Оглавление

[Лабораторная работа № 2. Основы ООП. Инкапсуляция. 1](#_Toc79410813)

[Общие сведения о классах 1](#_Toc79410814)

[Поля класса 1](#_Toc79410815)

[Методы класса 2](#_Toc79410816)

[Конструкторы класса 4](#_Toc79410817)

[Создание объекта класса, работа с экземплярами класса 4](#_Toc79410818)

[Задание 5](#_Toc79410819)

[Требования к программе, общие для всех вариантов. 5](#_Toc79410820)

[Вариант 1. Требования к программе 5](#_Toc79410821)

[Вариант 2. Требования к программе 6](#_Toc79410822)

[Вариант 3. Требования к программе 7](#_Toc79410823)

[Дополнительное задание 9](#_Toc79410824)

# **Лабораторная работа** № 2. Основы ООП. Инкапсуляция.

## Общие сведения о классах

C# является полноценным объектно-ориентированным языком, это значит, что программу на C# можно представить в виде взаимосвязанных взаимодействующих между собой объектов.

Описанием объекта является класс, а объект представляет экземпляр этого класса. Можно провести следующую аналогию. У нас у всех есть некоторое представление о человеке - наличие двух рук, двух ног, головы, пищеварительной, нервной системы, головного мозга и т.д. Все это определяет некоторый шаблон, структуру строения человека, который в терминах ООП будет называться классом. Реально же существующий человек при переходе в ООП будет являться объектом этого класса.

Класс определяется с помощью ключевого слова сlass:

class ClassName

{

}

Вместо ClassName нужно указать имя вашего шаблона данных, сущности, которую вы описываете, например, для класса «человек» это могут быть следующие имена: People, Human, Person и т.д.

Вся функциональность класса представлена его членами: полями (полями называются переменные класса), свойствами, методами, событиями, поэтому следующим шагом будет описание содержимого класса.

### Поля класса

Для начала рекомендуется определиться с полями, т.е. данными, которые содержит данный класс. Шаблон описания полей данных имеет следующий вид:

[мод. доступа] <data type> variable1;

Разберем эту запись подробнее:

* [мод. доступа] – модификатор доступа. Определяет видимость объекта, может принимать 3 значения (на самом деле больше, на данном этапе достаточно 3-х):
  + private – поле доступно только внутри класса, объекты других классов доступа к нему не имеют;
  + protected – тоже самое, что и private + возможность доступа из производных классов;
  + public – поле доступно отовсюду, в т.ч. для клиентов класса.

Для переменных рекомендуется использовать только private.

Если модификатор доступа не указан, считается, что он равен private.

* <data type> - тип данных. Определяет природу хранимого значения, может быть представлен как одним из простых типов, так и представлять собой структуру или другой класс.

Из простых типов будем использовать:

* + int – целое число;
  + double – дробное число;
  + bool – логическая переменная (принимает 2 значения: true или false);
  + string – строка (значение присваивается в кавычках, например “Иванов Иван”);
* variable1 – имя переменной.

С учетом этого, класс «Студент», содержащий 2 характеристики – среднюю оценку и имя студента, может быть объявлен следующим образом:

class Student

{

private double assessment; //private – доступ только из класса, double – дробное значение, assessment – имя переменной (англ. - оценка)

protected String name; //protected – доступ из класса и его наследников, String – строка, name - имя

}

### Методы класса

Только хранения данных для работы программы как правило недостаточно, необходимо производить какую-либо обработку этих данных, взаимодействие с пользователем, внешним оборудованием, т.о. необходимо определить логику поведения объекта. Для этих целей в ООП введено понятие методов, функций - членов класса. Метод представляет собой блок кода, содержащий набор инструкций под заданным именем. Программа инициирует выполнение этого кода, вызывая метод по его имени, при необходимости задавая необходимые аргументы (параметры) метода.

C# не допускает объявления функций вне класса, и даже метод Main, который является точкой входа для каждого приложения C#, по умолчанию является членом класса Program.

Синтаксис объявления метода приведен ниже:

|  |
| --- |
| [мод. доступа] <return type> MethodName( [parameter\_list] )  {  тело метода  } |

* [мод. доступа] - модификатор доступа, назначение описано ранее.
* [parameter\_list] – список параметров. Представляет из себя перечисление объявлений переменных, от которых может зависеть поведение и конечный результат работы метода. Например, функция, складывающая 2 числа, будет иметь 2 параметра, и их определение будет иметь вид:

double a, double b

При вызове метода для каждой из этих переменных будет подставлено какое-то значение.

В случае, если работа метода не зависит ни от каких внешних данных, список параметров может быть пустым.

* <return type> - возвращаемое значение

Если функция генерирует какое-то содержимое, какой-то объект, который ***можно сохранить в другой переменной***, то в качестве возвращаемого значения указывается тип данных этого объекта, например, для мат. функций (sin, cos, sum) – double, для работы со строками – String. Кроме того, в этих случаях, в теле метода необходимо использовать оператор return для возврата результата вызывающей стороне. Например так:

return “Hello, World!”;

return a \* b;

return sum;

Подробнее см. в разделе [return (Справочник по C#)](https://msdn.microsoft.com/ru-ru/library/1h3swy84(v=vs.90).aspx).

Если функция не производит никаких новых данных, которые нужны вызывающему коду, а только обрабатывает существующий объект, работает с глобальными объектами или изменяет другие объекты, то в качестве return\_type указывается void

* тело метода – набор инструкций, отвечающих за логику работы метода.

Примеры определения методов:

class SimpleMath

{

public int AddTwoNumbers(int number1, int number2)

{

return number1 + number2;

}

public int SquareANumber(int number)

{

return number \* number;

}

public int GetSum(int first, int delta, int count)

{

int sum = first;

for(int i = 0; i < count; ++i)

{

sum = sum + delta;

}

return sum;

}

void PrintSum(int a, int b)

{

int s = AddTwoNumbers(a,b);

Console.WriteLine($“Сумма двух чисел равна {s}”);

}

}

### Конструкторы класса

Кроме обычных методов в классах используются также и специальные методы, которые называются **конструкторами**. Конструкторы вызываются при создании нового объекта данного класса. Отличительной чертой конструктора является то, что его название должно совпадать с названием класса:

|  |
| --- |
| class Book  {  //объявления полей и методов пропущены, см. пример из лекции  public Book() //имя совпадает с названием класса  {  name = “”;  year = 0;  }  public Book( string name, int year)  {  this.name = name;  if( year < 0 )  Console.WriteLine(“Год не может быть отрицательным!”);  else  this.year = year;  }  } |

Одно из назначений конструктора - начальная инициализация членов класса. В данном случае мы использовали два конструктора. Один пустой. Второй конструктор наполняет поля класса начальными значениями, которые передаются через его параметры.

### Создание объекта класса, работа с экземплярами класса

Объявление переменной класса ничем не отличается от объявления переменных простого типа, например int, однако, при инициализации переменной, если создается новый объект, всегда должен быть указан оператор new с вызовом подходящего в данном контексте конструктора, например Book b = new Book();

Доступ к членам класса (поля, методы, свойства), осуществляется через оператор . , который ставится после имени объекта к которому осуществляется доступ, после этого оператора указывается то поле или метод, к которому необходимо получить доступ. Например:

class Program

{

    static void Main(string[] args)

    {

        Book b1 = new Book( “Война и мир”, 18xx);

String bookName= “У Лукоморья дуб зеленый”;

Book b2 = new Book( bookName, 0);

b2.Comment = “какой-то комментарий”;

b1.GetInformation();

        Book b2 = new Book();

        b2.GetInformation();

        Console.ReadLine();

    }

}

## Задание

### Требования к программе, общие для всех вариантов.

Определить класс Person, который имеет:

* закрытое поле типа string, в котором хранится имя;
* закрытое поле типа string, в котором хранится фамилия;
* закрытое поле типа System.DateTime для даты рождения.

В классе Person определить конструкторы:

* конструктор c тремя параметрами типа string, string, DateTime для инициализации всех полей класса;
* конструктор без параметров, инициализирующий все поля класса некоторыми значениями по умолчанию.

В классе Person определить свойства c методами get и set:

* свойство типа string для доступа к полю с именем;
* свойство типа string для доступа к полю с фамилией;
* свойство типа DateTime для доступа к полю с датой рождения;
* свойство типа int c методами get и set для получения информации(get) и изменения (set) года рождения в закрытом поле типа DateTime, в котором хранится дата рождения.

В классе Person определить методы:

* метод string ToFullString() для формирования строки со значениями всех полей класса;
* метод string ToShortString(), который возвращает строку, содержащую только имя и фамилию.

### Вариант 1. Требования к программе

Определить тип Education - перечисление(enum) со значениями Specialist, Вachelor, SecondEducation.

Определить класс Exam, который имеет три открытых автореализуемых свойства, доступных для чтения и записи:

* свойство типа string, в котором хранится название предмета;
* свойство типа int, в котором хранится оценка;
* свойство типа System.DateTime для даты экзамена.

В классе Exam определить:

* конструктор с параметрами типа string, int и DateTime для инициализации всех свойств класса;
* конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
* метод string ToFullString() для формирования строки со значениями всех свойств класса.

Определить класс Student, который имеет:

* закрытое поле типа Person, в котором хранятся данные студента;
* закрытое поле типа Education для информации о форме обучения;
* закрытое поле типа int для номера группы;
* закрытое поле типа Exam[] для информации об экзаменах, которые сдал студент.

В классе Student определить:

* конструктор c параметрами типа Person, Education, int для инициализации соответствующих полей класса;
* конструктор без параметров, инициализирующий поля класса значениями по умолчанию;
* свойство типа Person для доступа к полю с данными студента;
* свойство типа Education для доступа к полю с формой обучения;
* свойство типа int для доступа к полю с номером группы;
* свойство типа Exam[] для доступа к полю со списком экзаменов.
* свойство типа double ( только с методом get), в котором вычисляется средний балл как среднее значение оценок в списке сданных экзаменов;
* метод void AddExams ( params Exam [] ) для добавления элементов в список экзаменов;
* метод string ToFullString() для формирования строки со значениями всех полей класса, включая список экзаменов;
* метод string ToShortString(), который формирует строку со значениями всех полей класса без списка экзаменов, но со значением среднего балла.

В методе Main():

* Создать один объект типа Student, преобразовать данные в текстовый вид с помощью метода ToShortString() и вывести данные.
* Присвоить значения всем определенным в типе Student свойствам, преобразовать данные в текстовый вид с помощью метода ToFullString() и вывести данные.
* C помощью метода AddExams( params Exam[] ) добавить элементы в список экзаменов и вывести данные объекта Student, используя метод ToFullString().

### Вариант 2. Требования к программе

Определить тип Frequency - перечисление(enum) со значениями Weekly, Monthly, Yearly.

Определить класс Article, который имеет три открытых автореализуемых свойства, доступных для чтения и записи:

* свойство типа Person, в котором хранятся данные автора статьи;
* свойство типа string для названия статьи;
* свойство типа double для рейтинга статьи.

В классе Article определить:

* конструктор c параметрами типа Person, string, double для инициализации всех свойств класса;
* конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
* метод string ToFullString() для формирования строки со значениями всех свойств класса.

Определить класс Magazine, который имеет

* закрытое поле типа string c названием журнала;
* закрытое поле типа Frequency с информацией о периодичности выхода журнала;
* закрытое поле типа DateTime c датой выхода журнала;
* закрытое поле типа int с тиражом журнала;
* закрытое поле типа Article[] со списком статей в журнале.

В классе Magazine определить:

* конструктор с параметрами типа string, Frequency, DateTime, int для инициализации соответствующих полей класса;
* конструктор без параметров, инициализирующий поля класса значениями по умолчанию;
* свойство типа string для доступа к полю с названием журнала;
* свойство типа Frequency для доступа к полю с информацией о периодичности выхода журнала;
* свойство типа DateTime для доступа к полю c датой выхода журнала;
* свойство типа int для доступа к полю с тиражом журнала;
* свойство типа Article[] для доступа к полю со списком статей;
* свойство типа double ( только с методом get), в котором вычисляется среднее значение рейтинга в списке статей;
* метод void AddArticles (params Article[]) для добавления элементов в список статей в журнале;
* метод string ToFullString() для формирования строки со значениями всех полей класса, включая список статей;
* метод string ToShortString(), который формирует строку со значениями всех полей класса без списка статей, но со значением среднего рейтинга статей.

В методе Main()

* Создать один объект типа Magazine, преобразовать данные в текстовый вид с помощью метода ToShortString() и вывести данные.
* Присвоить значения всем определенным в типе Magazine свойствам,преобразовать данные в текстовый вид с помощью метода ToString() и вывести данные.
* C помощью метода AddArticles( params Article[] ) добавить элементы в список статей и вывести данные объекта Magazine, используя метод ToFullString().

### Вариант 3. Требования к программе

Определить тип TimeFrame - перечисление(enum) со значениями Year, TwoYears, Long.

Определить класс Paper, который имеет три открытых автореализуемых свойства, доступных для чтения и записи:

* свойство типа string, в котором хранится название публикации;
* свойство типа Person для автора публикации;
* свойство типа DateTime c датой публикации.

В классе Paper определить

* конструктор c параметрами типа string, Person, DateTime для инициализации всех свойств класса;
* конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
* метод string ToFullString() для формирования строки со значениями всех полей класса.

Определить класс ResearchTeam, который имеет:

* закрытое поле типа string c названием темы исследований;
* закрытое поле типа string с названием организации;
* закрытое поле типа int – регистрационный номер;
* закрытое поле типа TimeFrame для информации о продолжительности исследований;
* закрытое поле типа Paper[], в котором хранится список публикаций.13

В классе ResearchTeam определить:

* конструктор c параметрами типа string, string, int, TimeFrame для инициализации соответствующих полей класса;
* конструктор без параметров, инициализирующий поля класса значениями по умолчанию.
* свойство типа string для доступа к полю с названием темы исследований;
* свойство типа string для доступа к полю с названием организации;
* свойство типа int для доступа к полю с номером регистрации;
* свойство типа TimeFrame для доступа к полю с продолжительностью исследований;
* свойство типа Paper[] для доступа к полю со списком публикаций по теме исследований.
* свойство типа Paper ( только с методом get), которое возвращает ссылку на публикацию с самой поздней датой выхода; если список публикаций пустой, свойство возвращает значение null;
* метод void AddPapers ( params Paper[] ) для добавления элементов в список публикаций;
* метод string ToFullString() для формирования строки со значениями всех полей класса, включая список публикаций;
* метод string ToShortString(), который формирует строку со значениями всех полей класса без списка публикаций.

В методе Main()

* Создать один объект типа ResearchTeam, преобразовать данные в текстовый вид с помощью метода ToShortString() и вывести данные.
* Присвоить значения всем определенным в типе ResearchTeam свойствам, преобразовать данные в текстовый вид с помощью метода ToString() и вывести данные.
* С помощью метода AddPapers (params Paper []) добавить элементы в список публикаций и вывести данные объекта ResearchTeam.
* Вывести значение свойства, которое возвращает ссылку на публикацию с самой поздней датой выхода;

## Дополнительное задание

Сравните время, необходимое для выполнения операций с элементами одномерного, двумерного прямоугольного и двумерного ступенчатого массивов с одинаковым числом элементов. Для этого в методе Main() необходимо создать:

* одномерный массив;
* двумерный прямоугольный массив;
* двумерный ступенчатый массив.

Тип элементов массивов зависит от варианта лабораторной работы. Массивы должны иметь одинаковое число элементов. Если число строк в двумерном прямоугольном массиве равно nrow, а число столбцов ncolumn, то одномерный массив должен содержать nrow\*ncolumn элементов, в двумерном ступенчатом массиве общее число элементов также должно быть равно nrow\*ncolumn. Значения nrow и ncolumn вводятся в процессе работы приложения в виде одной строки с разделителями. В приглашении, которое получает пользователь, должна быть информация о том, какие символы можно использовать как разделители, число разделителей должно быть больше 1. С помощью метода Split класса System.String приложение разбирает введенную пользователем текстовую строку с информацией о числе строк и числе столбцов двумерного массива и присваивает значения переменным, которые содержат значения nrow и ncolumn.

Приложение распределяет память для всех массивов и инициализирует элементы массивов. Для инициализации элементов можно использовать конструктор без параметров. Для всех элементов массивов выполняется одна и та же операция, например, присваивается значение одному из свойств, определенных для элементов массива. Требуется сравнить время выполнения этой операции для одномерного, двумерного прямоугольного и двумерного ступенчатого массивов с одинаковым числом элементов. Для измерения времени выполнения операций можно использовать свойство Environment.TickCount. Cтатическое свойство TickCount класса Environment имеет тип int, использует информацию системного таймера и содержит время в миллисекундах, которое прошло с момента перезагрузки компьютера. Чтобы получить время выполнения некоторого блока кода, необходимо вызвать Environment.TickCount непосредственно перед блоком и сразу же после последнего оператора блока и взять разность значений. В блоке кода, для которого измеряется время, не должно быть операций распределения памяти для массивов, инициализации элементов массивов и операций вывода данных на консоль. Блоки кода должны содержать только операции с элементами массива. Вычисленные значения времени выполнения операций для трех типов массивов, а также число строк nrow и столбцов ncolumn выводятся на консоль. Вывод должен быть подписан, т.е. вывод должен содержать информацию о том, какому типу массива отвечает выведенное значение.

Для первого варианта элементы массива должны иметь тип Exam, для второго – Article, для третьего – Paper.

Попытайтесь объяснить полученные результаты.