# Лабораторная работа № 1 – Классы

**1 Цель работы**

Изучить основные принципы разработки классов в C#.

1. **Порядок выполнения работы**
   * прочесть краткие теоретические сведения;
   * выполнить задания раздела;
   * составить отчет по лабораторной работе и защитить его у преподавателя.
2. **Общая часть**

**3.1 Основные понятия класса**

Встроенные типы данных используются для хранения одного значения в объявленной переменной. Например, int x хранит целое значение в переменной с именем x.  
В объектно-ориентированном программировании, класс является типом данных, определяющим набор переменных и методов для объявленного объекта.

Например, если необходимо создать программу, которая управляет счетами в банке, то класс BankAccount может быть использован для объявления объекта, который будет иметь все свойства и методы, необходимые для управления индивидуальным счетом банка, такие как переменную balance и методы Deposit и Withdrawal.  
Класс похож на шаблон. Он определяет данные и поведение типа. Определение класса начинается с ключевого слова class, за которым следует имя класса. Тело класса содержит данные и действия, заключенные в фигурные скобки.

class BankAccount

{

//данные (поля) и действия (методы)

}

Так же, как и встроенные типы могут быть использованы для объявления множества переменных, классы могут быть использованы для объявления множества объектов. Мы определяем (разрабатываем) класс, который является шаблоном для создания объектов.

Как только мы создали класс, мы можем создавать объекты, основанные на этом классе. Создание объекта называется инстанциацией (создание экземпляра класса) и производится с использованием ключевого слова new.

BankAccount test = new BankAccount();

Каждый объект имеет свои собственные характеристики - поля.   
Значения этих полей описывают текущее состояние объекта. Например, Person (объект класса Person) может быть в возрасте 30 лет, мужского пола, и называться Antonio.

class Person

{

public string Name;

public bool IsMale;

public int Age;

}

…

Person Antonio = new Person();

test.Name = "Antonio Sanchez";

test.IsMale = true;

test.Age = 30;

**3.2 Инкапсуляция**

Частью значения слова инкапсуляция является идея "окружения" объекта, не только, чтобы сохранить вместе то, что находится внутри, а также чтобы защитить его.

В программировании инкапсуляция значит больше, чем просто комбинирование порций функционала внутри классов; она также значит ограничение доступа к этому классу от внешнего воздействия. Инкапсуляция реализуется с помощью модификаторов доступа. Модификаторы доступа определяют область видимости членов класса.

C# поддерживает следующие модификаторы доступа: public, private, protected, internal, protected internal.

Как уже было отображено в предыдущих примерах, модификатор доступа public делает элемент класса доступным снаружи класса.   
Модификатор доступа private делает элементы класса доступными только изнутри класса, и прячет их от доступа снаружи.

Пример инкапсуляции:

class BankAccount

{

private double Balance = 0.0;//баланс счета

public void Deposit(double n)//пополнение

{

Balance += n;

}

public void Withdraw(double n)//списание

{

Balance -= n;

}

public double GetBalance()//получение баланса

{

return Balance;

}

}

…

BankAccount test = new BankAccount();

test.Deposit(10000);//пополнение

test.Withdraw(99.99);//списание

Console.WriteLine(test.GetBalance());//вывод баланса

В данном случае инкапсуляция использована для защиты значения переменной Balance от внешнего кода. Доступ к переменной предоставлен через публичные методы, однако нет возможности использовать переменную напрямую. Это помогает управлять целостностью данных, а также избежать ошибок.

**3.3 Свойства**

Как мы видели в предыдущих примерах, инкапсуляция элементов класса является хорошей практикой и предоставляет доступ к ним только через публичные методы.

Свойство является элементом, предоставляющим гибкий механизм чтения, записи или вычисления значения приватного поля. Свойства могут использоваться, в качестве публичных элементов данных, но они в действительности включают специальные методы, называемые элементами доступа.

Элемент доступа свойства содержит выполняемые выражения, которые помогают получить (прочитать или вычислить) или установить (записать) соответствующее поле. Объявление элементов доступа может включать элементы get, set, или и тот и другой.

Например:

class Person

{

private int age;

public int Age

{

get { return age; }

set { age = value; }

}

}

...

Person test = new Person();

test.Age = 30;

value является специальным ключевым словом, представляющим значение, присваиваемое свойству.

Любой элемент доступа свойства может быть опущен.   
Например, следующий код создает свойство, доступное только для чтения:

class BankAccount

{

private double balance = 0.0;//баланс счета

public double Balance

{

get { return balance; }

}

}

Так зачем использовать свойства? Почему просто не объявить переменную-элемент публичной и не получить к ней доступ напрямую?

Со свойствами у вас есть возможность контролировать логику доступа к переменной.

Например, вы можете проверить, больше ли 0 значение переменной age, до присваивания его к переменной:

class Person

{

private int age;

public int Age

{

get { return age; }

set { if (value > 0) age = value; }

}

}

**3.4 Конструкторы**

Конструктор класса является специальным элементом метода класса, который выполняется при создании нового объекта этого класса.  
Конструктор имеет такое же имя, как у класса, является публичным и не имеет никакого возвращаемого типа.

class Person

{

public string Name;

public bool IsMale;

public int Age;

public Person()

{

Console.WriteLine("Создан новый экземпляр класса Person!");

}

}

Это может быть полезным во множестве ситуаций. Например, при создании объекта типа BankAccount, вы можете отправить уведомление по электронной почте к его обладателю.  
Такая же функциональность может быть достигнута, с помощью отдельного публичного метода. Преимуществом конструктора является то, что он вызывается автоматически.

Использование конструкторов может быть очень полезно для установки начальных значений для определенных переменных.  
Конструктор по умолчанию не имеет параметров. Однако, при необходимости, параметры могут быть добавлены к конструктору. Это делает возможным присваивание первоначального значения к объекту при его создании, как отображено в следующем примере:

class Person

{

public string Name;

public bool IsMale;

public int Age;

public Person(string name, bool isMale, int age)

{

Name = name;

IsMale = isMale;

Age = age;

}

}

...

Person test = new Person("Иван", true, 32);

Console.WriteLine(test.Name);

Console.WriteLine(test.IsMale);

Console.WriteLine(test.Age);

Конструктор может быть перегружен, как и любой метод, с использованием необходимого количества аргументов.

class Person

{

public string Name;

public bool IsMale;

public int Age;

public Person(string name, bool isMale, int age)

{

Name = name;

IsMale = isMale;

Age = age;

}

public Person()

{

Name = "Default";

IsMale = true;

Age = 18;

}

}

**3.5 Статические элементы и классы**

Теперь пришло время обсудить ключевое слово static.  
Впервые вы обнаружили его в объявлении метода Main.

Члены класса (переменные, свойства, методы) могут быть также объявлены с использованием ключевого слова static. Таким образом эти методы становятся принадлежащими самому классу, а не отдельным объектам. Не важно, сколько объектов класса создано, существует только одна копия статического члена.

Например:

class Person

{

public static int Count = 0;

public string Name;

public bool IsMale;

public int Age;

public Person()

{

Count++;

}

}

...

Person test1 = new Person();

Person test2 = new Person();

Person test3 = new Person();

Person test4 = new Person();

Console.WriteLine(Person.Count);

Не важно, сколько создано объектов класса Person, всегда будет только одна переменная count, которая принадлежит классу Person, потому что она объявлена с использованием ключевого слова static.

Из-за своей общей сущности, доступ к статическим членам может быть получен напрямую, с помощью имени класса без объекта.

Такая же концепция применяется к статическим методам.

class Person

{

private static int Count = 0;

public string Name;

public bool IsMale;

public int Age;

public Person()

{

Count++;

}

public static int GetCount()

{

return Count;

}

}

…

Person test1 = new Person();

Person test2 = new Person();

Person test3 = new Person();

Person test4 = new Person();

Console.WriteLine(Person.GetCount());

Статические методы могут получать доступ только к статическим элементам.

Целый класс может быть объявлен статическим.  
Статический класс может содержать только статические члены.   
Вы не можете создавать экземпляры статического класса, так как только один экземпляр статического класса может существовать в программе.  
Статические классы полезны для комбинации логических свойств и методов.

Хорошим примером этого является класс Math.  
Он содержит различные полезные свойства и методы для математических операций.

1. **Общее задание для выполнения на паре**

Разработать класс клиента интернет-магазина. Класс содержит следующие поля:

- Идентификатор в системе (Id);

- Имя;

- Возраст;

- Пол;

- Логин;

- Пароль.

Инкапсулировать значения полей «Пароль» и «Id» с использованием соответствующих методов.

Используя свойства, запретить ввод возраста меньше 18.

Создать два конструктора:

- Задающий все значения полей, принимаемые в качестве аргументов;

- Без аргументов, задающий значения полей по умолчанию.

Создать приватное статическое поле Count, итерируемое в конструкторах.

В конструкторах приравнять полю Id значение поля Count после итерации.

Создать статический метод, принимающий в качестве аргумента экземпляр созданного класса и выводящий к консоли с новой строки значения полей в формате «Название поля: Значение».

1. **Индивидуальные** **задания для выполнения работы**

Каждый разрабатываемый класс должен, как правило, содержать следующие элементы: скрытые поля, конструкторы с параметрами и без параметров, методы, свойства, статические элементы.

**Вариант 1**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о некоторых транспортных средствах, которое будет хранить значения модели(string), мощности(л/с)(int), года выпуска(int) и цвета(string) некоторого транспортного средства. Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 2**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о кредитных картах, которое будет хранить значения владелец(string); срок службы карты: месяц(int), год(int), а также номер карты(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 3**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о сотовых телефонах, которое будет хранить значения фирмы производителя(string), модели(string), количество(int) и метку о том является ли товар Б/У(bool). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 4**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о результатах футбольных матчей, которое будет хранить значения название команды встречающих(string), название команды гостей(string), счет сыгранного матча(int):(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 5**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о часах, которое будет хранить значения тип часов(string), срок гарантии(int), цена(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 6**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о продуктах на складе овощей, которое будет хранить значения название(string); дата завоза(DateTime); а также цена за кг(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 7**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о телепрограмме на день, которое будет хранить значения название телепередачи(string), жанр(string), время показа(int):(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 8**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о складе бытовой техники, которое будет хранить значения вид устройства(string); название фирмы производителя(string); модель(string), метку о том является ли товар Б/У(bool). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 9**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о транспортных перевозках, которое будет хранить значения город отправки(string), город назначения(string), расстояние(int), срок доставки(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 10**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о зернохранилище, которое будет хранить значения сорт пшеницы(string); дата (DateTime); а также объем в тоннах(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 11**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о продукции завода по производству плитки, которое будет хранить значения тип плитки(string); материал(string); цена за штуку(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 12**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о исторических постройках, которое будет хранить значения название постройки(string), примерный год создания(int), местоположение(срана)(string) некоторого транспортного средства. Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 13**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о компьютерных запчастях, которое будет хранить значения фирмы производителя(string), наименования(string), модель(string), цена за шт(int) и метку о том является ли товар Б/У(bool). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 14**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о продуктах на складе фруктов, которое будет хранить значения название(string); дата завоза(DateTime); а также цена за кг(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 15**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о продукции завода по производству кирпича, которое будет хранить значения тип кирпича(string); материал(string), цена за штуку(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 16**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о продукции цеха по производству ковров, которое будет хранить значения тип материала(string); тип рисунка(string), страна производитель(string). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 17**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о работе столярного цеха, которое будет хранить значения наименования изделия(string), наименования(string), материал(string), сложность работы(string), и метку о том находиться ли предмет на реставрации(bool). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 18**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о предоставляемом перечне каналов компанией спутникового ТВ, которое будет хранить значения название канала (string), тип канала (string), длительность эфира(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 19**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о товаре цветочного магазина, которое будет принимать сорт цветов(string); цвет(string); дата завоза (DateTime); а также цена за шт.(int). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 20**

На примере, приведенном в теоретическом материале разработать приложение, хранящее информацию о товарах магазина детских игрушек, которое будет хранить значения возраст ребенка, которому подходит данная игрушка(int); тип игрушки(string), страна производитель(string). Реализовать статический метод, принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 21\***

На примере, приведенном в теоретическом материале разработать приложение, реализующее базу библиотеки, внутренняя структура представляется из 4 классов(без наследования), 1-ый Книги выдаваемые на руки, 2-ой книги для читального зала, 3-ий книги в электронном формате, четвертый класс должен содержать перегруженный конструктор сохраняющий каждый новый элемент класса в отдельном списке(List<>), так же должны быть реализованы методы возвращающие элементы хранящиеся в списке(для этого в каждом классе должны быть прописаны свойства для переменных). Все классы принимают одинаковые значения: название книги(string), издание(string), год издания(int), к какому классу книг принадлежит. Также должен быть реализован метод удаления элемента из списка по индексу. Реализовать статический метод (причем для каждой категории книг свой), принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 22\***

На примере, приведенном в теоретическом материале разработать приложение, реализующее базу фильмов, внутренняя структура представляется из 4 классов(без наследования), 1-ый Зарубежные фильмы, 2-ой Отечественные фильмы, 3-ий советские фильмы, четвертый класс должен содержать перегруженный конструктор, сохраняющий каждый новый элемент класса в отдельном массиве(ArrayList), также должны быть реализованы методы, возвращающие элементы, хранящиеся в массиве (для этого в каждом классе должны быть прописаны свойства для переменных). Все классы принимают одинаковые значения: название фильма(string), компания снявшая фильм(string), год выпуска в прокат(int), к какой категории фильмов принадлежит. Реализовать статический метод (причем для каждой категории фильмов свой), принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 23\***

На примере, приведенном в теоретическом материале разработать приложение, реализующее базу музыкальных инструментов, внутренняя структура представляется из 5 классов(без наследования), 1-ый Духовые инструменты, 2-ой клавишные инструменты, 3-ий струнные инстменты,4-ый ударные инструменты, пятый класс должен содержать перегруженный конструктор сохраняющий каждый новый элемент класса в отдельном списке(List<>), так же должны быть реализованы методы возвращающие элементы хранящиеся в списке(для этого в каждом классе должны быть прописаны свойства для переменных). Все классы принимают одинаковые значения: фирма производитель(string), страна производитель(string), материал из которого изготовлен инструмент(string), к какому классу инструментов относиться. Также должен быть реализован метод удаления элемента из списка по индексу. Реализовать статический метод (причем для каждой категории инструментов свой), принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 24\***

На примере, приведенном в теоретическом материале разработать приложение, реализующее базу винной фабрики, внутренняя структура представляется из 4 классов(без наследования), 1-ый красное вино, 2-ой белое вино, 3-ий розовые вино, четвертый класс должен содержать перегруженный конструктор сохраняющий каждый новый элемент класса в отдельном массиве(List<>), так же должны быть реализованы методы возвращающие элементы хранящиеся в массиве(для этого в каждом классе должны быть прописаны свойства для переменных). Все классы принимают одинаковые значения: сорт винограда(string), количество(int), год(int), к какому типу вина относиться. Также должен быть реализован метод изменения элемента из массива по индексу. Реализовать статический метод (причем для каждой категории вина свой), принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».

**Вариант 25\***

На примере, приведенном в теоретическом материале разработать приложение, реализующее базу завода по производству лампочек, внутренняя структура представляется из 4 классов(без наследования), 1-ый люминесцентные лампы , 2-ой лампа накаливания 3-ий энергосберегающие лампы, четвертый класс должен содержать перегруженный конструктор сохраняющий каждый новый элемент класса в отдельном массиве(ArrayList), так же должны быть реализованы методы возвращающие элементы хранящиеся в массиве(для этого в каждом классе должны быть прописаны свойства для переменных). Все классы принимают одинаковые значения: цветовая температура(int), срок службы(int), световая отдача(int), к какой категории ламп принадлежит. Также должен быть реализован метод удаления элемента из массива по индексу. Реализовать статический метод (причем для каждой категории ламп свой), принимающий в качестве аргумента экземпляр класса и выводящий значения в консоль (с новой строки), в формате «Название поля: Значение поля».