Практическая работа 21
Разработка схемы классов. Создание классов

Цель занятия: Получить практические навыки разработки схемы и создания классов.

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio)
Microsoft Visual Studio

Краткие теоретические сведения

UML-модели помогают понимать, обсуждать и разрабатывать системы программного обеспечения. Visual Studio предоставляет шаблоны для пяти часто используемых UML-схем: активности, классов, компонентов, последовательностей и вариантов использования. Кроме того, можно создавать схемы слоев, которые помогают определить структуру системы.

UML-схемы моделирования и схемы слоев могут существовать только внутри проекта моделирования. Все проекты моделирования содержат общую UML-модель и несколько UML-схем. Каждая схема является представлением части модели. UML-модель содержит все элементы, отображаемые на UML-схемах, и может просматриваться с помощью обозревателя моделей UML.

Создание схемы в проекте моделирования

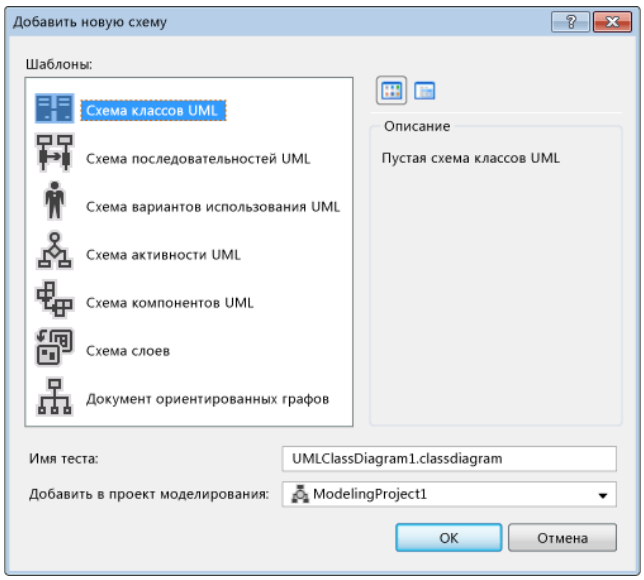
Вам необходим Visual Studio Ultimate для создания проектов моделирования. Вы можете просматривать проекты моделирования в Visual Studio Professional.

Создание схемы и добавление ее в проект

В меню Архитектура (Architecture), выберите пункт Создать схему (New Diagram).

Меню Архитектура доступно только в Visual Studio Ultimate.

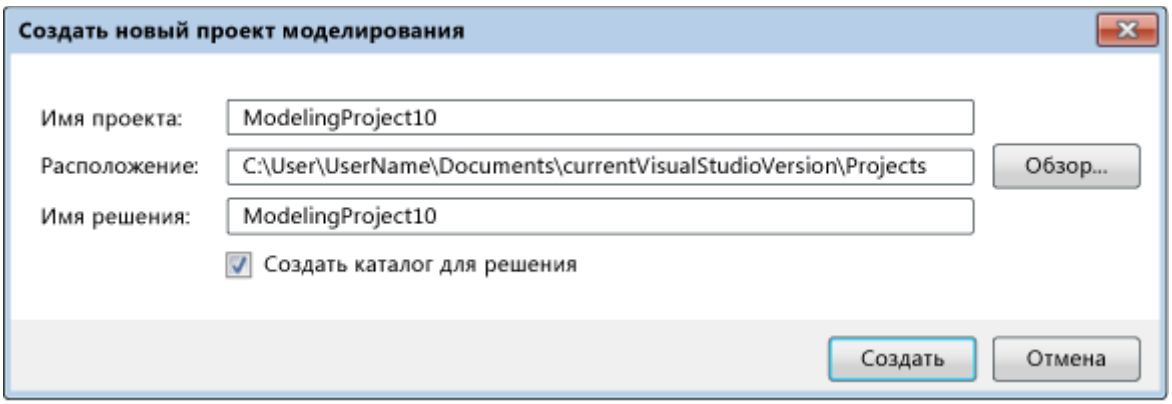
В диалоговом окне Добавление новой схемы выберите требуемый тип схемы моделирования.



Диалоговое окно "Добавление новой схемы"
Введите имя новой схемы.

В окне Добавить в проект моделирования выполните следующее.
Выберите проект моделирования, который уже существует в решении, и нажмите кнопку ОК.
- или -
Выберите Создать новый проект моделирования и нажмите кнопку ОК.
В диалоговом окне Создание нового проекта моделирования введите имя и расположение нового проекта, затем нажмите кнопку Создать.

Диалоговое окно "Создание проекта модели"



Если решение открыто, новый проект добавляется в решение. Если решение не открыто, можно ввести имя нового решения.
Если проект моделирования уже есть, можно использовать следующую процедуру.

Добавление схемы в существующий проект моделирования
В обозревателе решений щелкните узел проекта моделирования.

В меню Проект выберите команду Добавить новый элемент.

В диалоговом окне Добавление нового элемента - <имя проекта> в разделе Шаблоны щелкните тип схемы моделирования, например Схема компонентов UML.

Введите имя схемы и нажмите кнопку Добавить.

Открытая схема моделирования отображается в проекте моделирования.

Не следует добавлять, копировать или перетаскивать файлы существующей схемы слоев в другой проект моделирования или в другие местоположения в решении. Это приведет к исчезновению элементов из скопированных схем или к ошибкам при открытии схем. Файл схемы должен открываться в том проекте моделирования, в котором он создан. Это связано с тем, что UML-схема является представлением модели, принадлежащей проекту моделирования. Чтобы скопировать файл схемы, создайте новую схему, а затем скопируйте элементы из исходной схемы в новую схему. Дополнительные сведения см. в разделе Устранение неполадок в проектах моделирования и схемах.

Создание пустого проекта моделирования

В меню Файл последовательно выберите пункты Создать и Проект.

В диалоговом окне Создать проект в разделе Установленные шаблоны щелкните пункт Проекты моделирования.

В среднем окне щелкните Проект моделирования.

Назовите проект и укажите расположение в полях Имя и Расположение.

В поле Решение выберите Добавить в решение, чтобы добавить новый проект в открытое решение, или Создать новое решение, чтобы закрыть открытые решения и добавить проект в новое решение.

В Visual Studio можно добавлять классы C# или пространства имен из Обозревателя архитектуры или графов зависимостей на uml-схему классов. Также можно добавлять классы C# из обозревателя решений.

Перетаскиваемые классы (или класс) отобразятся на схеме. Классы, от которых они зависят, отобразятся в окне Обозреватель моделей UML. См. Представление типов в модели.

Добавление классов из кода программы в UML-модель

Откройте проект C#.

Создайте проект моделирования UML.

Добавьте UML-схему классов в проект моделирования:

В меню Архитектура (Architecture), выберите пункт Создать схему (New Diagram). В диалоговом окне Добавление новой схемы выберите UML-схема классов.

Откройте Обозреватель архитектуры:

В меню Архитектура выберите Окна, Обозреватель архитектуры.

Перетащите пространства имен или типы из средства Обозреватель архитектуры на поверхность UML-схемы классов.

Можно также перетаскивать пространства имен или типы из диаграмм зависимостей.

Код и схема не обновляются автоматически. Можно обновить схему, чтобы обсудить варианты изменений, не изменяя код. Последующие изменения в коде отобразятся на схеме автоматически, только если снова перетащить классы на схему.

Открытие кода C#, связанного с UML-классом

Дважды щелкните фигуру класса, атрибут или операцию на UML-схеме классов.

Отобразится исходный код.

Представление типов в модели

Типы, явно переносимые на схему, представлены в модели и на схеме напрямую.

Типы, от которых зависят эти явные типы, представлены в модели заполнителями. В модели не представлены ни подробные сведения об этих типах, ни их зависимости.

Однако, если перетащить тип-заполнитель из средства Обозреватель архитектуры или с графов зависимостей на схему, заполнитель будет заменен полноценным типом.

Пример объявления полей в классе:

```
class Student
{
  public string firstName;
  public string lastName;
  }
```

Создание объектов

Объявив класс, мы теперь можем создавать объекты. Делается это при помощи ключевого слова new и имени класса:

```
namespace Program_First
{
  class Student
  {
    public string firstName;
    public string lastName;
    }
```

```
class Program
{
  static void Main(string[] args)
  {
    Student student1 = new Student(); //создание объекта student1 класса Student
    Student student2 = new Student();
  }
}
```

Доступ к членам объекта осуществляется при помощи оператора точка «.» :

```
static void Main(string[] args)
{
  Student student1 = new Student();
  student1. firstName = "Петров";
  student2. lastName = "Василий";

  Console.WriteLine(student1. lastName); // выводит на экран "Ваилий"
   Console.ReadKey();
}
```

Задания

1 Изучить теоретические сведения и задание к работе

2 В соответствии с вариантом задания составить отлаженную программу.

Порядок выполнения работы

Создадим класс Gun (Пушка), в нем объявим поля: логическое isLoaded (заряжена) и целое Ammunition (боеприпасы). Т.к. экземпляры созданного класса планируем создавать в другом классе (class Program), используем спецификатор public, в противном случае при создании объекта класса возникнет ошибка:

```
namespace ConsoleApplication13
{
    ссылок: 4
    class Gun
    {
        public bool isLoaded;
        private int Ammunition;
    }
    ссылок: 0
    class Program
    {
        ссылок: 0
        static void Main(string[] args)
        {
            Gun gun = new Gun();
            Gun howitzer = new Gun();
            gun.Ammunition = 10;
            gun.isLoaded = true;
            gun.Ammunition++;
        }
```

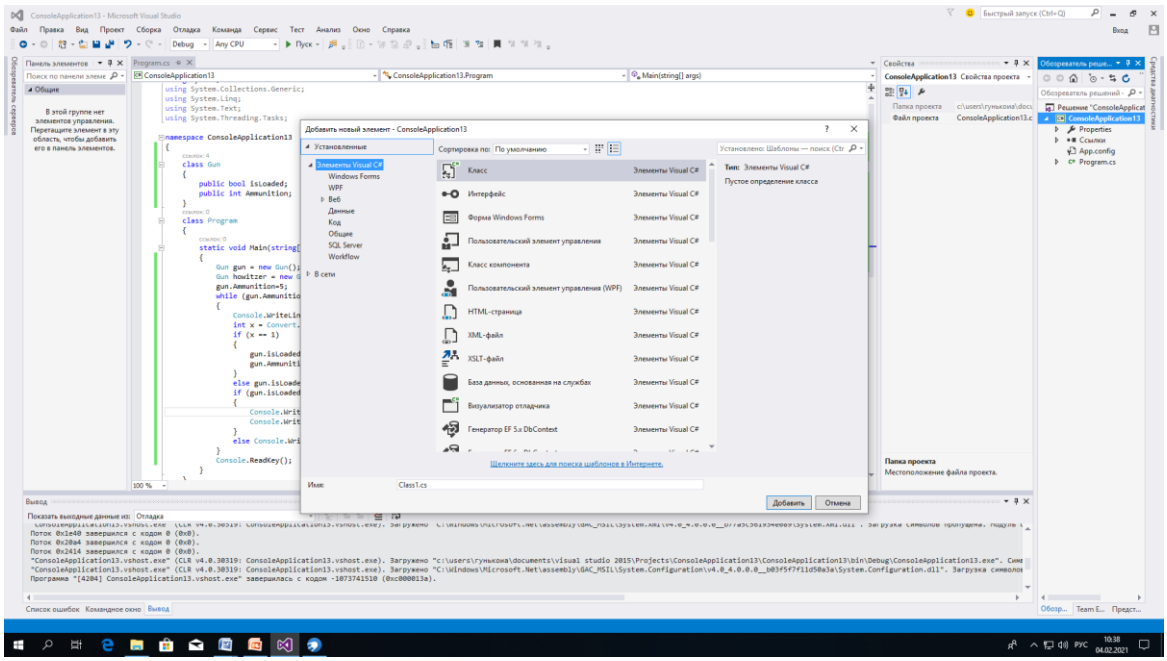
Теперь можно создавать объекты. Делается это при помощи ключевого слова new и имени класса, после чего выполнять действия с этими объектами.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication13
{
    class Program
    {
        static void Main(string[] args)
        {
            Gun gun = new Gun();
            Gun howitzer = new Gun();
            gun.Ammunition=5;
            while (gun.Ammunition > 0)
            {
                Console.WriteLine("Зарядить gun? Если да, введите 1, если нет, введите 0");
                int x = Convert.ToInt32(Console.ReadLine());
                if (x == 1 && gun.Ammunition>0)
                {
                    gun.isLoaded = true;
                    gun.Ammunition--;
                }
                else gun.isLoaded = false;
                if (gun.isLoaded)
                {
                    Console.WriteLine("Пушка  gun заряжена, можно стрелять, боеприпасов осталось  " + gun.Ammunition);
                    Console.ForegroundColor = ConsoleColor.Red;

                    Console.WriteLine("Пыщ - пыщ - пыщ");
                    Console.ForegroundColor = ConsoleColor.White;
                }
                else Console.WriteLine("Пушка  gun не заряжена, стрелять нельзя, боеприпасов осталось  " + gun.Ammunition);
            }
            Console.ReadKey();
        }
    }
}
```

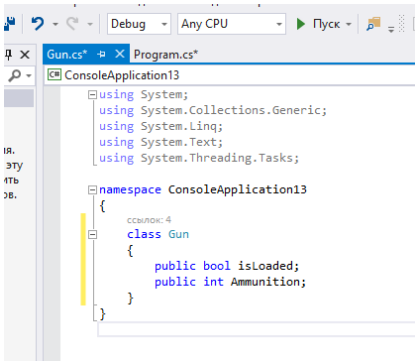
Если в программе предусмотрен ввод данных, необходимо обработать исключения. Результат работы:

```
Зарядить gun? Если да, введите 1, если нет, введите 0
1
Пушка  gun заряжена, можно стрелять, боеприпасов осталось  4
Пыщ - пыщ - пыщ
Зарядить gun? Если да, введите 1, если нет, введите 0
1
Пушка  gun заряжена, можно стрелять, боеприпасов осталось  3
Пыщ - пыщ - пыщ
Зарядить gun? Если да, введите 1, если нет, введите 0
1
Пушка  gun заряжена, можно стрелять, боеприпасов осталось  2
Пыщ - пыщ - пыщ
Зарядить gun? Если да, введите 1, если нет, введите 0
1
Пушка  gun заряжена, можно стрелять, боеприпасов осталось  1
Пыщ - пыщ - пыщ
Зарядить gun? Если да, введите 1, если нет, введите 0
0
Пушка  gun не заряжена, стрелять нельзя, боеприпасов осталось  1
Зарядить gun? Если да, введите 1, если нет, введите 0
```

Теперь вынесем описание класса в отдельный файл, для чего правой кнопкой мыши нажмем на имя проекта , выберем Добавить, а затем Создать элемент, в списке выберем Класс и зададим имя класса



В открывшееся окно вставим описание класса из программы, сохраним файл и убедимся, что программа работает.



Содержание отчета

1 Название работы
2 Цель работы
3 Технические средства обучения
4 Задания (условия задач)
5 Порядок выполнения работы
6 Ответы на контрольные вопросы
7 Вывод

Варианты заданий

1. Класс "Квадрат" с полями: центр квадрата и его сторона.
2. Класс "Время" с полями: часы, минуты и секунды.
3. Класс "Прямоугольник" с полями: верхняя левая и правая нижняя точки.
4. Класс "Рациональное число" с полями: знак числа, целая часть, числитель, знаменатель.
5. Класс "Ромб" с полями: верхняя и правая точки ромба.
6. Класс "Конус" с полями: радиус окружности основания и высота.
7. Класс "Куб" с полями: координаты центра и сторона.
8. Класс «Компьютер» с полями: тактовая частота процессора, объем ОЗУ, емкость диска.
9. Класс «Служащий» с полями: табельный номер, номер подразделения и оклад.
10. Класс «Накладная» с полями: номер, дата отпуска и сумма.
11. Класс «Проект» с полями: номер проекта, сумма, дата исполнения
12. Класс «Здание» с полями: количество этажей, подъездов и квартир.
13. Класс «Физическая величина» с полями: величина, единица измерения, направление.
14. Класс "Дата" с полями: число, номер месяца и две последние цифры года.
15. Класс "Вектор на плоскости" с данными: проекция вектора на оси X и Y.
16. Класс "Окружность" с полями: центр и радиус окружности.
17. Класс "Часть речи" с полями: вопрос, обозначение, член предложения.
18. Класс "Товар" с полями: код, упаковка, цена.
19. Класс «Домашняя видеотека» с полями: название фильма, жанр, страна, длительность.
20. Класс "Параллелограмм" с полями: верхняя левая, верхняя правая и правая нижняя точки.
21. Класс «Домашняя библиотека» с полями: автор, жанр, тип, количество страниц.
22. Класс «Расписание поездов» с полями: название пункта назначения, номер поезда, время отправления.
23. Класс "Дробь" с полями: числитель и знаменатель.
24. Класс «Студент» с полями: ФИО, номер группы, номер зачетки.
25. Класс «Комплексное число» с полями: действительная часть, мнимая часть.

Контрольные вопросы

1. Что называется классом?
2. Укажите элементы класса?
3. Что такое "Тело класса"?
4. Что такое "Спецификаторы класса"

5. Какие спецификаторы доступа определены для классов в C#?
6. Что называется объектом?
7. Укажите операцию, с помощью которой программист создает экземпляр класса
8. Как называются переменные, описанные в классе?
9. Если не указан спецификатор доступа к элементу класса, то такой элемент считается...
10. Какой вид доступа является наиболее предпочтительным для полей класса?

Используемая литература

1. Гниденко, И. Г. Технология разработки программного обеспечения : учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Издательство Юрайт, 2017.
2. Шарп Джон Ш26 Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017.
3. Васильев А.Н. Программирование на C# для начинающих. Основные сведения. – Москва: Эксмо, 2018.
4. Васильев А.Н. Программирование на C# для начинающих. Особенности языка. – Москва: Эксмо, 2019.
5. http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.