2 Выполнить тестирование ПО п. 5.1-5.3;

6.3 Ответить на контрольные вопросы п. 8;

6.4 Заполнить отчет п. 7.

7 Содержание отчета

7.1 Титульный лист;

7.2 Цель работы;

7.3 Протокол тестирования;

7.4 Ответы на контрольные вопросы п. 6.3;

7.5 Вывод по проделанной работе.

8 Контрольные вопросы

8.1 Как минимизировать количество тест-кейсов при тестировании?

8.2 Каковы этапы тестирования по методу белого ящика?

9 Приложение

9.1 Код программы

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

class Program
{
    // Класс для представления книги
    class Book
    {
        public string Title { get; set; }
        public string Author { get; set; }

        public Book(string title, string author)
        {
            Title = title;
            Author = author;
        }

        public override string ToString()
        {
            return $"\"{Title}\" by {Author}";
        }
    }

    static void Main(string[] args)
    {
        List<Book> library = new List<Book>();
        bool running = true;

        while (running)
        {
            Console.WriteLine("\nБиблиотека книг");
            Console.WriteLine("1. Добавить книгу");
            Console.WriteLine("2. Удалить книгу");
            Console.WriteLine("3. Показать все книги");
            Console.WriteLine("4. Выход");
            Console.Write("Выберите действие: ");

            string choice = Console.ReadLine();

            switch (choice)
            {
                case "1":
                    AddBook(library);
                    break;
                case "2":
                    RemoveBook(library);
                    break;
                case "3":
                    ShowBooks(library);
                    break;
                case "4":
                    running = false;
                    break;
                default:
                    Console.WriteLine("Неверный выбор. Пожалуйста, попробуйте
снова.");
                    break;
            }
        }
    }
```

```
    static void AddBook(List<Book> library)
    {
        Console.Write("Введите название книги: ");
        string title = Console.ReadLine();
        Console.Write("Введите автора книги: ");
        string author = Console.ReadLine();

        library.Add(new Book(title, author));
        Console.WriteLine("Книга добавлена.");
    }

    static void RemoveBook(List<Book> library)
    {
        ShowBooks(library);

        if (library.Count == 0) return;

        Console.Write("Введите номер книги для удаления: ");
        if (int.TryParse(Console.ReadLine(), out int index) && index > 0 && index
<= library.Count)
        {
            library.RemoveAt(index - 1);
            Console.WriteLine("Книга удалена.");
        }
        else
        {
            Console.WriteLine("Неверный ввод.");
        }
    }

    static void ShowBooks(List<Book> library)
    {
        if (library.Count == 0)
        {
            Console.WriteLine("В библиотеке нет книг.");
        }
        else
        {
            Console.WriteLine("\nСписок книг в библиотеке:");
            for (int i = 0; i < library.Count; i++)
            {
                Console.WriteLine($"{i + 1}. {library[i]}");
            }
        }
    }
}
```

9.2 Шаблон

Шаблон тест-кейса для тестирования белого ящика

1. **Идентификатор теста**:
   Уникальный идентификатор тест-кейса (например, WB-001).
2. **Название теста**:
   Краткое описание, что именно тестируется.
3. **Цель теста**:
   Цель тестирования (например, проверить корректность выполнения ветвления в функции).

4. **Предусловия**:
   Условия, которые должны быть выполнены перед началом теста (например, определенные значения переменных, конфигурации и т.д.).
5. **Описание тестируемого метода (модуля)**:
   Указать имя функции, модуля или блока кода, который тестируется. Желательно приложить короткий фрагмент кода или ссылку на соответствующий участок.
6. **Шаги выполнения теста**:
   Подробное описание шагов для выполнения теста:
   - Входные данные.
   - Ожидаемые значения на каждом шаге выполнения.
7. **Ожидаемый результат**:
   Что должно произойти при выполнении теста, если код корректен.
8. **Фактический результат**:
   Что произошло в действительности (заполняется после выполнения теста).
9. **Примечания**:
   Дополнительная информация, которая может быть полезной (например, ссылки на документацию).
10. **Статус**:
    Пройден/Не пройден.

Пример заполненного тест-кейса

1. **Идентификатор теста**: WB-001
2. **Название теста**: Тестирование ветвления в методе расчета суммы заказа
3. **Цель теста**:
   Проверить корректность ветвления в методе `CalculateTotalOrderPrice()`, особенно в части применения скидок.
4. **Предусловия**:
   - Метод `CalculateTotalOrderPrice()` должен быть реализован.
   - Система скидок настроена на 10% скидку для заказов более чем на 1000 единиц.
5. **Описание тестируемого метода (модуля)**: Метод `CalculateTotalOrderPrice()` принимает список товаров и их количество, рассчитывает общую стоимость и применяет скидку, если сумма заказа превышает 1000.

   ```
   csharp
   Копировать код
   public decimal CalculateTotalOrderPrice(List<Product> products)
   {
       decimal total = products.Sum(p => p.Price * p.Quantity);
       if (total > 1000)
       {
           total *= 0.9M; // Применение скидки 10%
       }
       return total;
   }
   ```

6. **Шаги выполнения теста**:
   - Шаг 1: Передать в метод список товаров на общую сумму 1200.

- Шаг 2: Проверить, что метод корректно применяет скидку 10%, итоговая сумма должна быть 1080.
- Шаг 3: Передать в метод список товаров на сумму 800.
- Шаг 4: Проверить, что метод не применяет скидку, итоговая сумма должна остаться 800.

7. **Ожидаемый результат**:
   - При сумме заказа 1200 итоговая сумма должна быть 1080.
   - При сумме заказа 800 итоговая сумма должна остаться 800.
8. **Фактический результат**:
   - При сумме заказа 1200 итоговая сумма составила 1080 (тест пройден).
   - При сумме заказа 800 итоговая сумма составила 800 (тест пройден).
9. **Примечания**:
   Метод работает корректно, все ветвления отработали согласно ожиданиям.
10. **Статус**:
    Пройден.