Часть 1
Наследование

C# Dev

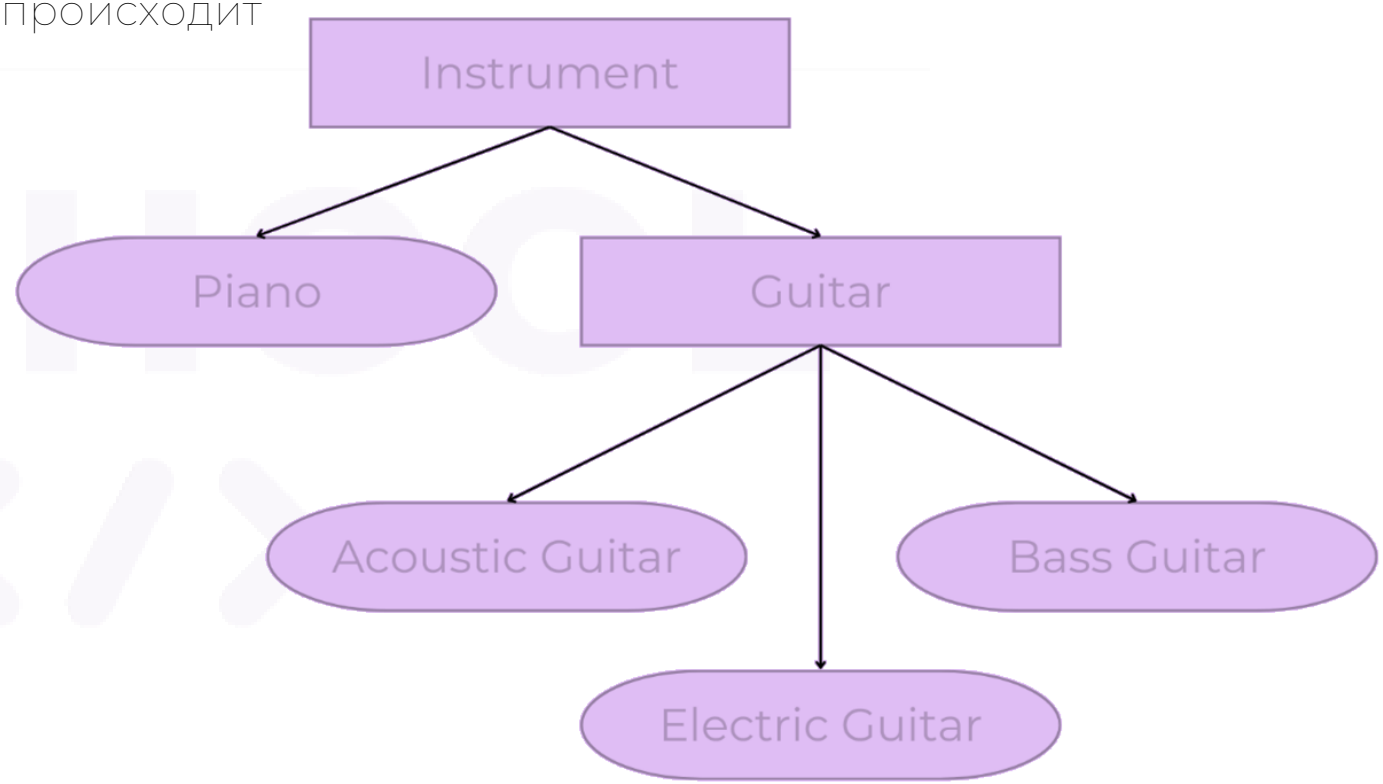
1. Какие есть циклы и их особенность
2. Запишите перебор элементов массива строк циклами while, foreach
3. Запишите перебор элементов массива объектов с конца циклами while, for
4. Какие есть типы данных
5. Преобразование типов данных и привидите их примеры
6. Что такое класс?
7. Что такое конструктор?
8. Что такое ключевое слово this?
9. Что такое поле и свойство. В чем их отличия?
10. 4 принципа ООП
11. SOLID
12. Создать объект условного класс Test, у которого конструтор принимает string и decimal

Наследование

Наследование является одним из ключевых принципов объектно-ориентированного программирования. За счёт наследования можно создать один общий класс (класс родитель), в котором определяются характерные особенности, присущие множеству связанных элементов, и создать множество других классов (классы наследники), наследуемые от родительского класса, добавляя в него свои индивидуальные особенности.

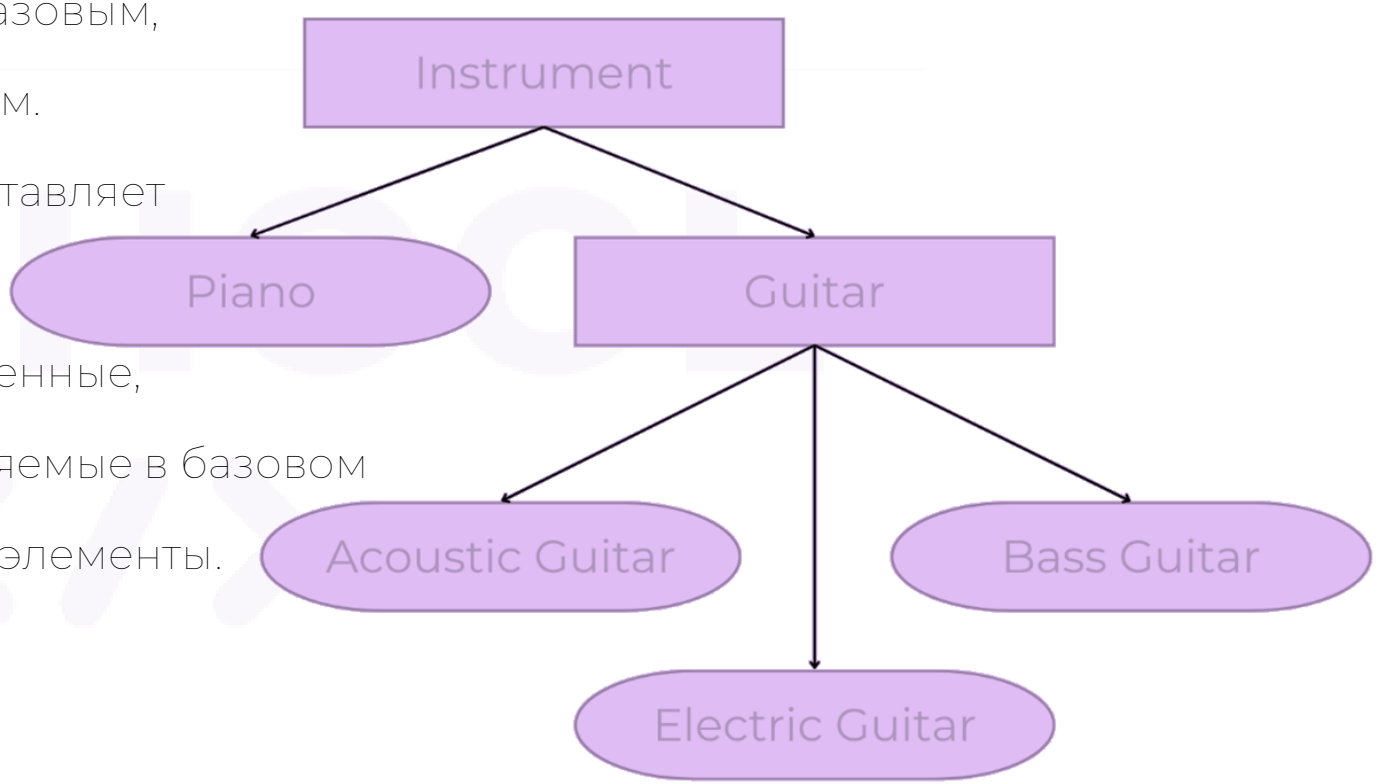
Наследование

Таким образом с помощью наследования происходит создание иерахической классификации



Наследование

Класс, который наследуется, называется базовым, а класс, который наследует — производным. Следовательно, производный класс представляет собой специализированный вариант базового класса. Он наследует все* переменные, методы, свойства и индексаторы, определяемые в базовом классе, добавляя к ним свои собственные элементы.



* Исключение:

Статические конструкторы, конструкторы экземпляров, методы завершения

Зачем использовать наследование?

Главное преимущество наследования заключается в следующем: как только будет создан базовый класс, в котором определены общие для множества объектов атрибуты, он может быть использован для создания любого числа более конкретных производных классов. А в каждом производном классе может быть точно выстроена своя собственная классификация.

Общий вид наследования

```
модификатор_доступа class Родительский_класс {
        //тело Родительского класса
}

модификатор_доступа class Производный_класс_1 : Родительский_класс {
        //тело Производного класса 1
}

модификатор_доступа class Производный_класс_2 : Родительский_класс {
        //тело Производного класса 2
}
```

Когда один класс наследует от другого, после имени производного класса указывается имя базового класса, отделяемое двоеточием.

C#

Наследование полей, свойств и методов

Все поля, свойтва и методы родительского класса доступны и видны в производных классах

```
1   namespace ConsoleApplication.TransportHierarchy;
2
3   public class Transport
4   {
5       public double Spead;
6       public string Name;
7       public int CountPassengers { get;set; }
8
9       public void Moving()
10      {
11          Console.WriteLine("Transport moving");
12      }
13  }
```

```
1   namespace ConsoleApplication.TransportHierarchy;
2
    0 references
3   public class Car : Transport
4   {
        0 references
5       public double EngineVolume { get; set; }
6
        0 references
7       public void Stopping()
8       {
9           Console.WriteLine("Car stopping");
10      }
11  }
```

```
9           Transport transport = new Transport();
10          transport.CountPassengers = 5;
11          transport.Name = "transpot";
12          transport.Spead = 50;
13          transport.Moving();
14
15          Car car = new Car();
16          car.CountPassengers = 5;
17          car.Name = "transpot";
18          car.Spead = 50;
19          car.EngineVolume = 550;
20          car.Moving();
21          car.Stopping();
```

*Исключение прописано на 5 слайде, а так же поля могут быть не видны из-за модификаторов доступа, будет проговорено в теме инкапсуляция

Ключевое слово base

base – с помощью данного ключевого слова мы можем обратиться к свойствам, методам, полям, конструкторам экземпляров родительского класса, работает по аналогии с ключевым словом this. В таком случае, чтобы не было дубляжа кода на помощь приходит ключевое слово base.

*В случае определения в родительском классе конструторов с параметрами (нет конструтора по умолчанию без параметров), необходимо опреледелять/расширять конструторы и в производных классах, т.к. каждый класс в данной ситуации обязан иметь свои конструтор.
В случае определения в родительском классе конструтора по умолчанию, определять/расширять либо нет конструторы в производных решает программист.

C#

Ключевое слово base

```
1   namespace ConsoleApplication.TransportHierarchy;
2
    4 references
3   public class Transport {
        5 references
4       public double Spead;
        6 references
5       public string Name;
        2 references
6       public int CountPassengers { get; set; }
        1 reference
7       public Transport()
8       {
9           Spead = 60;
10          Name = "Transpot";
11      }
12
        1 reference
13      public Transport(double spead, string name)
14      {
15          Spead = spead;
16          Name = name;
17      }
18
        2 references
19  >   public void Moving()···
23  }
```

```
1   namespace ConsoleApplication.TransportHierarchy;
    2 references
2   public class Car : Transport
3   {
        3 references
4       public double EngineVolume { get; set; }
5
        1 reference
6       public Car()
7       {
8           Spead = 60;
9           EngineVolume = 999;
10          Name = "Car";
11      }
12
        0 references
13      public Car(double spead, string name, double engineVolume) : base(spead, name)
14      {
15          EngineVolume = engineVolume;
16      }
17
        1 reference
18  >   public void Stopping()···
22  }
```

Наследование и связь "является"

Обычно наследование выражает связь вида "is a" (является) между базовым классом и одним или несколькими производными классами. Производные классы рассматриваются как специализированные версии базового класса, то есть как подтипы базового класса.

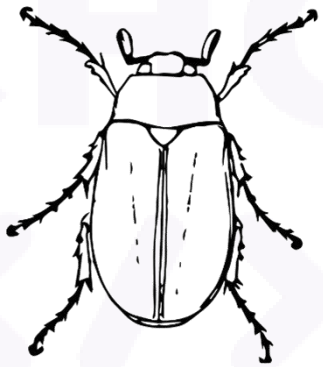
То есть любой производный класс можно неявно преобразовать к версии родительского класса, но не наоборот.

```
using ConsoleApplication.TransportHierarchy;

namespace ConsoleApplication;

0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        Transport car = new Car(); //можно
        bool result = car is Car; //true

        Car transport = new Transport(); //нельзя
    }
}
```

Особенность наследования в C#

В языке C# не поддерживается множественное наследование!!!

Производный класс может наследоваться только от одного родительского класса!!!



Итог

Наследование в C# позволяет вам строить /порождать новый класс (производный класс) из существующего класса (родительского класса). Строя таким способом один класс из другого, вы уменьшаете объем программирования, что, в свою очередь, экономит ваше время.

1. Классы Employee и ProductionWorker. Напишите класс Ernployee (Сотрудник), который содержит атрибуты приведенных ниже данных:
   · имя сотрудника;
   · номер сотрудника.
   Затем напишите класс ProductionWorker (Рабочий), который является подклассом класса Ernployee. Класс ProductionWorker должен содержать атрибуты приведенных ниже данных:
   · номер смены (целое число, к примеру, 1, 2 или 3);
   · ставка почасовой оплаты труда.
   Рабочий день разделен на две смены: дневную и вечернюю. Атрибут смены будет содержать целочисленное значение, представляющее смену, в которую сотрудник работает. Дневная смена является сменой 1, вечерняя смена – сменой 2.
   Напишите соответствующие методыполучатели и методы-модификаторы для каждого класса.
   После того как эти классы будут написаны, напишите программу, которая создает объект класса ProductionWorker и предлагает пользователю ввести данные по каждому атрибуту данных этого объекта. Сохраните данные в объекте и примените в этом объекте методы получатели, чтобы получить эти данные и вывести их на экран.

2. Класс ShiftSupervisor. На некой фабрике начальник смены является штатным сотрудником, который руководит сменой. В дополнение к фиксированному окладу начальник смены получает годовую премию за выполнение его сменой производственного плана. Напишите класс ShiftSupervisor (Начальник смены), который является подклассом класса Ernployee, созданного в задаче по программированию 1. Класс ShiftSupervisor должен содержать атрибут данных для годового оклада и атрибут данных для годовой производственной премии, которую заработал начальник смены. Продемонстрируйте класс, написав программу, которая применяет объект ShiftSupervisor.
3. Классы Person и Customer. Напишите класс Person с атрибутами данных для имени, адреса и телефонного номера человека. Затем напишите класс Customer (Клиент), который является подклассом класса Person. Класс Customer должен иметь атрибут данных для номера клиента и атрибут булевых данных, указывающий, хочет ли клиент быть в списке рассылки или нет. Продемонстрируйте экземпляр класса Custorner в простой программе.

4. Реализуйте класс Singer, который будет наследовать от класса Worker. Класс Worker состоит из: свойства Name, свойства Surname, метода GetFullName(), с помощью которого можно вывести одновременно имя и фамилию; Класс Singer наследует Name, Surname, также состоит из свойств Years, Birth, Style и метода GetMoreInfo(), который выводит информацию о певце: Years, Birth, Style. Необходимо вызвать метод GetMoreInfo
5. Реализовать следующие классы Круг, Треугольник и Квадрат. Каждый экземпляр класса должен содержать свойства Площадь, Периметр и Имя (свойства родительского класса). Создвать массив из фигур и вывести всю информацию о фигурах в консоль. Площадь и периметр должны рассчитываться по математическим формулам.
6. *Расписать классы Самолёт, Поезд, Параход, Велосипед для Транспорта из лекции

ДЗ:

1. https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/tutorials/inheritance
2. https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/base
3. https://www.dotnettricks.com/learn/csharp/understanding-virtual-override-and-new-keyword-in-csharp
4. Задачи
5. Английский