**Пример разработки программы, которая размещается в разных файлах. Реализация наследования в классах. Средства MS Visual Studio 2019 разбивки программы на части**

При создании иерархий классов, содержащих большие объемы программного кода, целесообразно реализовывать каждый класс в отдельном файле (модуле). Разбивка объемных программ на отдельные части (файлы) имеет следующие преимущества:

* облегчает восприятие общей структуры программы;
* ускоряет поиск нужных фрагментов кода;
* уменьшает количество трудноуловимых ошибок, связанных с разнообразием кодов, сконцентрированных в одном файле.

В данном примере с помощью средств Microsoft Visual Studio 19 разработана программа, код которой разбит на части. Каждая часть программы реализована в отдельном файле. С целью удобства использования, в каждом файле объявляется отдельный класс. Имя каждого файла совпадает с именем класса, который в нем реализован.  
Пример демонстрирует реализацию 3 классов, взаимодействующих между собой с помощью отношения наследования (отношение is-a).

**Содержание**

* [Условие задачи](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#task)
* [Выполнение](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#instr)
  + [1. Создание проекта по шаблону Console Application](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q01)
  + [2. Начальная конфигурация программы](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q02)
  + [3. Добавление класса Human и файла Human.cs](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q03)
  + [4. Добавление класса Citizen и файла Citizen.cs](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q04)
  + [5. Добавление класса ForeignCitizen и файла ForeignCitizen.cs](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q05)
  + [6. Структура программы](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q06)
  + [7. Программный код класса Human](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q07)
  + [8. Программный код класса Citizen](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q08)
  + [9. Программный код класса ForeignCitizen](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q09)
  + [10. Текст функции Main (). Тестирование работы программы](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q10)
  + [11. Результат работы программы](https://www.bestprog.net/ru/2020/03/26/c-an-example-of-the-development-program-which-is-placed-in-different-files-ru/#q11)

### Условие задачи

Реализовать иерархию классов. Базовый класс Human содержит информацию:

* фамилия;
* имя;
* пол.

От класса Human нужно унаследовать производный класс Citizen, который содержит следующую информацию:

* номер паспорта;
* адрес проживания.

От класса Citizen нужно унаследовать производный класс ForeignCitizen, который содержит дополнительную информацию:

* номер загранпаспорта;
* дата открытия визы;
* дата окончания визы.

Во всех классах должны быть следующие элементы:

* конструктор с параметрами, инициализирующий поля конкретного класса;
* свойства get/set для доступа к полям классов;
* метод Print(), который выводит информацию о полях класса на экран.

Дополнительные условия:

* классы должны быть в разных модулях. Класс Human реализовать в модуле (файле) «human.cs», класс Citizen реализовать в модуле «citizen.cs», класс ForeignCitizen реализовать в модуле «foreignc.cs»;
* класс ForeignCitizen должен быть последним в иерархии и не поддерживать наследования (sealed-класс).

### Выполнение

##### 1. Создание проекта по шаблону Console Application

После запуска MS Visual Studio 2019 нужно создать новый проект. Это осуществляется стандартным способом с помощью последовательности команд

File-> New-> Project...

Если Visual Studio 2019 только что запустился на выполнение, то система автоматически откроет соответствующее окно, в котором можно создать новый проект или выбрать уже существующие проекты.

С помощью оконного интерфейса нужно выполнить следующие действия (рисунок 1):

* в элементе выбора языка программирования выбрать C#;
* в перечне шаблонов выбрать Console App (.NET Core);
* для перехода дальше выбрать кнопку Next.

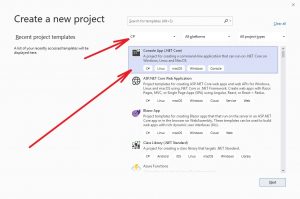
[](https://www.bestprog.net/wp-content/uploads/2020/03/02_02_02_11_06_01_.jpg)

Рисунок 1. Окно «Create a New Project» создание нового проекта в MS Visual Studio 2019

После выполненных действий откроется новое окно «Configure your new project» (рисунок 2). В этом окне нужно указать имя проекта, его размещение и имя решения.  
В нашем случае задается имя DemoSplitFiles. Размещение Location и имя решения можно оставить по умолчанию, которое установлено Visual Studio. После выбора кнопки Create система сгенерирует шаблон консольного приложения.

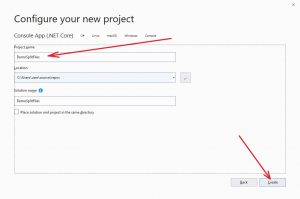
[](https://www.bestprog.net/wp-content/uploads/2020/03/02_02_02_11_06_02_.jpg)

Рисунок 2. Окно «Configure your new project»

##### 2. Начальная конфигурация программы

На данный момент программа состоит из одного файла Program.cs. В этом файле объявляется класс Program, который содержит точку входа в программу – статическую функцию Main ().

На данный момент текст модуля Program.cs следующий

using System;

namespace DemoSplitFiles

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

}

}

}

##### 3. Добавление класса Human и файла Human.cs

В этом пункте подробно описывается добавление нового файла к проекту. В соответствии с условием задачи, класс Human должен быть реализован в файле (модуле) human.cs.

Система Visual Studio 2019 (да и другие версии) построена таким образом, что при добавлении нового класса с помощью интерфейса Visual Studio, этот класс автоматически будет размещаться в новом файле. Нужно только указать имя этого файла.

Чтобы добавить новый класс к проекту, в главном меню нужно вызвать последовательность команд

Project->Add Class...

как показано на рисунке 3.

[](https://www.bestprog.net/wp-content/uploads/2020/03/02_02_02_11_06_03_.jpg)

Рисунок 3. Команда «Add Class…». Добавление нового класса к проекту

В результате откроется окно «Add New Item — DemoSplitFiles», в котором нужно выполнить следующие действия:

* в перечне групп инсталлированных элементов выбрать «Visual C# Items»;
* в центре окна из перечня элементов выбрать элемент Class;
* в поле Name задать имя файла Human.cs;
* после выполнения настроек выбрать кнопку Add.

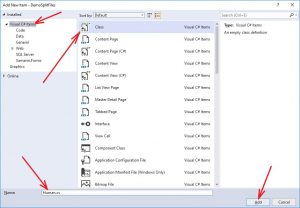
[](https://www.bestprog.net/wp-content/uploads/2020/03/02_02_02_11_06_04_.jpg)

Рисунок 4. Окно «Add New Item». Указание имени файла для класса Human

В результате выполненных действий будет создан класс Human, который размещается в файле Human.cs (рисунок 5).

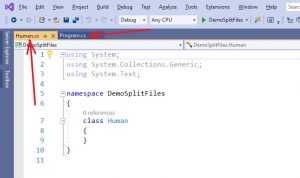
[](https://www.bestprog.net/wp-content/uploads/2020/03/02_02_02_11_06_05_.jpg)

Рисунок 5. Окно программы с активным файлом Human.cs

Как видно из рисунка 5, на данный момент программа состоит из двух файлов:

* Human.cs – файл, в котором размещается класс Human;
* Program.cs – файл, в котором размещается класс Program.

Оба класса размещаются в одном пространстве имен DemoSplitFiles.

Текст класса Human пока что следующий

using System;

using System.Collections.Generic;

using System.Text;

namespace DemoSplitFiles

{

class Human

{

}

}

##### 4. Добавление класса Citizen и файла Citizen.cs

На этом этапе нужно добавить класс Citizen и файл Citizen.cs. Добавление класса Citizen осуществляется по образцу добавления класса Human (смотрите предыдущий пункт 2.3).

После того, как класс Citizen будет добавлен, текст файла Citizen.cs будет следующим

using System;

using System.Collections.Generic;

using System.Text;

namespace DemoSplitFiles

{

class Citizen

{

}

}

Класс Citizen размещается в пространстве имен DemoSplitFiles как и предыдущие классы Program и Human.

##### 5. Добавление класса ForeignCitizen и файла ForeignCitizen.cs

По образцу предыдущих классов Human, Citizen добавляется класс ForeignCitizen. После добавления, класс размещается в отдельном файле ForeignCitizen.cs и содержит следующий код

using System;

using System.Collections. Generic;

using System.Text;

namespace DemoSplitFiles

{

class ForeignCitizen

{

}

}

##### 6. Структура программы

На данный момент программа состоит из 4 файлов:

* Program.cs – содержит текст класса Program, в котором описывается статическая функция Main (). Эта функция есть точкой входа в программу, она выполняется первой при запуске приложения;
* Human.cs – содержит текст класса Human, который есть базовым для классов Citizen и ForeignCitizen;
* Citizen.cs – содержит текст класса Citizen, который унаследован от класса Human и есть базовым для класса ForeignCitizen;
* ForeignCitizen.cs – содержит текст класса ForeignCitizen, который унаследован от класса Citizen.

##### 7. Программный код класса Human

В соответствии с условием задачи класс Human есть базовым в иерархии классов. Текст класса Human следующий:

using System;

namespace DemoSplitFiles

{

// Класс Human - описывает человека

class Human

{

// 1. Внутренние поля класса

protected string name; // Имя

protected string surname; // Фамилия

protected bool sex; // Пол: true - женский, false - мужской

// 2. Конструктор с 3 параметрами

public Human (string \_name, string \_surname, bool \_sex)

{

name = \_name;

surname = \_surname;

sex = \_sex;

}

// 3. Свойства для доступа к внутренним полям класса

public string Name

{

get {return name;}

set {name = value;}

}

public string Surname

{

get {return surname;}

set {surname = value;}

}

public bool Sex

{

get {return sex;}

set { sex = value; }

}

// 4. Метод Print () - выводит имена полей класса на экран

public void Print ()

{

Console.WriteLine("name: {0}", name);

Console.WriteLine("surname: {0}", surname);

if (sex)

Console.WriteLine("sex: Female");

else

Console.WriteLine("sex: Male");

}

}

}

##### 8. Программный код класса Citizen

Текст файла Citizen.cs, в котором размещается класс Citizen следующий

using System;

namespace DemoSplitFiles

{

// Класс Citizen - унаследован от класса Human

class Citizen: Human

{

// 1. Внутренние поля класса

protected string password; // Номер паспорта

protected string address; // Адрес проживания

// 2. Конструктор класса - вызывает конструктор базового класса

public Citizen (string \_name, string \_surname, bool \_sex,

string \_password, string \_address): base (\_name, \_surname, \_sex)

{

password = \_password;

address = \_address;

}

// 3. Свойства доступа к полям класса

public string Password

{

get {return password;}

set {password = value;}

}

public string Address

{

get {return address;}

set {address = value;}

}

// 4. Метод, который виводит значения полей класса

public new void Print ()

{

base. Print (); // вызов метода базового класса

Console.WriteLine("password: {0}", password);

Console.WriteLine("address: {0}", address);

}

}

}

##### 9. Программный код класса ForeignCitizen

Текст файла ForeignCitizen.cs, в котором размещается класс ForeignCitizen следующий

using System;

namespace DemoSplitFiles

{

// Класс ForeignCitizen - унаследован от класса Citizen.

// Данный класс есть запечатанным - не может быть унаследован другим классом

sealed class ForeignCitizen: Citizen

{

// 1. Внутренние поля класса

private string foreignPassport; // номер заграничного паспорта

private DateTime visaOpen; // дата открытия визы

private DateTime visaClose; // дата закрытия визы

// 2. Конструктор класса - вызывает конструктор базового класса

public ForeignCitizen (string \_name, string \_surname, bool \_sex,

string \_password, string \_address,

string \_foreignPassport, DateTime \_visaOpen, DateTime \_visaClose)

: base (\_name, \_surname, \_sex, \_password, \_address)

{

foreignPassport = \_foreignPassport;

visaOpen = \_visaOpen;

visaClose = \_visaClose;

}

// 3. Свойства для доступа к полям класса

public string ForeignPassport

{

get {return foreignPassport;}

set {foreignPassport = value;}

}

public DateTime VisaOpen

{

get {return visaOpen;}

set {visaOpen = value;}

}

public DateTime VisaClose

{

get {return visaClose;}

set {visaClose = value;}

}

// 4. Метод Print () - вывод полей класса

public new void Print ()

{

base. Print (); // вызов метода базового класса

Console.WriteLine("foreignPassport: {0}", foreignPassport);

Console.WriteLine("visaOpen: {0}", visaOpen);

Console.WriteLine("visaClose: {0}", visaClose);

}

}

}

##### 10. Текст функции Main (). Тестирование работы программы

Функция Main () реализована в классе Program, который объявлен в файле Program.cs. В функции Main () демонстрируется создание экземпляров разработанных классов, реализованных в других файлах. Поскольку все классы объявлены в пространстве имен DemoSplitFiles, то для использования имен этих классов не нужно указывать дополнительных уточнений или иных действий. Как показывает данный пример, работа с классами, размещенными в разных файлах MS Visual Studio — C# есть довольно удобной.  
Ниже приведен текст файла Program.cs.

using System;

namespace DemoSplitFiles

{

class Program

{

static void Main (string [] args)

{

// 1. Использование экземпляра класса Human

Human hm = new Human ("John", "Johnson", false);

hm.Print();

// 2. Использование экземпляра класса Citizen

Citizen ct = new Citizen ("Petr", "Petrenko", false, "A101ksdl233", "New City");

Console.WriteLine("------------------------");

ct.Print(); // вывести значение полей класса

// 3. Использование экземпляра класса ForeignCitizen

ForeignCitizen fc = new ForeignCitizen ("Ivanov", "Ivan", false,

"0230902ADK", "London", "023329032JJK",

Convert.ToDateTime("02.02.2020"), Convert.ToDateTime("03.03.2085"));

Console.WriteLine("------------------");

fc. Print ();

}

}

}

##### 11. Результат работы программы

После запуска на выполнение, программа выдает следующий результат

name: John

surname: Johnson

sex: Male

------------------------

name: Petr

surname: Petrenko

sex: Male

password: A101ksdl233

address: New City

------------------

name: Ivanov

surname: Ivan

sex: Male

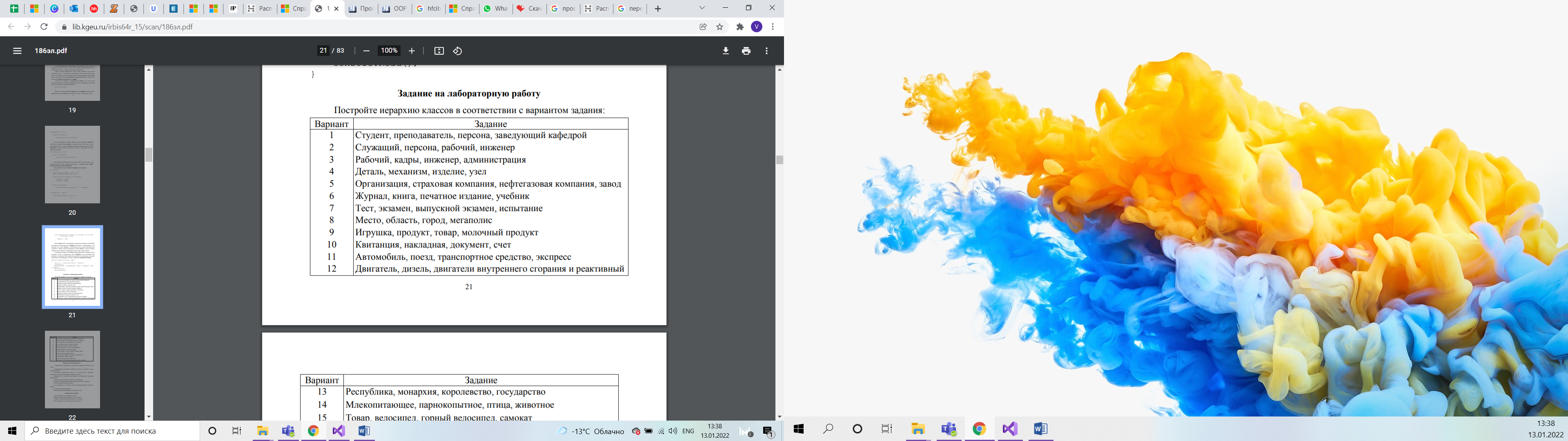
password: 0230902ADK

address: London

foreignPassport: 023329032JJK

visaOpen: 2/2/2020 12:00:00 AM

visaClose: 3/3/2085 12:00:00 AM



**Номер 11**