Программирование на C#

Семинар №4

Модуль №2

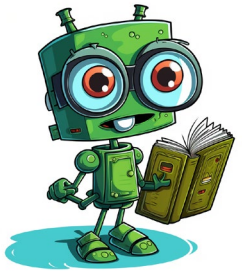
Тема:

Наследование

Виртуальные члены класса

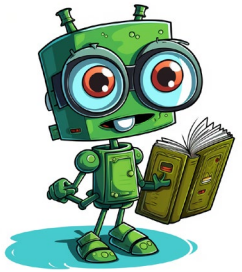
Абстрактные классы

Полезные Материалы



1. Virtual https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/virtual
2. Abstract https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/abstract

Наследование Классов в C#.



```
class <имя_производного_класса> : <имя_базового_класса>
```

Спецификация базового класса

Пример

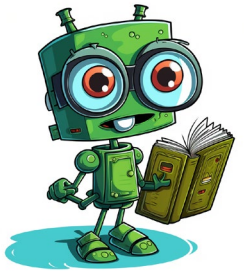
```
class A    {
        public void PrintA()        {
            Console.Write("A");
        }
    }
```

Объявление базового класса

Объявление производного от A класса

```
class B : A    {
        public void PrintB()      {
            Console.Write("B");
        }
    }
```

Виртуальные Методы (сравните код и результат).



```
class A    {
   public void PrintA() { Console.Write("A"); }
}
class B : A    {
    public void PrintA() { Console.Write("B"); }
}
class Program  {
    static void Main()      {
          A objA;
          objA = new B();
          objA.PrintA();
     }
}
```

A

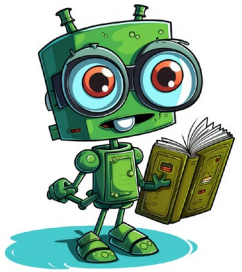
B

Модификатор подмены метода базового класса

Замещение базового метода

```
class A    {
   public virtual void PrintA()
       { Console.Write("A"); }
}
class B : A    {
   public override void PrintA()
       { Console.Write("B"); }
}
class Program    {
   static void Main()      {
          A objA;
          objA = new B();
          objA.PrintA();
       }
    }
```

Demo 01. Пример Наследования.

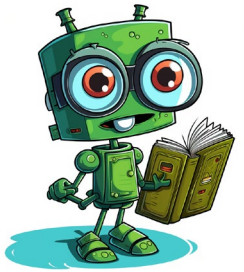


Создайте консольное приложение и разместите в нем объявления классов **A** и **B**, как показано на предыдущих слайдах.

В основной программе создайте массив из 10 ссылок типа **A**, присвойте элементам массива ссылки на случайно создаваемые элементы типов **A** или **B**.
Выведите значения, возвращаемые методом **PrintA**(), вызванным последовательно для всех объектов, адресованных элементами массива.
Выведите значения, возвращаемые методом **PrintA()**, вызванным последовательно только для объектов типа **B**.
Затем выведите значения, возвращаемые методом **PrintA()**, вызванным последовательно только для объектов типа **A**.

Выполнить программу при экранировании и при виртуальности методов **PrintA()**.

Demo 01. Пример Наследования.

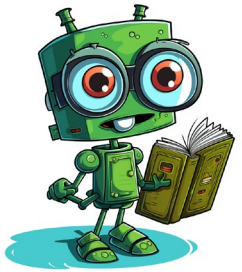


```
static readonly Random random = new Random();

public static void Main(string[] args)
{
    A[] a = new A[10];
    for (int i = 0; i < a.Length; i++)
        a[i] = random.Next(0, 2) < 1 ? new A() : new
B();

    Console.WriteLine("Все объекты: ");
    foreach (A element in a)
        element.PrintA();
    Console.WriteLine();

    Console.WriteLine("Объекты класса B: ");
    foreach (A element in a)
        if (element is B)
            element.PrintA();
    Console.WriteLine();

}
```

```
class A   {
      public void PrintA()      {
          Console.Write("A");
       }
  }
```

```
class B : A   {
      public void PrintB()     {
          Console.Write("B");
       }
  }
```

Demo 02. Виртуальные Методы

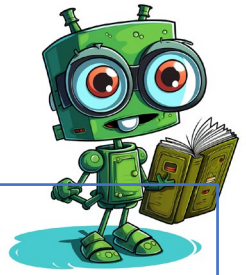


В консольном приложении создайте объекты базового (**Point**) и производных (**Circle** и **Square**) классов, объявите ссылку с типом базового класса **Point**. Последовательно присваивая ссылке «адреса» объектов базового и производных классов, выведите для каждого объекта с помощью ссылки значения свойства «площадь» и вызовите метод **Display()**.

```
static void Main() {
   Point p = new Point();
   p.Display();
   Console.WriteLine("p.Area для Point = " + p.Area);
   p = new Circle(1, 2, 6);
   p.Display();
   Console.WriteLine("p.Area для Circle = " + p.Area);
   p = new Square(3, 5, 8);
   p.Display();
   Console.WriteLine("p.Area для Square = " + p.Area);
}   // end of Main()
```

Получив результаты, уберите из определения классов модификаторы virtual и вновь запустите приложение. Объясните, что произошло.

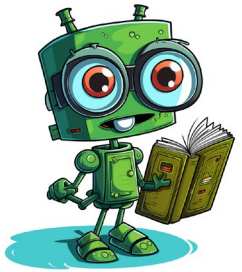
Demo 02.



```
public class Point {
  public double X { get; set; }
  public double Y { get; set; }

  public Point(double x = 0, double y = 0) {
     X = x;
     Y = y;
  }

  public virtual double Area {
    get => 0;
  }

  public virtual void Display(){
    Console.WriteLine($"Point: x={X}, y={Y}");
  }
}
```

```
public class Circle : Point {
  private double _r;

  public Circle(double x = 0, double y = 0, double r = 1)
              : base(x, y) {
                _r = r; }

  public double R
  {
    get => _r;
    set {
      if (value < 0) {
                throw new ArgumentException("Radius of circle
                          can't be negative.");
              }

              _r = value;
            }
  }

  public override double Area => Math.PI * Math.Pow(_r, 2);

  public override void Display() {
            Console.WriteLine($"Circle: x={X}, y={Y}, R={R}");
  } }
```

Demo 02. Виртуальные Методы



```
public class Square : Point {
   private double _h;

   public Square(double x, double y, double h) : base(x, y) {
                  _h = h;
   }

   public double H {
      get => _h;
      set{
         if (value < 0) {
                              throw new ArgumentException("Height of square can't be negative.");
                    }

                  _h = value;
   }}

   public override double Area => Math.Pow(_h, 2);

   public override void Display(){
                  Console.WriteLine($"Square: x={X}, y={Y}, h={_h}");
   }
}
```

ToDo 01 к Demo 02.

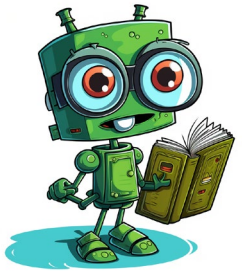
1. В приложении определите статический метод **FigArray()** для создания массива ссылок типа **Point** на случайно формируемые объекты классов **Circle** и **Square**. Количества объектов каждого типа – случайные числа из диапазона [0,10]. Параметры объектов **Circle** и **Square** случайные вещественные значения из диапазона [10;100).

2. В методе Main() создайте массив ссылок типа **Point** и при помощи метода **FigArray()** свяжите ссылки с объектами.

3. Подсчитайте и выведите на экран количество объектов каждого из классов в массиве, и выведите их средние значения площади.

4. Используя метод **Sort()** библиотечного класса **Array**, упорядочите массив по возрастанию площадей фигур, представляемых объектами массива.

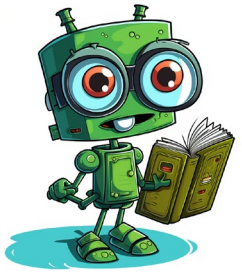
Demo 03. Абстрактные Классы



```
public abstract class Animal{
    public string Name {get; init;}

    protected Animal(string name) {
                Name = name;
    }
    public abstract string Say();
}

public class Cat: Animal {
    public Cat(string name): base(name) { }
    public override string Say() {
        return "Meo-Meo";
    }
}

public class Dog: Animal {
    public Dog(string name): base(name) { }
    public override string Say() {
        return "Gav-Gav";
    }
}
```

```
public class Program
{
public static void Main()
{
Animal animal = new Cat("Pushok");
Console.WriteLine($"{animal.Name} say {animal.Say()}");
animal = new Dog("Gavrik");
Console.WriteLine($"{animal.Name} say {animal.Say()}");
}
}
```

Создание Библиотеки Классов.

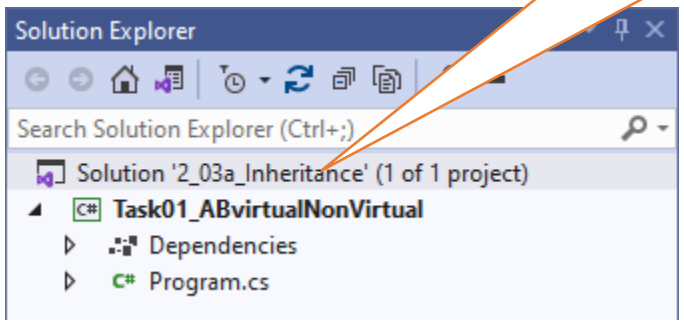


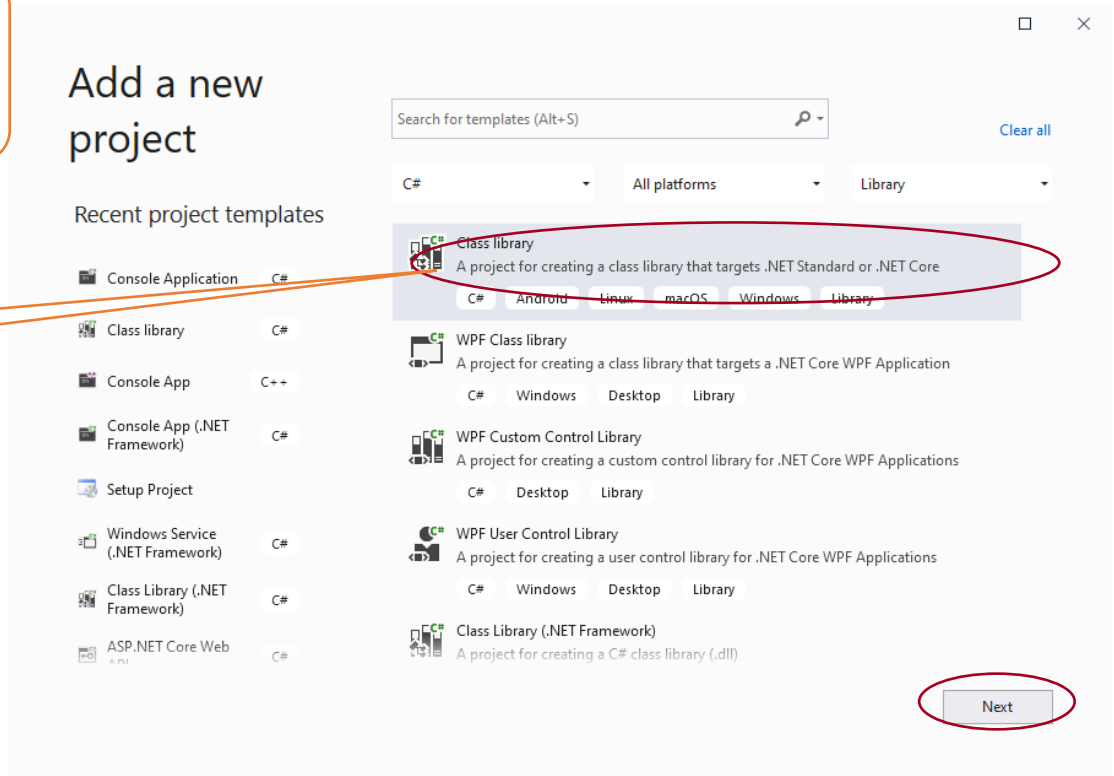
В следующих программах все методы, кроме Main() должны быть размещены в отдельной библиотеке классов.

Шаг 1. В решение добавить новый проект Class Library

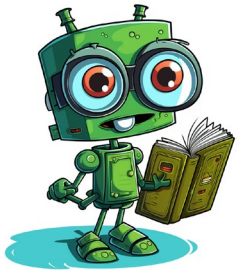
Контекстное меню для «Решение» -> Add new project

Class Library

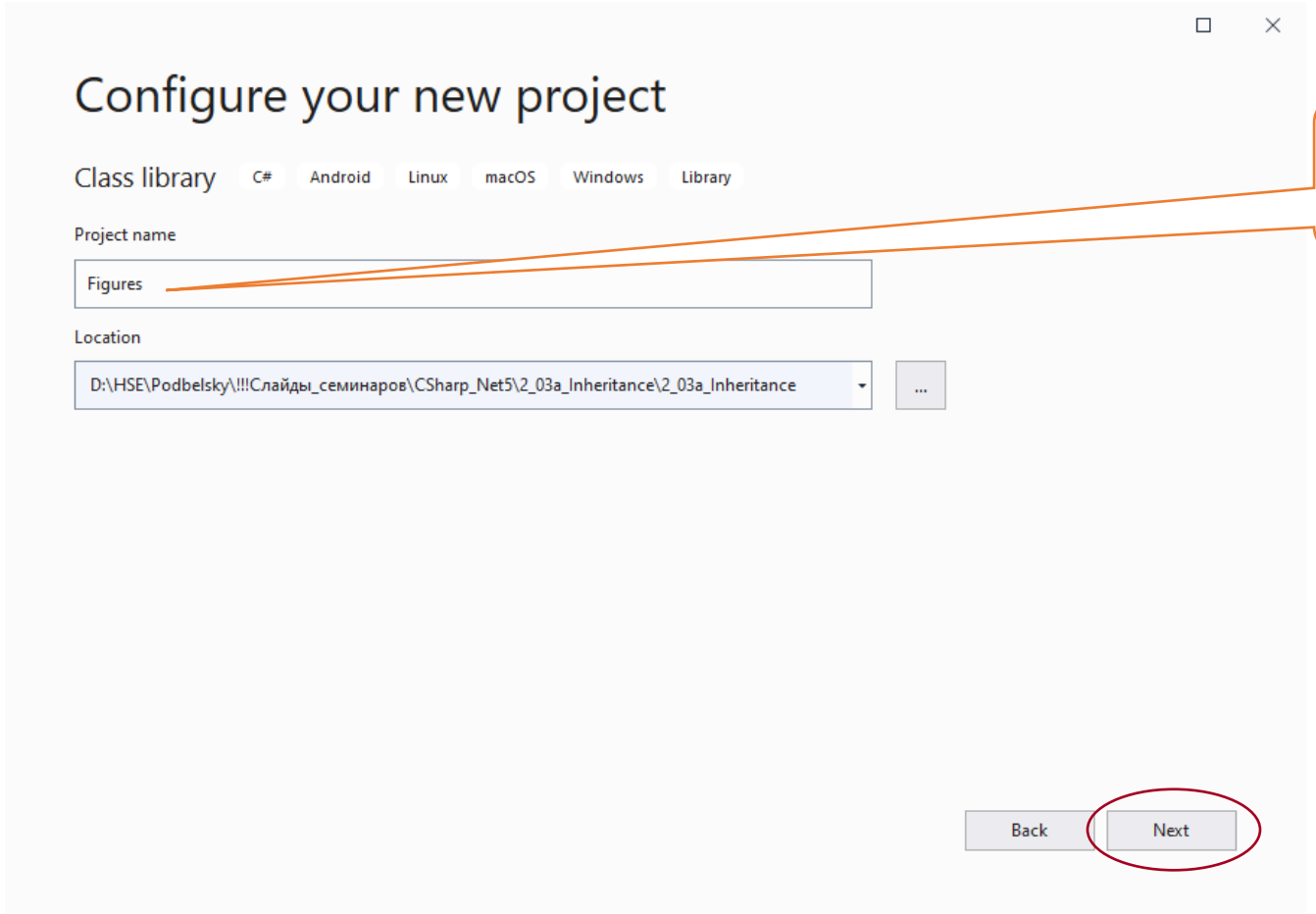




Создание Библиотеки Классов.



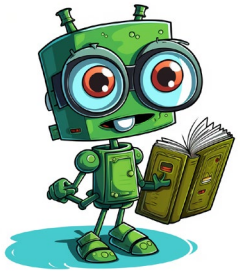
Шаг 2. Задайте имя библиотеки классов



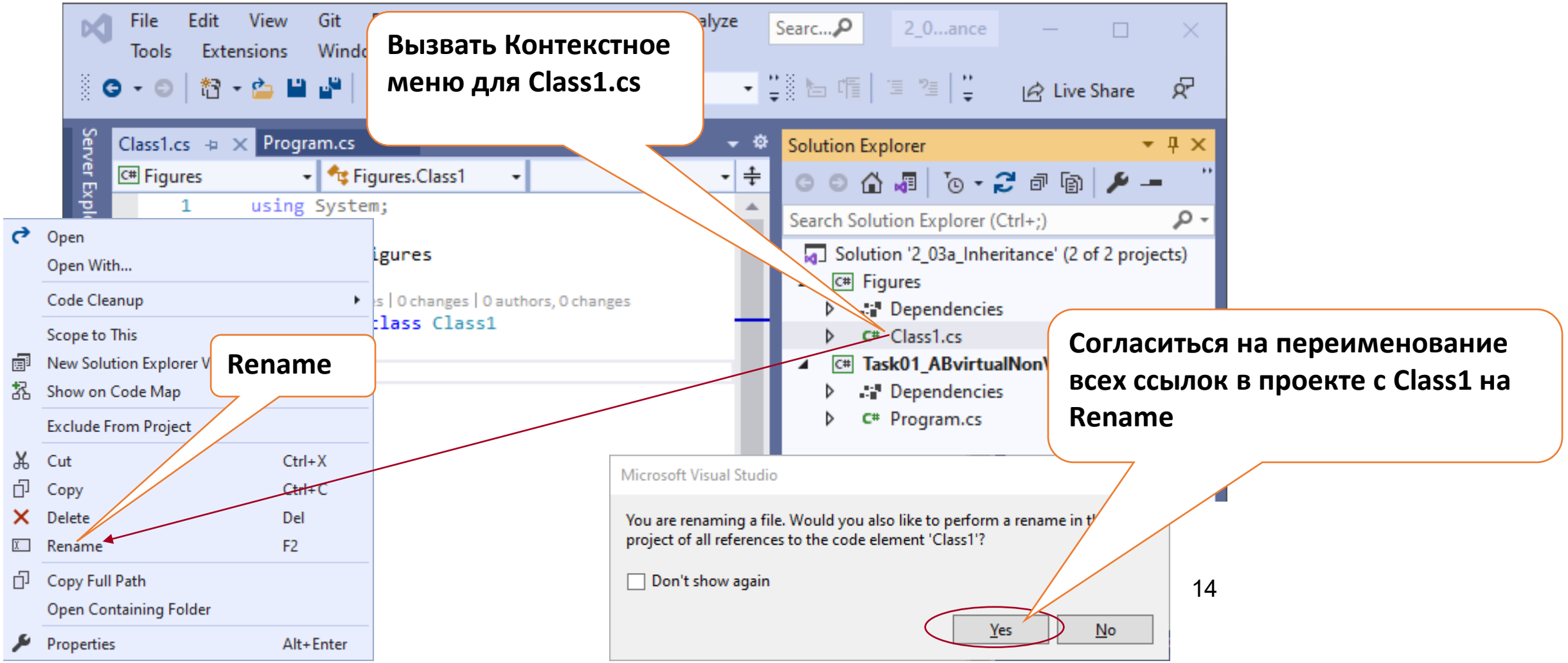
Название библиотеки, например, Figures

Шаг 3. После нажатия Next выберите платформу (.Net 6.0)

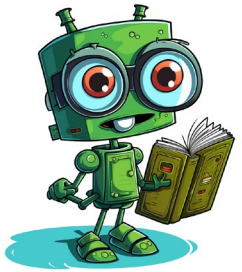
Создание Библиотеки Классов.



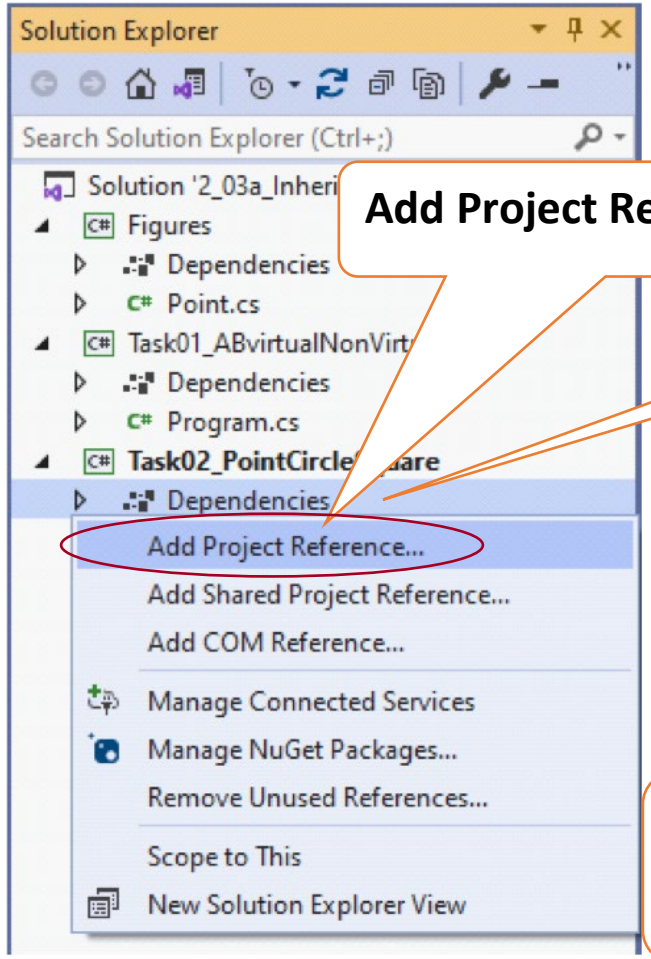
Шаг 4. Переименовать Class1.cs



Создание Библиотеки Классов.



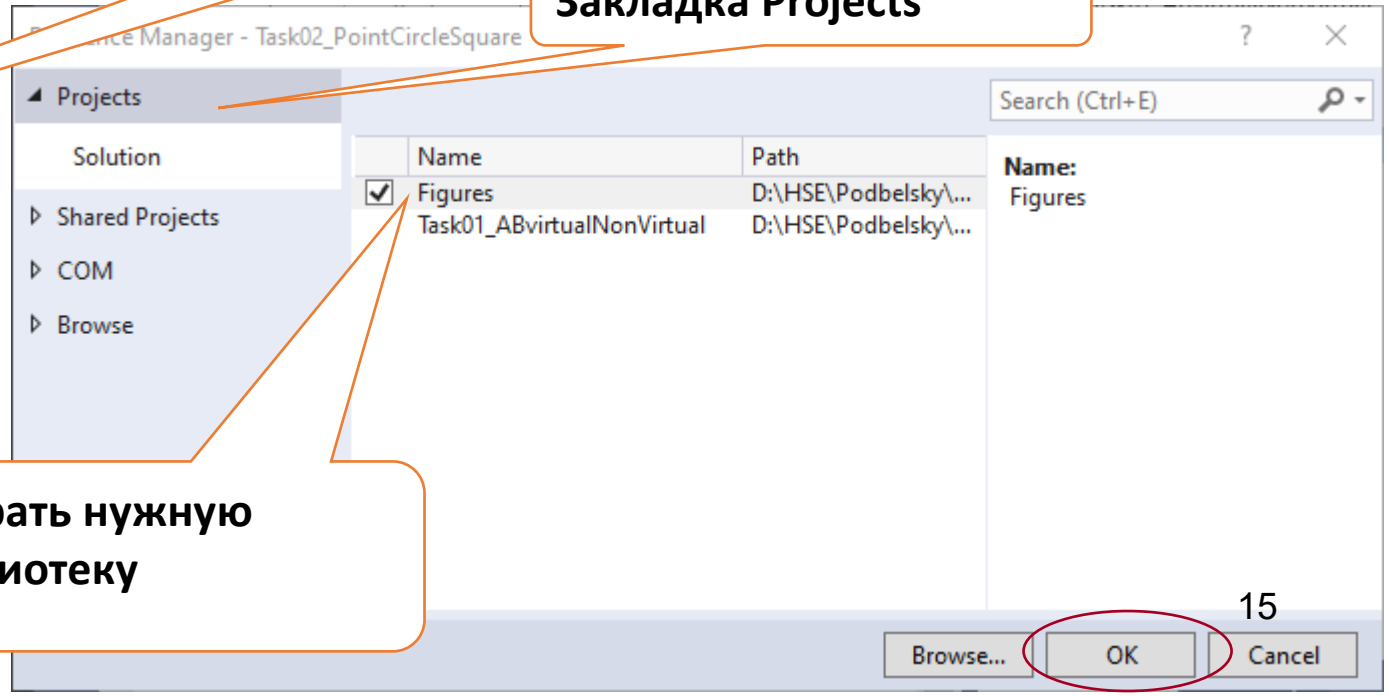
Шаг 5. Для подключения библиотеки классов к проекту необходимо в зависимостях проекта добавить ссылку на библиотеку классов.



Add Project Reference…

Контекстное меню для пункта Dependencies (Зависимости)

Закладка Projects



Выбрать нужную библиотеку

Self 01.

1.Дополните программу из ToDo 1 средствами для упорядочения фигур в массиве по возрастанию расстояний от их центров до начала координат.
2.Модифицируйте код задач, заменив метод Display() методом ToString(). Программа должна функционировать без изменений.
3.Расширьте иерархию наследования из задачи 2 классом треугольников Triangle. В задаче 3 в массив объектов, добавьте объекты типа Triangle.
4.Прочитайте статью «Использование ключевых слов override и new» (https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/knowing-when-to-use-override-and-new-keywords) и реализуйте описанный в ней пример.

Self 02.

Создать библиотеку классов **Lib_Employee**, описывающую классы **Employee** и **SalesEmployee**. За основу взять код соответствующих классов из примера в **Справочник C#, модификатор override**: https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/override

1) Расширьте иерархию классов наследником класса **Employee** – **PartTimeEmployee** (внештатный сотрудник). **PartTimeEmployee** содержит поле **workingDays** – количество рабочих дней. Переопределите метод **CalculatePay()** для расчёта оплаты труда внештатного работника пропорционально количеству рабочих дней. Считать, что базовая оплата устанавливается за 25 рабочих дней в месяц.
2) В консольном приложении создать массив ссылок типа **Employee**. Каждую ссылку связать с объектом **SalesEmployee** или **PartTimeEmployee**. На экран вывести группы **SalesEmployee** и **PartTimeEmployee**, упорядоченные по убыванию заработной платы.

Self 03.

Создать библиотеку классов **Lib_Shape**, описывающих геометрические фигуры и тела. За основу взять код классов **Shape**, **Circle**, **Cylinder** и **Sphere** из примера в **Справочник C#, модификатор virtual**:
https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/virtual

1) В консольном приложении создать массив ссылок на объекты типа **Shape**, связать их с **N1** объектами **Circle**, **N2** объектами **Cylinder** и **N3** объектами **Sphere**. Количество фигур – случайные числа [**3**;**5**], параметры – произвольные случайные значения.
2) Организовать вывод площадей поверхностей всех фигур.
3) Организовать вывод фигур с названиями фигуры или геом. тела и их площадей поверхности. Использовать **is**.
4) * При помощи метода **Array.Sort()** упорядочить объекты массива по группам: окружности, цилиндры, сферы.