Министерство образования и науки
Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра информатики, вычислительной техники и информационной безопасности

Третьяков А.А., Сучкова Л.И.

Методические указания к выполнению лабораторных работ по дисциплине «Программная инженерия»

Барнаул 2018

УДК 004.42

Третьяков, А.А., Сучкова, Л.И. Методические указания к выполнению лабораторных работ по дисциплине «Программная инженерия»/ А.А. Третьяков; АлтГТУ им. И.И. Ползунова. – Барнаул, АлтГТУ, 2018. –30с.

Методические указания предназначены для студентов, обучающихся по направлению подготовки бакалавров 09.03.01 «Информатика и вычислительная техника». Структура и содержание методических указаний соответствуют образовательному стандарту высшего образования по указанному направлению подготовки.

Рассмотрены и одобрены на заседании
кафедры информатики, вычислительной техники
и информационной безопасности.
Протокол № 2 от 27.09.2018 г.

Содержание

Лабораторная работа № 1.

Тема: Описание предметной области.

Цели и задачи работы: Выполнить описание предметной области в соответствии с выданным заданием.

Теоретические сведения о работе приведены в [1,2,3,4,7] и конспекте лекций.

Описание используемых средств для выполнения работы : текстовый редактор.

Методика выполнения работы:

1. Изучить теоретический материал по теме «Жизненный цикл проекта и его разновидности».
2. Выполнить задание к работе.

Требования к отчету:

Отчет должен содержать титульный лист и описание предметной области, соответствующей выбранному заданию.

Контрольные вопросы по лабораторной работе включают теоретические вопросы, предложенные преподавателем.

Индивидуальные задания:

1. Информационная система «Адвокатская контора»
2. Информационная система «Аренда помещений»
3. Информационная система «Библиотечный каталог»
4. Информационная система «Бюро технической инвентаризации»
5. Информационная система «Доставка пиццы»
6. Информационная система «Организация праздничных мероприятий»
7. Информационная система «Организация туристических поездок»
8. Информационная система «Риэлтерская контора»
9. Информационная система автомобильного завода
10. Информационная система аптеки
11. Информационная система аэропорта
12. Информационная система "Прокат автомобилей"
13. Информационная система городской телефонной сети
14. Информационная система гостиничного комплекса
15. Информационная система железнодорожной пассажирской станции
16. Информационная система магазина автозапчастей
17. Информационная система музея
18. Информационная система продуктовой сети
19. Информационная система ресторана
20. Информационная система спортивных организаций города

21. Информационная система строительной организации
22. Информационная система таксопарка
23. Информационная система транспортной компании
24. Информационная система учета документов в нотариальной конторе
25. Информационная система учета клиентов частного сыскного бюро
26. Информационная система "Доставка цветов"
27. Информационная система фотоцентра
28. Информационная система склада
29. Информационная система театра
30. Информационная система страховой компании

Лабораторная работа № 2.

Тема: Описание пользовательских и нефункциональных требований к программному продукту.

Цели и задачи работы: Изучение общих принципов управления требованиями к ПО. Изучение и разработка требований к программному продукту.

Теоретические сведения о работе приведены в [1,2,3,5,7] и конспекте лекций.

Задание к работе:

1. Выполнить описание пользовательских требований к программному продукту;
2. Выполнить описание нефункциональных требований к программному продукту.

Описание используемых средств для выполнения работы : текстовый редактор.

Методика выполнения работы:

1. Изучить теоретический материал по теме «Типы требований».
2. Выполнить задание к работе.

Требования к отчету:

Отчет должен содержать титульный лист, описание пользовательских требований и нефункциональных требований к программному продукту, соответствующему выбранной теме.

Контрольные вопросы по лабораторной работе включают теоретические вопросы, предложенные преподавателем.

Индивидуальные задания берутся из лабораторной № 1.

Лабораторная работа № 3.

Цели и задачи работы: Разработка логической модели системы и моделей ее вариантов использования.

Теоретические сведения о работе приведены в источниках [3,6] и в конспекте лекций по дисциплине.

Задание к работе:

1. Разработать модель вариантов использования системы (Use Case model);
2. Разработать один из видов диаграммы деятельностей экторов системы.
3. Разработать логическую или концептуальную (на выбор) модель системы.

Описание используемых средств для выполнения работы : текстовый редактор, программное обеспечение для построения диаграмм (Visual Paradigm Community Edition, Microsoft Office Visio). Для чернового варианта работы – бумага, карандаш, ластик.

Методика выполнения работы:

1. Изучить теоретический материал по теме «Унифицированный язык моделирования UML».
2. Выполнить задание к работе.

Требования к отчету:

Отчет должен содержать титульный лист и разработанные диаграммы UML, соответствующие индивидуальному заданию.

Контрольные вопросы по лабораторной работе включают теоретические вопросы, предложенные преподавателем.

Индивидуальные задания берутся из лабораторной № 1.

Справочная информация для выполнения работы:

Visual Paradigm Community Edition представляет собой case-средство визуального UML-моделирования. Средство предлагает объектноориентированный подход к анализу и проектированию систем различной сложности и позволяет создавать множество типов диаграмм в полностью визуализированной среде разработки посредством простых drag&drop операций.

При запуске программа предложит открыть существующий проект или выбрать тип вновь создаваемой диаграммы (Рисунок 3.1).

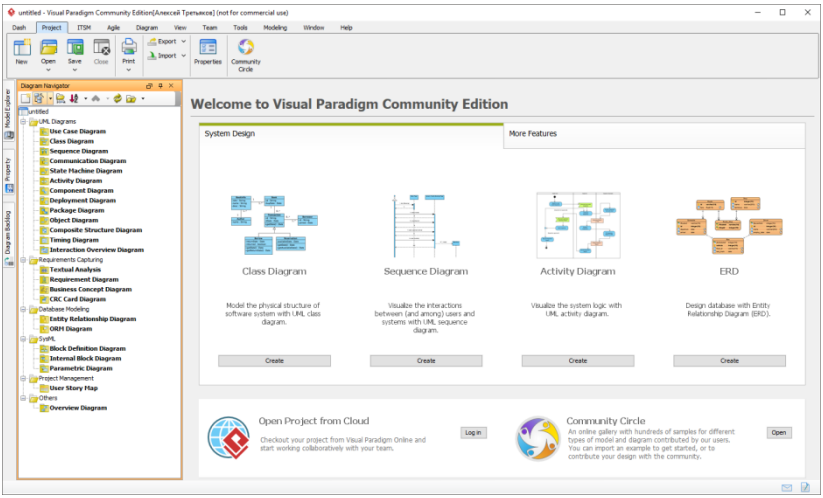

Рисунок 3.1 – Главное окноVisual Paradigm Community Edition

Рассмотрим подробнее некоторые особенности создания диаграмм в Visual Paradigm CE на примере модели вариантов использования (Use Case model).

Для создания новой диаграммы вариантов использования нужно выбрать соответствующую строку в списке вновь создаваемых диаграмм (в нашем случае– **New Use Case Diagram**) (Рисунок 3.2).

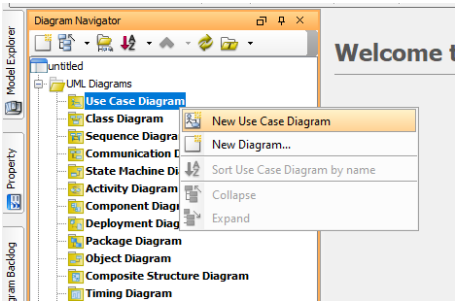

Рисунок 3.2 – Создание диаграммы вариантов использования

Для создания элемента следует выбрать его в панели объектов диаграммы щелчком мыши, затем, вторым кликом на области построения элемент может быть добавлен в диаграмму (Рисунок 3.3).

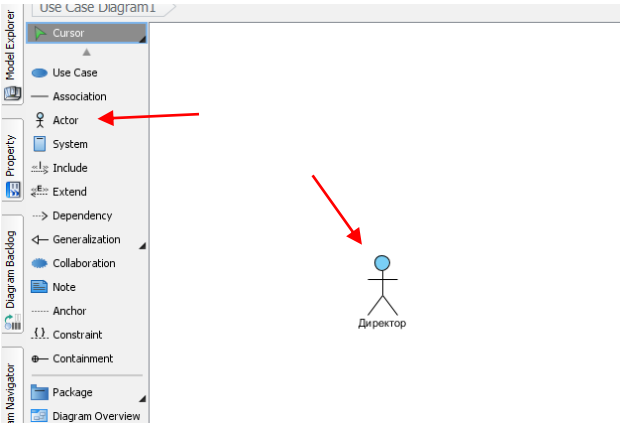

Рисунок 3.3. – Создание эктора

После создания очередного элемента, ему присваивается имя по умолчанию, которое может быть изменено сразу или позже, по двойному щелчку на элементе. Имя (и другие свойства) также можно задать в панели свойств объектов (Property) в левой части окна программы (Рисунок 3.4).

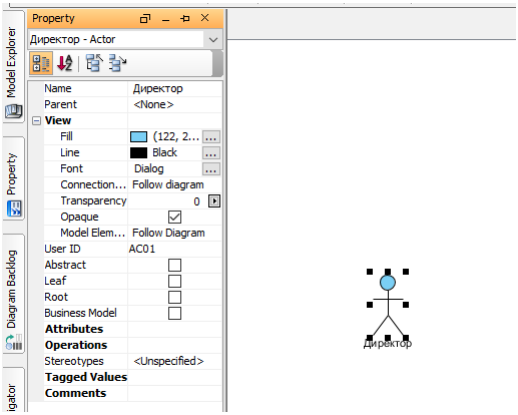

Рисунок 3.4 – Панель свойств объекта

Для создания новой связи выберите тип связи в панели объектов, затем щелкните на объекте-источнике, далее – на связываемом с ним объекте (Рисунок 3.5).

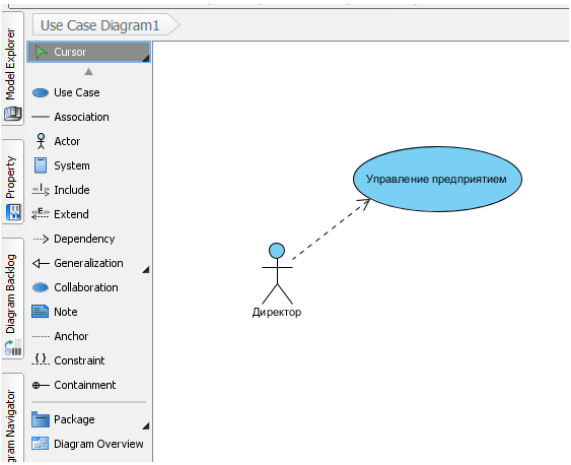

Рисунок 3.5 – Связь между объектами

При построении связей следует учитывать, что в Visual Paradigm CE имеется функция автоматической проверки синтаксиса UML, которая не позволяет создавать неверные связи между объектами. С помощью двойного клика по связи можно задать подпись к связи.

Кроме этого, для каждой связи можно установить собственный стиль. Для этого необходимо в контекстном меню связи выбрать **Styles and Formatting → Connector Style** и в появившемся списке выбрать один из предложенных типов (Рисунок 3.6). В этом же меню задается стиль отображения подписи связи.

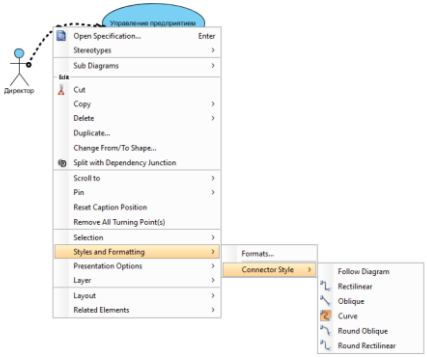

Рисунок 3.6 – Изменение стиля связи

Готовую диаграмму можно полностью или частично копировать в формате изображения в буфер обмена для экспорта в другое приложение.

Для этого следует выделить нужные объекты или всю диаграмму (**Ctrl+A**) и в меню **Diagram** выбрать подходящую Вам команду (Рисунок 3.7). После выполненных действий изображение диаграммы готово к вставке в другом приложении.

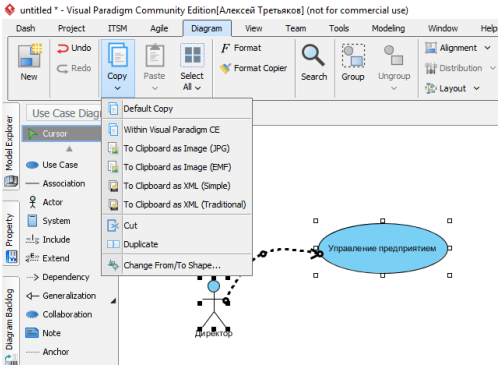

Рисунок 3.7 – Копирование диаграммы в буфер обмена

Лабораторная работа № 4.

Цели и задачи работы: Проектирование архитектуры программной системы.

Теоретические сведения о работе приведены в источниках [3,5,6] и в конспекте лекций по дисциплине.

Задание к работе:

1. Предложить варианты архитектуры программной системы и выбрать наиболее подходящий;
2. Выбрать язык программирования для разработки программной системы;
3. Выбрать наиболее подходящую технологию для хранения данных;
4. Разработать варианты графического интерфейса для программной системы.

Описание используемых средств для выполнения работы: среда разработки (Visual Studio, Eclipse, NetBeans), Microsoft Office Visio, Toad Data Modeler.

Методика выполнения работы:

1. Изучить теоретический материал по теме «Проектирование структур данных и архитектуры программных систем»;
2. Выполнить задание к работе.

Требования к отчету:

Отчет должен содержать титульный лист, описание выбранной архитектуры, языка программирования и выбранной технологии хранения данных, варианты графического интерфейса.

Контрольные вопросы по лабораторной работе включают теоретические вопросы, предложенные преподавателем.

Индивидуальные задания берутся из лабораторной № 1.

Лабораторная работа № 5.

Цели и задачи работы: Разработка диаграммы классов и модели базы данных.

Теоретические сведения о работе приведены в источнике [3,5,6] и в конспекте лекций по дисциплине.

Задание к работе:

1. Разработать диаграмму классов программной системы;
2. Разработать модель базы данных;
3. Сгенерировать скрипт базы данных из разработанной модели.

Описание используемых средств для выполнения работы: текстовый и графический редактор, среда разработки (Visual Studio, Eclipse, NetBeans), программное обеспечение для построения диаграмм (Visual Paradigm Community Edition, Microsoft Office Visio), Toad Data Modeler. Для чернового варианта работы – бумага, карандаш, ластик.

Методика выполнения работы:

1. Изучить теоретический материал по теме «Объекты, структуры данных и обработка ошибок при создании кода»;
2. Выполнить задание к работе.

Требования к отчету:

Отчет должен содержать титульный лист, диаграмму классов и модель базы данных программной системы, исходный код скрипта базы данных.

Контрольные вопросы по лабораторной работе включают теоретические вопросы, предложенные преподавателем.

Индивидуальные задания берутся из лабораторной № 1.

Лабораторная работа № 6.

Цели и задачи работы: Реализация части программной системы, соответствующей выбранным требованиям.
Теоретические сведения о работе приведены в источниках [3,5,6] и в конспекте лекций по дисциплине.
Задание к работе:
1. Реализовать часть программной системы с использованием языка программирования высокого уровня;
2. Выполнить отладку разработанной системы.

Описание используемых средств для выполнения работы :язык программирования высокого уровня (C, C++, C#, Java), среда разработки (Visual Studio, Eclipse, NetBeans).
Методика выполнения работы:
1. Изучить теоретический материал по теме «Принципы разработки стабильного программного обеспечения».
2. Выполнить задание к работе.

Требования к отчету:
Отчет должен содержать титульный лист, текст разработанного программного обеспечения, тесты.
Контрольные вопросы по лабораторной работе включают теоретические вопросы, предложенные преподавателем.
Индивидуальные задания берутся из лабораторной работы №1.

Лабораторная работа № 7.

Цели и задачи работы: Тестирование программного продукта.
Теоретические сведения о работе приведены в источниках [3,4,6] и в конспекте лекций по дисциплине.
Задание к работе: Разработать модульные тесты (Unit Tests) для разработанного программного продукта.
Описание используемых средств для выполнения работы: язык программирования высокого уровня (C, C++, C#, Java), среда разработки (Visual Studio, Eclipse, NetBeans), библиотека для модульного тестирования ПО (Visual Studio Unit Testing, NUnit, JUnit).
Методика выполнения работы:
1. Изучить теоретический материал по теме «Технологии тестирования».
2. Выполнить задание к работе.

Требования к отчету:

Отчет должен содержать титульный лист, исходный код разработанных тестов, результаты выполнения модульных тестов.
Контрольные вопросы по лабораторной работе включают теоретические вопросы, предложенные преподавателем.
Индивидуальные задания берутся из лабораторной №1.
Справочная информация для выполнения работы:

Разработаем тестируемый метод в языке программированияC# с помощью платформы для выполнения модульных тестов Microsoft. Можно легко адаптировать его для других языков, а также использовать другие тестовые платформы, например NUnit.

Создание теста

В качестве примера используем небольшой проект MathApp, содержащий класс Mathи пару методов: сложение и вычитание.
Создадим тестовый проект с помощью контекстного меню решения.

В диалоговом окне **Добавить новый проект** выберите **Visual C# → Тест**, а затем выберите **Проект модульного теста** (Рисунок 7.1).

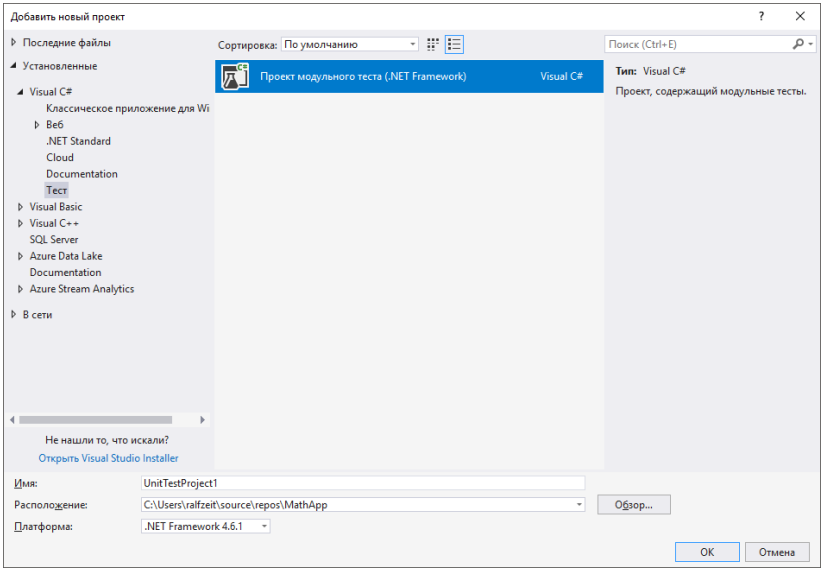

Рисунок 7.1 – Создание проекта модульного теста

Затем необходимо добавить в проект модульного теста ссылку на тестируемый проект. Для этого нужно кликнуть правой кнопкой мыши на проект модульного теста и выбрать **Добавить → Ссылка** (Рисунок 7.2) В открывшемся окне (Рисунок 7.3) отметим нужный проект.

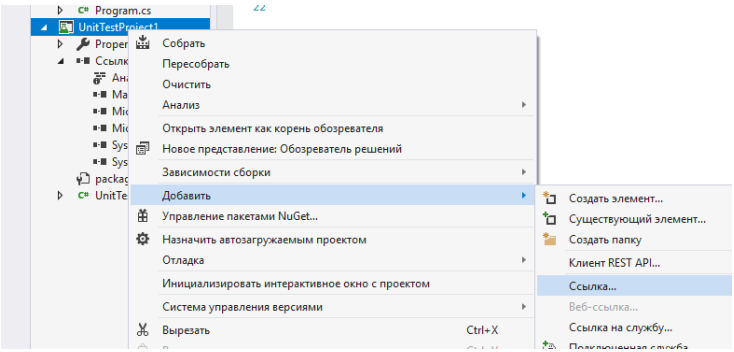

Рисунок 7.2 –Контекстное меню проекта тестирования

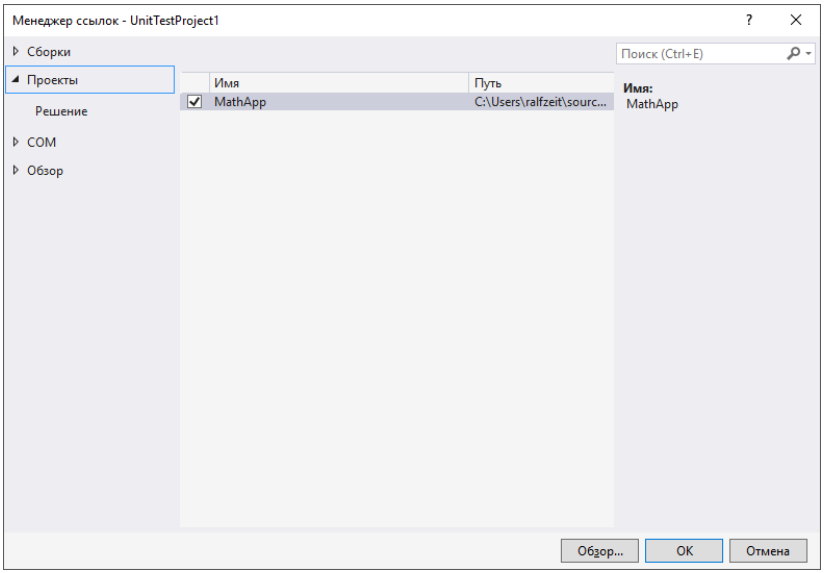

Рисунок 7.3 – Добавление ссылки на тестируемый проект

Создадим метод **MathAddTest** для проверки функции сложения двух целых чисел **Add** из класса **MathApp.Math**:

```
using System;
using MathApp;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void MathAddTest()
        {
            double expectedResult = 5;
            //Запуск функции
            double actualResult = MathApp.Math.Add(2,3);
            //Проверка результата
            Assert.AreEqual(expectedResult, actualResult,
               delta: expectedResult / 100);
        }
    }
}
```

Теперь запустим тестовый метод. Для этого перейдем в меню **Тест → Выполнить → Все тесты**. Проект будет пересобран и будет произведен запуск всех модульных тестов. Результаты тестирования будут отображены в **Обозревателе тестов** (Рисунок 7.4).

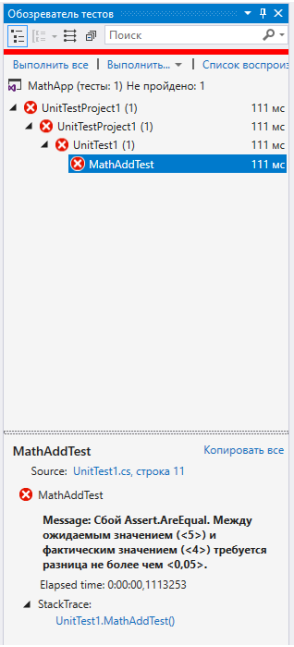

Рисунок 7.4 – Результаты тестирования

Как можно увидеть, тест не был пройден, т.к. фактический результат выполнения функции не совпал с заранее известным и правильным из-за ошибки в коде функции. Исправим ошибку в коде и запустим тест еще раз.

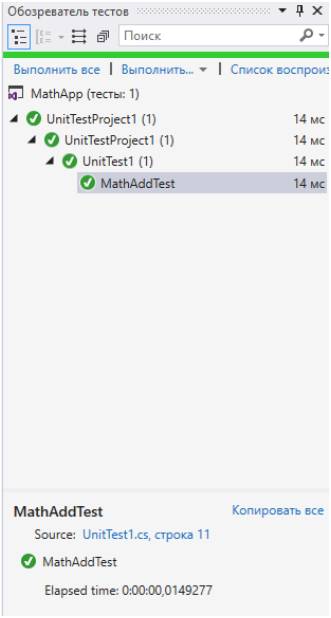

Рисунок 7.5 – Успешное прохождение тестов

Лабораторная работа № 8.

Цели и задачи работы: Документирование проекта.
Теоретические сведения о работе приведены в источниках [3,4,6] и в конспекте лекций по дисциплине.
Задание к работе:

1. Выполнить описание исходного кода программы с помощью xml-комментариев.
2. Выполнить автоматическую сборку документации для исходного кода проекта.

Описание используемых средств для выполнения работы : язык программирования высокого уровня (C, C++, C#, Java), среда разработки (Visual Studio, Eclipse, NetBeans).
Методика выполнения работы:

1. Изучить теоретический материал по теме «Документирование при разработке программных систем».
2. Изучить технологию автоматического документирования для разрабатываемого программного продукта.
3. Выполнить задание к работе.

Требования к отчету:
Отчет должен содержать титульный лист, описание процесса генерации и текст полученной документации к программному проекту.
Контрольные вопросы по лабораторной работе включают теоретические вопросы, предложенные преподавателем.
Индивидуальные задания берутся из лабораторной №1.
Справочная информация для выполнения работы:

Документирование кода с помощью XML-комментариев

Комментарии к XML-документации — это особый тип комментария, добавляемого перед определяемой пользователем частью кода. Их особенность в том, что компилятор может обрабатывать их для создания XML-файла документации во время компиляции. Созданный компилятором XML-файл можно распространять вместе со сборкой .NET, чтобы Visual Studio и другие интегрированные среды разработки могли использовать IntelliSense для отображения кратких сведений, например, о классах и функциях проекта. Кроме того, XML-файл можно запускать с помощью таких средств, как Sandcastle (https://github.com/EWSoftware/SHFB/releases), и создавать веб-сайты со справочными сведениями по API.

Для создания XML-документации в Visual Studio щелкните правой кнопкой мыши на проект и выберите **Свойства** (Рисунок 8.1). В диалоговом окне свойств откройте вкладку **Сборка** и поставьте флажок **XML-файл документации** (Рисунок 8.2). Можно также изменить расположение, в которое компилятор записывает файл.

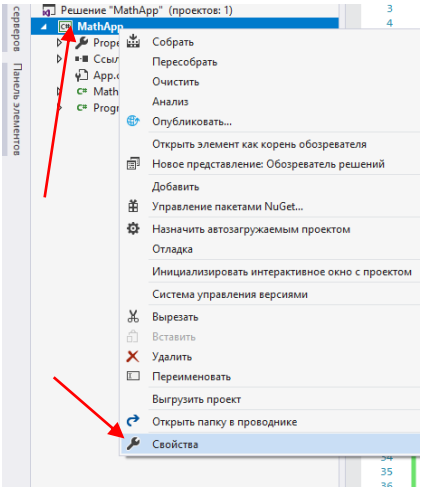

Рисунок 8.1 – Контекстное меню проекта

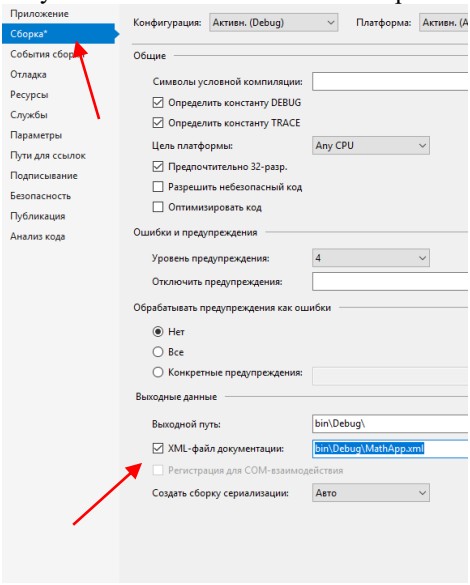

Рисунок 8.2 – Вкладка «Сборка»

Для вставки комментария к XML-документации используются три символа косой черты (///) и текст комментария в формате XML. Пример:

```
/// <summary>
/// Описание класса
/// </summary>
public class SomeClass
{

}
```

Пример выполнения

В качестве примера возьмем небольшой класс:

```
/*
Класс Math
Содержит некоторые методы для выполнения математических функций
*/
public class Math
{
        //Сложение двух целых чисел
        public static int Add(int a, int b)
        {
             if ((a == int.MaxValue && b > 0) ||
                (b == int.MaxValue && a > 0))
                 throw new System.OverflowException();
                return a + b;
         }

         //Разница двух целых чисел
         public static int Subtract(int a, int b)
        {
                return a - b;
        }
}
```

Теперь нам нужна возможность создания справочного документа по API из кода для сторонних разработчиков, которые используют библиотеку, но не имеют доступа к исходному коду. Как упоминалось ранее, это можно сделать с помощью тегов XML-документации. Рассмотрим основные теги XML-документации

Тег **<summary>** имеет очень важное значение, и его рекомендуется включать, так как его содержимое является основным источником информации о типе или члене в IntelliSense или справочном документе по API.

Тег **<remarks>** дополняет сведения о типах или членах, предоставляемые тегом **<summary>**. В этом примере вы просто добавите его в класс.

Тег **<returns>** описывает возвращаемое значение объявления метода.

Тег **<example>** применяется для включения примера в XML-документацию. В этом случае нужно использовать дочерний тег **<code>**.

С помощью тега **<exception>** можно сообщить разработчикам, что метод может генерировать определенные исключения. Атрибут **cref** представляет ссылку на исключение, которое доступно из текущей среды компиляции. Это может быть любой тип, определенный в проекте или ссылочной сборке. Компилятор выдаст предупреждение, если его значение не может быть разрешено.

Тег **<param>** используется для описания параметров метода.

Тег **<list>** используется для форматирования сведений о документации в виде упорядоченного списка, неупорядоченного списка или таблицы.

Упорядоченный список или таблицу можно сформировать, изменив атрибут **type** на **number** или **table** соответственно.

Пример полученного кода с XML-коментариями:

```
/// <summary>
/// Класс Math
/// Содержит некоторые методы для выполнения математических функций
/// <list type="bullet">
/// <item>
/// <term>Add</term>
/// <description>Операция сложения</description>
/// </item>
/// <item>
/// <term>Subtract</term>
/// <description>Операция вычитания</description>
/// </item>
/// </list>
/// </summary>
/// <remarks>
/// Этот класс может складывать и вычитать числа
/// </remarks>
public class Math
{
        /// <summary>
        /// Сложение двух целых чисел
        /// </summary>
        /// <returns>
        /// Возвращает сумму двух целых чисел
        /// </returns>
        /// <example>
```

```
        /// <code>
        /// int c = Math.Add(4, 5);
        /// if (c > 10)
        /// {
        ///     Console.WriteLine(c);
        /// }
        /// </code>
        /// </example>
        ///<exception cref="System.OverflowException">Возникает,
когда один из параметров равен максимальному значению int,
        /// а другой - больше нуля</exception>
        /// <param name="a">Целое число</param>
        /// <param name="b">Целое число</param>
        public static int Add(int a, int b)
        {
                if ((a == int.MaxValue && b > 0) ||
                        (b == int.MaxValue && a > 0))
                throw new System.OverflowException();
                return a + b;
        }
        /// <summary>
        /// Разница двух целых чисел
        /// </summary>
        /// <returns>
        /// Возвращает разницу двух целых чисел
        /// </returns>
        /// <example>
        /// <code>
        /// int c = Math.Subtract(10, 5);
        /// Console.WriteLine(c);
        /// </code>
        /// </example>
        /// <param name="a">Целоечисло</param>
        /// <param name="b">Целоечисло</param>
        public static int Subtract(int a, int b)
        {
                return a - b;
        }
}
```

Содержимое файла XML-документации:

```
<?xml version="1.0"?>
<doc>
<assembly>
<name>MathApp</name>
</assembly>
<members>
<member name="T:MathApp.Math">
<summary>
Класс Math
Содержит некоторые методы для выполнения математических функций
```

```
<list type="bullet">
<item>
<term>Add</term>
<description>Операция сложения</description>
</item>
<item>
<term>Subtract</term>
<description>Операция вычитания</description>
</item>
</list>
</summary>
<remarks>
Этот класс может складывать и вычитать числа
</remarks>
</member>
<member name="M:MathApp.Math.Add(System.Int32,System.Int32)">
<summary>
Сложение двух целых чисел
</summary>
<returns>
Возвращает сумму двух целых чисел
</returns>
<example>
<code>
        int c = Math.Add(4, 5);
        if (c > 10)
        {
                Console.WriteLine(c);
        }
</code>
</example>
<exception  cref="T:System.OverflowException">Возникает,  когда
один из параметров равен максимальному значению int,
а другой - больше нуля</exception>
<param name="a">Целое число</param>
<param name="b">Целое число</param>
</member>
<member
name="M:MathApp.Math.Subtract(System.Int32,System.Int32)">
<summary>
Разница двух целых чисел
</summary>
<returns>
Возвращает разницу двух целых чисел
</returns>
<example>
<code>
        int c = Math.Subtract(10, 5);
        Console.WriteLine(c);
</code>
</example>
<param name="a">Целое число</param>
<param name="b">Целое число</param>
```

```
</member>
</members>
</doc>
```

Лабораторная работа № 9.

Цели и задачи работы: Конфигурирование и протоколирование ПО.
Теоретические сведения о работе приведены в конспекте лекций по дисциплине.
Задание к работе:

1. Задать в файле конфигурации следующие настройки приложения: сервер базы данных, имя базы данных, логин и пароль.
2. Реализовать загрузку настроек из файла конфигурации.
3. Реализовать возможность изменения параметров в файле конфигурации через отдельную форму настроек в графическом интерфейсе программной системы.
4. С помощью профилировщика среды разработки выполнить оценку производительности программной системы и оптимизировать наименее производительные части системы.
5. Повторно выполнить оценку производительности и сравнить результаты.

Описание используемых средств для выполнения работы: язык программирования высокого уровня (C, C++, C#, Java), среда разработки (Visual Studio, Eclipse, NetBeans).
Методика выполнения работы:

1. Изучить теоретический материал по теме «Эксплуатация и сопровождение ПО. Конфигурирование и протоколирование».
2. Изучить возможности профилировщика в среде разработки (IDE).
3. Изучить структуру файлов конфигурации проекта и способы работы с ними.
4. Выполнить задание к работе.

Требования к отчету:
Отчет должен содержать титульный лист, диаграммы с разметкой для интерпретации, перечень семантических подпрограмм интерпретатора с их краткой характеристикой.
Контрольные вопросы по лабораторной работе включают теоретические вопросы, предложенные преподавателем.
Индивидуальные задания берутся из лабораторной №1.
Краткие сведения для выполнения работы:

Средства профилирования Visual Studio

Средства профилирования Visual Studio можно использовать для анализа проблем с производительностью в приложении. Для запуска профилировщика в Visual Studio выберите пункт меню **Отладка → Профилировщик производительности** (Рисунок 9.1).

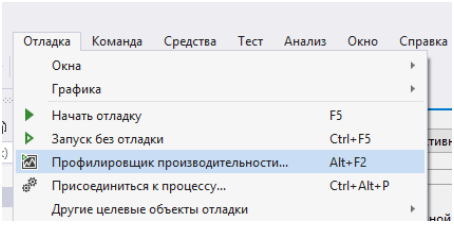

Рисунок 9.1 – Меню «Отладка»

На первой странице мастера выберите необходимый инструмент и нажмите **Начать** (Рисунок 9.2).

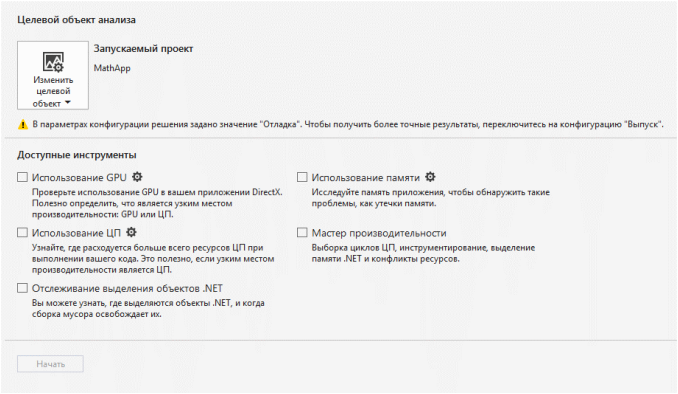

Рисунок 9.2 –Выбор инструмента профилирования

Когда приложение завершит выполнение, откроется отчет профилирования. Рассмотрим функционал профилировщика на примере отчета по использованию ЦП.

На рисунке ниже показано окно профилировщика с результатами, где можно видеть ветви в стеке вызовов, на выполнение которых было затрачено больше всего времени, и функции с наибольшим количеством исключительных попаданий:

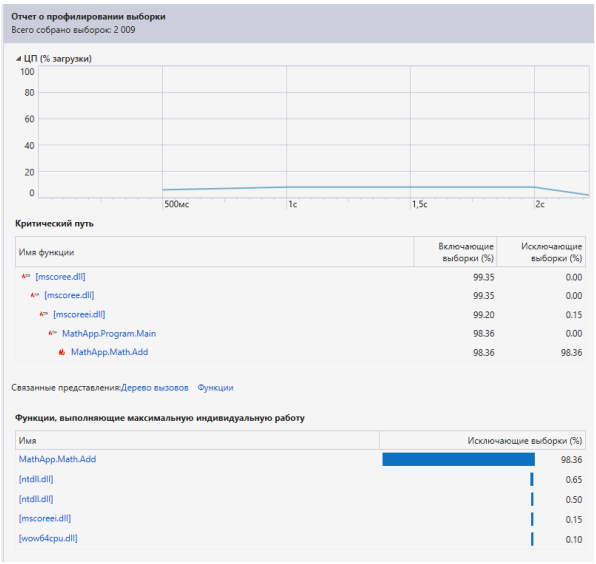

Рисунок 9.3 – Окно с результатами работы профилировщика

На следующем рисунке показан отчет, где перечислено несколько методов, на выполнение которых затрачено больше всего процессорного времени.

Рисунок 9.4 – Функции с наибольшим временем выполнения

Можно заметить, что функция **MathApp.Math::Add** занимает большую часть процессорного времени. Если выполнить двойной клик на методе в списке, откроется окно детализации со строками исходного кода в приложении, в которых обнаружено наибольшее число попаданий:

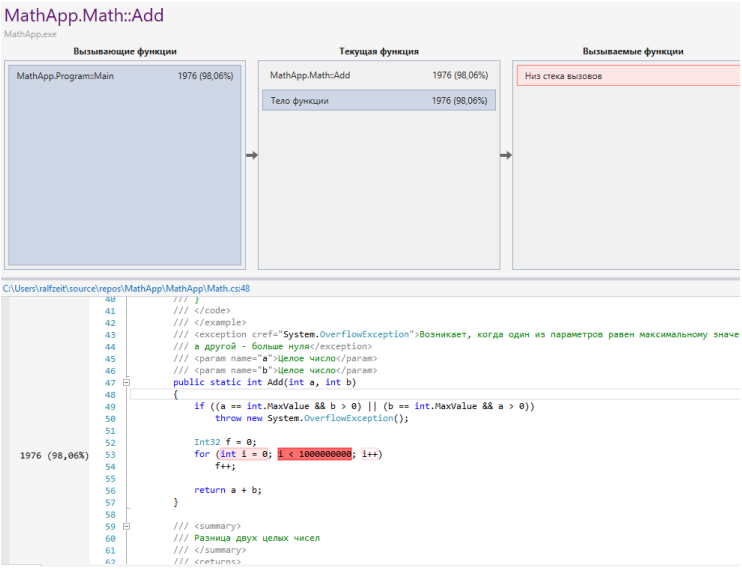

Рисунок 9.5 – Детализация

Как можно увидеть, в функции Add есть бесполезная часть кода, которая занимает неоправданно много процессорного времени. Убрав данные строки кода из функции, мы сократим время выполнения данной функции и всей программы в целом.

Конфигурационные файлы

Рассмотрим простейший конфигурационный файл на примере Visual Studio и языка программирования C#. По умолчанию в проекте он отображается как **App.config**.

Пример содержимого App.config:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
        <startup>
                <supportedRuntime version="v4.0"
                sku=".NETFramework,Version=v4.6.1" />
        </startup>
        <appSettings>
                <add key="server" value="SERVER_NAME" />
                <add key="database" value="DATABASE_NAME" />
        </appSettings>
</configuration>
```

Для работы с файлом конфигурации необходимо импортировать библиотеку **System.Configuration** в проект. Для этого кликните правой кнопкой мыши на проект и выберите **Добавить → Ссылка**. В открывшемся окне выберите System.Configuration (Рисунок 9.6).

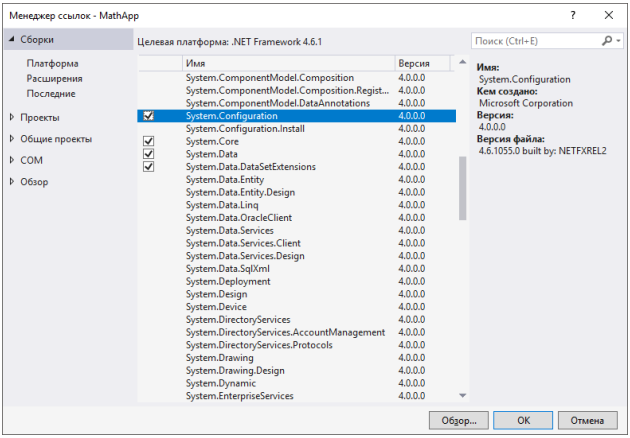

Рисунок 9.6 – Менеджер ссылок

Затем объявите библиотеку в коде программы с помощью:

```
using System.Configuration;
```

Теперь Вы можете работать с файлом конфигурации напрямую из программы.

Пример работы с файлом конфигурации:

```
_serverTextBox.Text =
       ConfigurationManager.AppSettings["server"];
```

Редактирование файла конфигурации:

```
System.Configuration.Configuration currentConfig =
       ConfigurationManager.OpenExeConfiguration(
              ConfigurationUserLevel.None );
currentConfig.AppSettings.Settings["server"].Value =
       _serverTextBox.Text;
currentConfig.AppSettings.Settings["database"].Value =
       _serverTextBox.Text;
currentConfig.Save(ConfigurationSaveMode.Modified);
ConfigurationManager.RefreshSection( "appSettings" );
```

Проверка наличия раздела в файле конфигурации и его создание, если таковой отсутствует:

```
if (!ConfigurationManager.AppSettings.AllKeys.Contains(
        "someParameter"))
{
    // открываем текущую конфигурацию специальным обьектом
    System.Configuration.Configuration currentConfig =
        ConfigurationManager.OpenExeConfiguration(
               ConfigurationUserLevel.None );
    // добавляем позицию в раздел AppSettings
    currentConfig.AppSettings.Settings.Add(
        "someParameter", "" );
    //сохраняем
    currentConfig.Save( ConfigurationSaveMode.Full );
    //принудительно перезагружаем соотвествующую секцию
    ConfigurationManager.RefreshSection( "appSettings" );
}
```

Список литературы:

1. Корячко, В.П. Процессы и задачи управления проектами информационных систем [Электронный ресурс] : учебное пособие / В.П. Корячко, А.И. Таганов. — Электрон. дан. — М. : Горячая линия-Телеком, 2014. — 376 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=63237
2. Вылегжанина, А.О. Информационно-технологическое и программное обеспечение управления проектом : учебное пособие / А.О. Вылегжанина. - М. ; Берлин : Директ-Медиа, 2015. - 429 с. : ил., схем., табл. - Библиогр. в кн. - ISBN 978-5-4475-4462-1 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=362892
3. Ехлаков Ю. П. Введение в программную инженерию [Текст]: учебное пособие /Ю.П. Ехлаков. - Томск, «Эль Контент», 2011. - 148 с. Доступ из ЭБС «Университетская библиотека ONLINE». Режим доступа: http://biblioclub.ru/index.php?page=book_red&id=209001&sr=1
4. Рудинский И. Д. Технология проектирования автоматизированных систем обработки информации и управления [Текст]: Учебное пособие для вузов. - М.: Горячая линия - Телеком, 2011. - 304 с. - Доступ из ЭБС «Лань». Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=5191
5. Грекул В. И. Проектное управление в сфере информационных технологий - М.: БИНОМ. Лаборатория знаний, 2013.-336 с. - Доступ из ЭБС «Лань». Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=8809
6. Сайт Интернет-университета информационных технологий [Элетронный ресурс]: офиц. сайт. – Электрон. дан. – Режим доступа: www.intuit.ru
7. ГОСТ Р ИСО/МЭК 12207-2010. Системная и программная инженерия. Процессы жизненного цикла программных средств. – Введ. 2012-03-01. [Электронный ресурс].-Режим доступа: http://docs.pravo.ru/document/view/22517272/21979091/.-Загл. с экрана