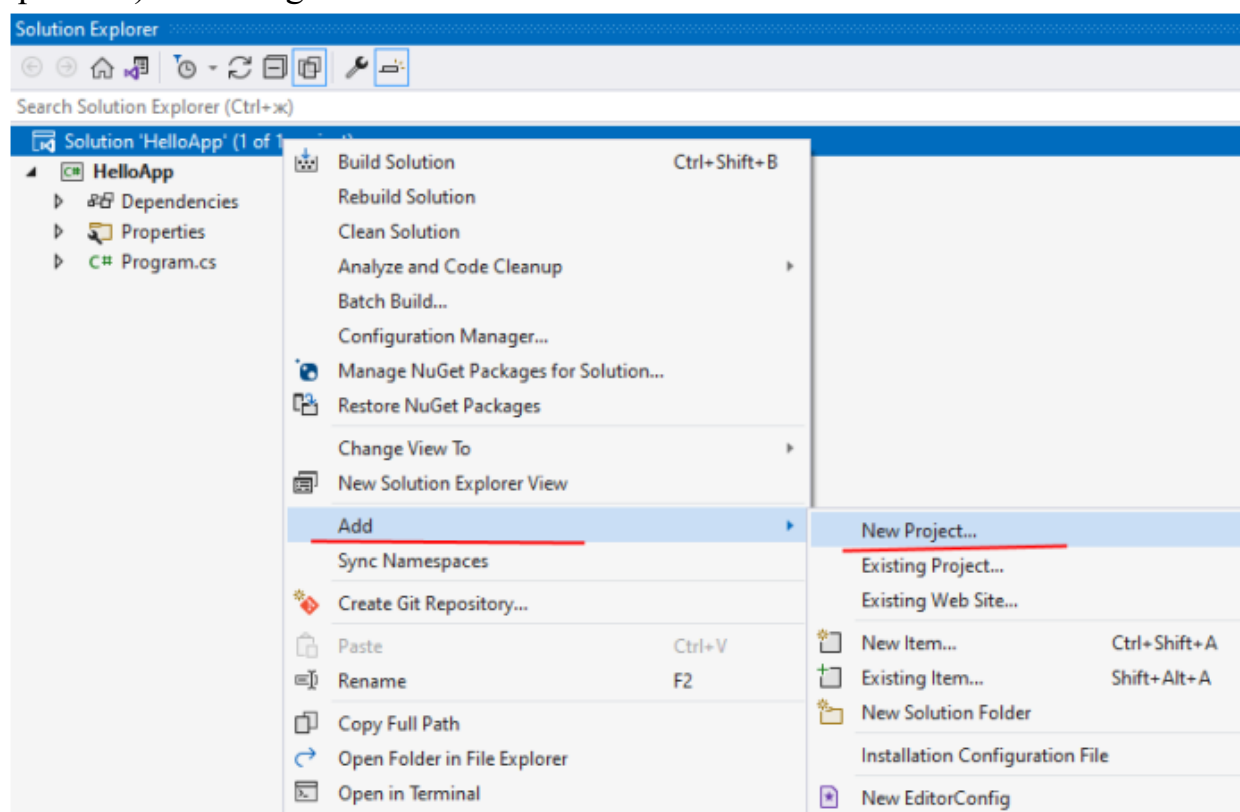


28-Amaliy mashg'ulot. Dll kutubxona yaratish va bog'lash.

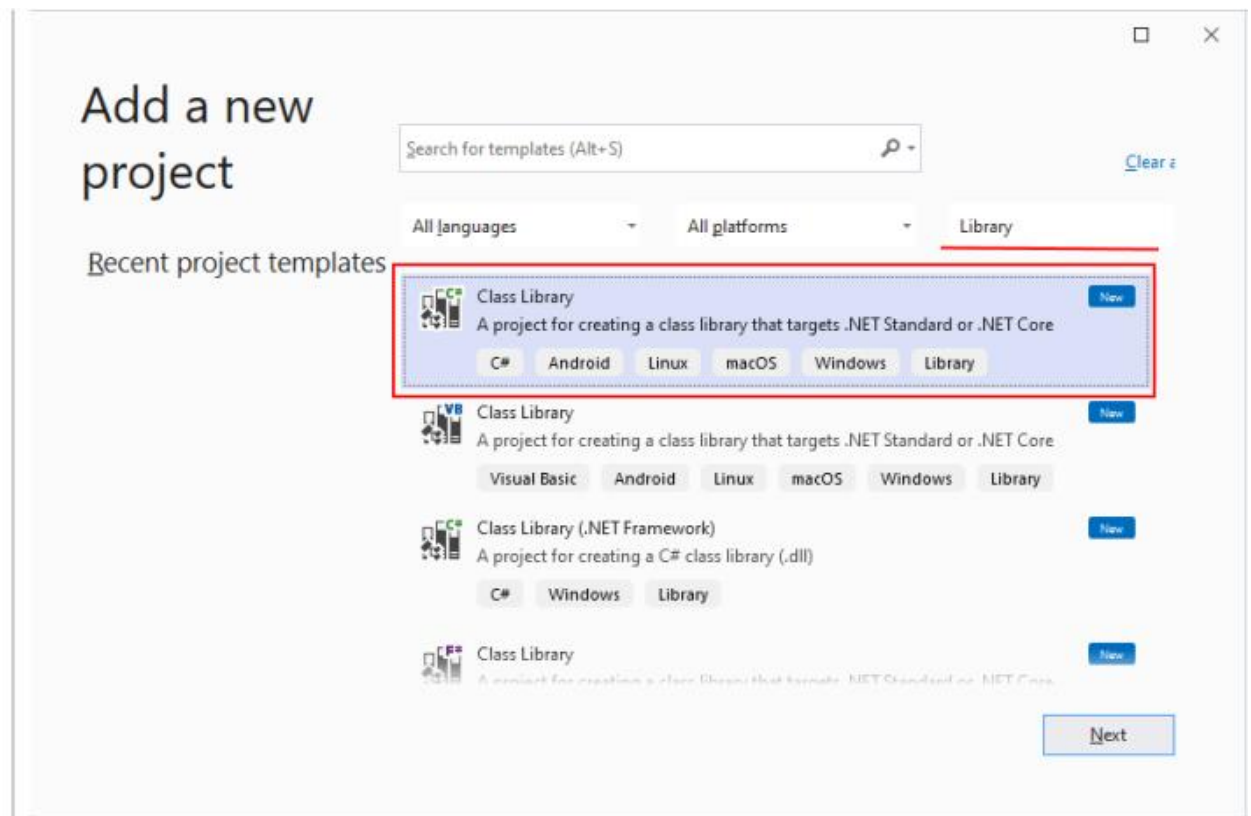
Ko'pincha, turli sinflar va tuzilmalar DLL fayllariga kompilyatsiya qilinadigan va keyin boshqa loyihalarga kiritilishi mumkin bo'lgan alohida kutubxonalar sifatida paketlanadi. Buning yordamida biz bir xil funktsiyani sinf kutubxonasi shaklida aniqlashimiz va uni turli loyihalarga kiritishimiz yoki foydalanish uchun boshqa ishlab chiquvchilarga topshirishimiz mumkin.

Keling, sinf kutubxonasini yaratamiz va ulaymiz.

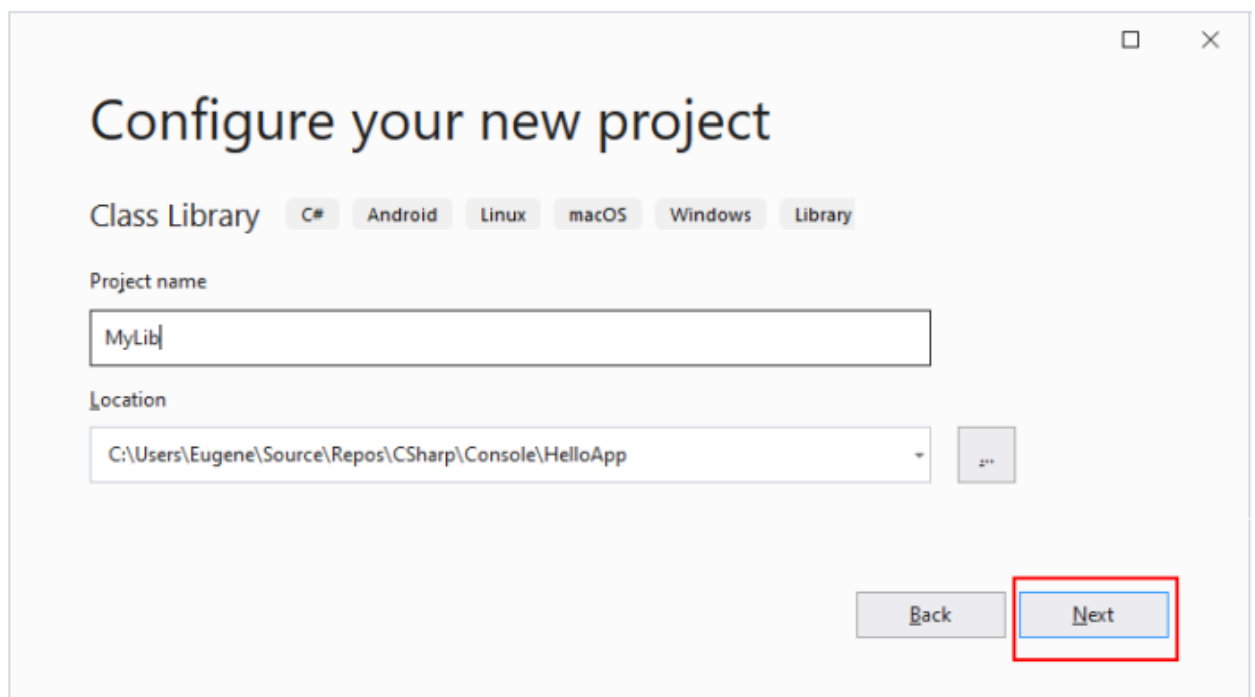
Oldingi mavzularda yaratilgan kabi mavjud C# konsol ilova loyihasini olaylik. Loyiha tuzilmasida yechim nomini sichqonchaning o'ng tugmasi bilan bosing va keyin paydo bo'lgan kontekst menyusida Qo'shish -> Yangi loyiha... (Yangi loyiha qo'shish) ni tanlang.



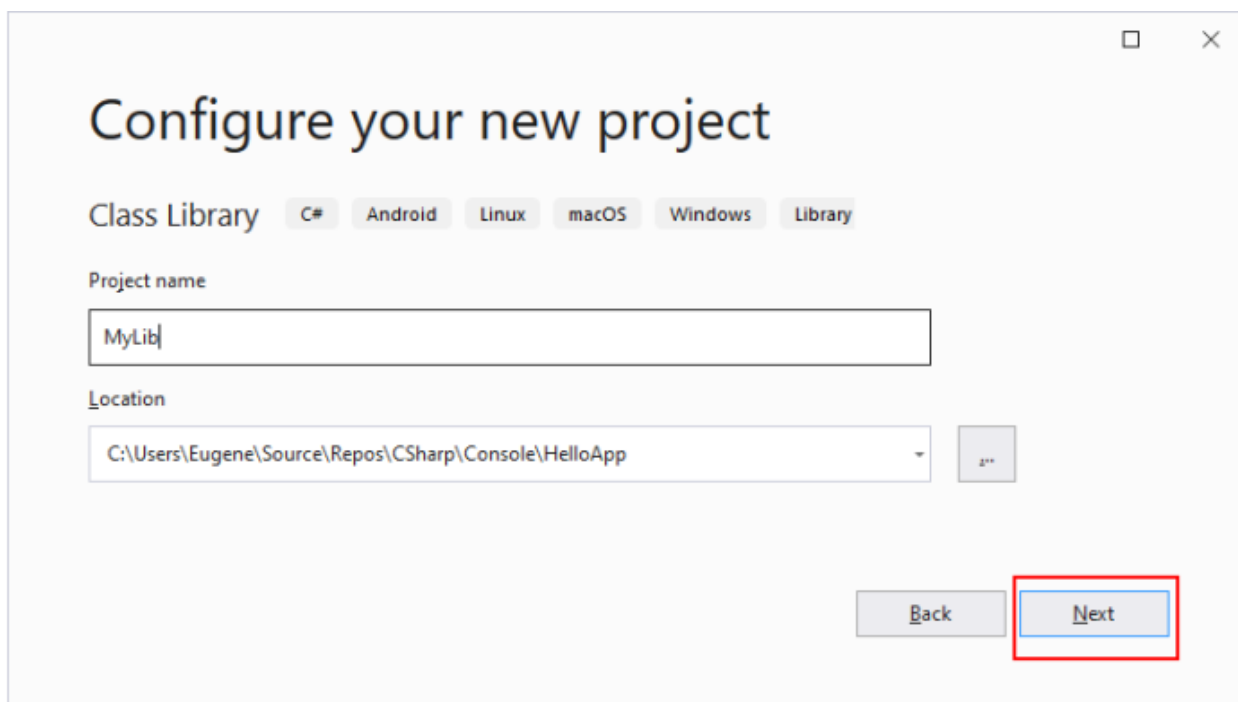
Keyin, loyiha shablonlari ro'yxatida Class Library elementini toping:



Keyin biz yangi loyihaga qandaydir nom beramiz, masalan, MyLib:



Keyin sizdan kutubxona yaratilayotgan framework versiyasini tanlashingiz so‘raladi:



Configure your new project

Class Library C# Android Linux macOS Windows Library

Project name

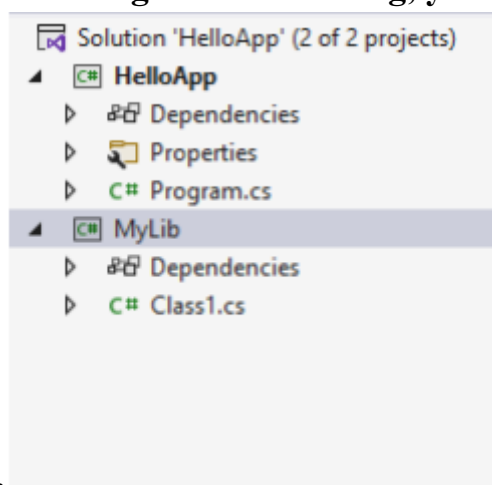
MyLib

Location

C:\Users\Eugene\Source\Repos\CSharp\Console\HelloApp

Back Next

.NET 6.0 (uzoq muddatli qo‘llab-quvvatlash) ni shu yerda qoldiring va Yaratish tugmasini bosning. Shundan so‘ng, yechimga MyLib nomi bilan yangi



loyiha qo‘shiladi:

Odatiy bo‘lib, yangi loyiha Class1.cs faylida bitta bo‘sh Class1ga ega. Biz bu faylni olamiz

o‘chirish yoki xohlagancha nomini o‘zgartirish.

Masalan, Class1.cs faylining nomini Person.cs, Class1 sinfini esa Person deb o‘zgartiramiz. Sinfda aniqlang

Shaxs - eng oddiy kod:

```
public class Person
{
```

```
string name;
```

```
public Person(string name)
```

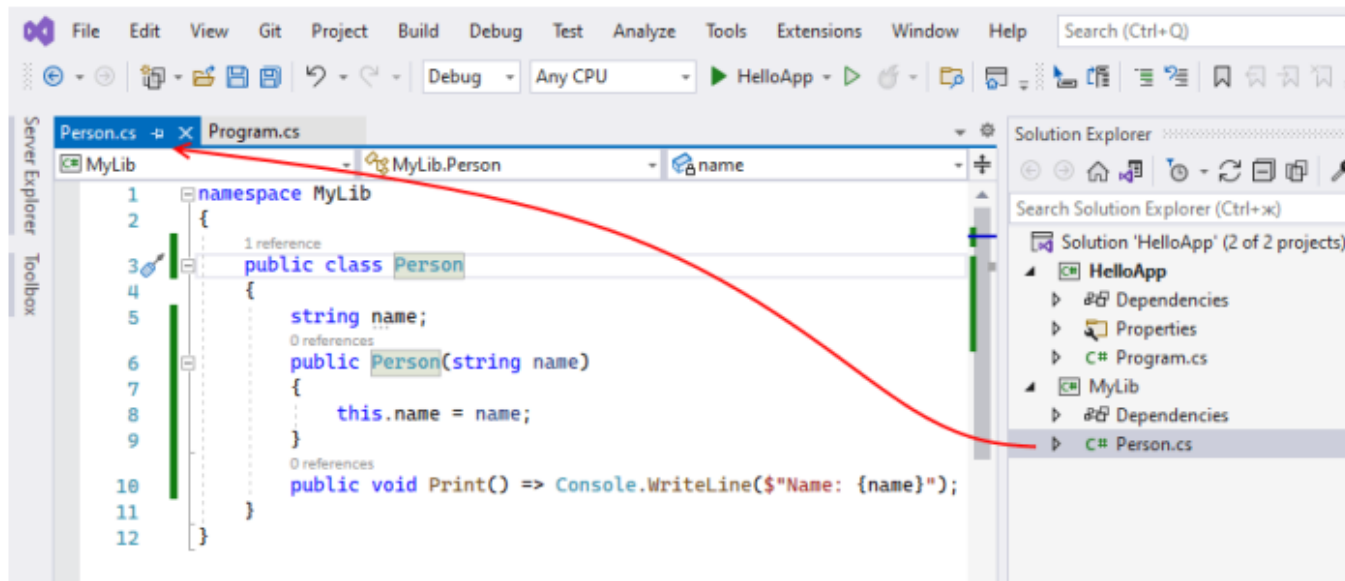
```
{
```

```
    this.name = name;
```

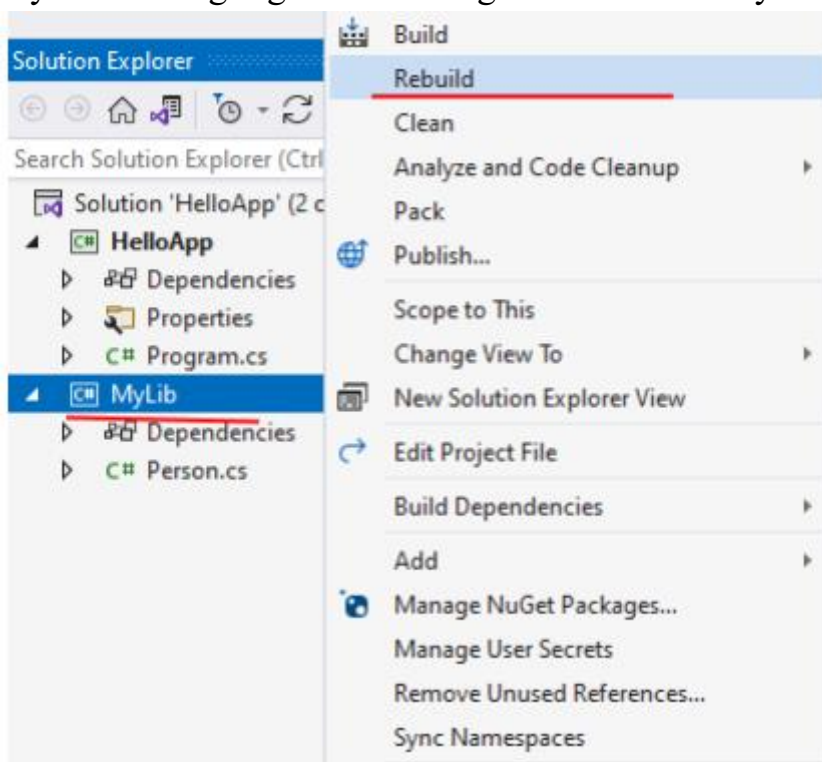
```
}
```

```
public void Print() => Console.WriteLine($"Name: {name}");
```

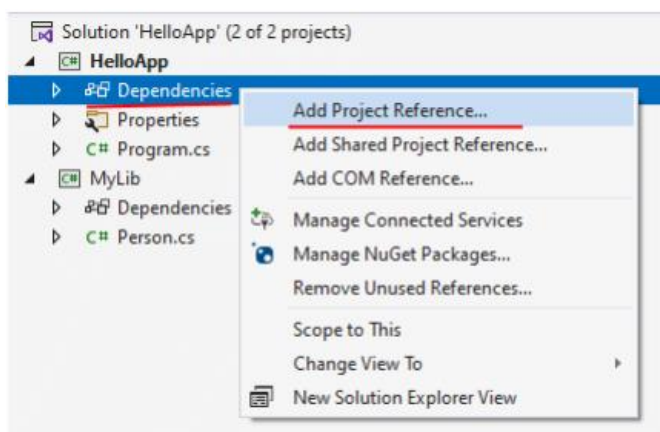
```
}
```



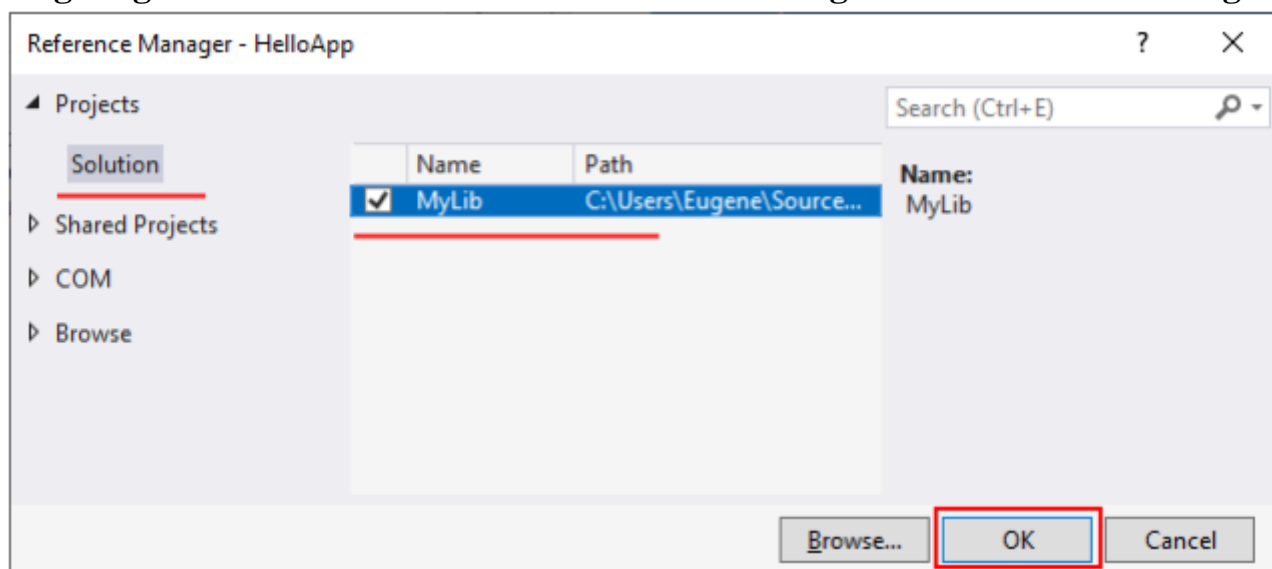
Endi sinf kutubxonasi tuzamiz. Buni amalga oshirish uchun sinf kutubxonasi loyihasini o'ng tugmasini bosing va kontekst menyusidan Qayta qurish-ni tanlang:



Sinf kutubxonasini kompilyatsiya qilgandan so‘ng, bin/Debug/net6.0 katalogidagi loyiha papkasida biz kompilyatsiya qilingan dll faylni (MyLib.dll) topishimiz mumkin. Keling, uni asosiy loyihaga ulaymiz. Buni amalga oshirish uchun asosiy loyihada bog‘liqliklar tuguniga sichqonchani o‘ng tugmachasini bosing va kontekst menyusidagi Loyiha ma‘lumotnomasini qo‘shish... bandini tanlang:



Keyin biz kutubxonalarni qo‘shish oynasini ko‘ramiz. Ushbu oynada, joriy yechim loyihalaridagi barcha sinf kutubxonalarini ko‘rish imkonini beruvchi Yechim elementini tanlang, kutubxonamiz yonidagi katakchani belgilang va OK tugmasini bosing:



Agar kutubxonamiz to‘satdan bizning yechimimizdagi biron bir loyiha bilan bog‘liq bo‘lmagan dll faylni taqdim etsa, u holda "Browse" tugmasi yordamida biz dll faylining joylashuvini topamiz va uni ham qo‘shamiz.

Kutubxonani asosiy loyihaga muvaffaqiyatli qo'shgandan so'ng, biz Program.cs faylini sinf kutubxonasidagi Person sinfidan foydalanishi uchun o'zgartiramiz:

```
using MyLib; // sinf kutubxonasidan nom maydonini o'z ichiga oladi
```

```
Person tom = new("Tom");
```

```
tom.Print(); // Name: Tom
```

28-Amaliy mashg'ulot topshiriqlari.

Har bir talaba quyida berilgan topshiriq misollarini alohida DLL kutubxona shaklida yarating va sinfdan obekt olib uni ishlating.

№	Masala sharti
1.	Talaba sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
2.	Avtomashina sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
3.	Mijoz sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
4.	Tovar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
5.	Avia reys sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
6.	Dars sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
7.	Kitob sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
8.	Kompyuter sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
9.	Odam sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
10.	O'quv xonasi nomli sinfni yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
11.	Olimlar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
12.	O'qituvchi sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
13.	Telefon sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
14.	Shahar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.

15.	Metro sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
16.	Nuqta sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
17.	Uchburchak sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
18.	To'rtburchak sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
19.	Doira sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
20.	Aylana sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
21.	Kub sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
22.	Ko'pburchak sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
23.	Matematika sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
24.	Hayvonlar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
25.	Marketlar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
26.	Bekat sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
27.	Geometriya sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
28.	Printerlar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
29.	Modemlar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
30.	Kompleks sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.

29-Amaliy mashg'ulot. Unar va binar operatorlarni qayta yuklash.

Usullar bilan bir qatorda biz operatorlarni ortiqcha yuklashimiz ham mumkin.

Masalan, bizda quyidagi Counter klassi bor deylik:

```
class Counter
{
    public int Value { get; set; }
}
```

Ushbu sinf ba'zi bir hisoblagichlarni ifodalaydi, ularning qiymati Value xususiyatida saqlanadi.

Aytaylik, bizda hisoblagich sinfining ikkita ob'ekti bor - biz ularni solishtirish yoki qo'shish istagan ikkita hisoblagichni Value xususiyati asosida standart taqqoslash va qo'shish operatsiyalari yordamida:

```
Counter c1 = new Counter { Value = 23 };
```

```
Counter c2 = new Counter { Value = 45 };
```

```
bool result = c1 > c2;
```

```
Counter c3 = c1 + c2;
```

Ammo hozirgi vaqtda Counter moslamalari uchun na taqqoslash operatsiyasi, na qo'shish operatsiyasi mavjud emas. Ushbu operatsiyalar bir qator ibtidoiy turlari uchun ishlatilishi mumkin. Masalan, sukut bo'yicha biz raqamli qiymatlarni

qo'shishimiz mumkin, ammo kompilyator murakkab turdagi ob'ektlar - sinflar va tuzilmalarni qanday qo'shishni bilmaydi. Va buning uchun biz kerakli operatorlarni ortiqcha yuklashimiz kerak.

Operatorning ortiqcha yuklanishi biz operatorni aniqlamoqchi bo'lgan ob'ektlar uchun sinfda maxsus usulni belgilashdan iborat:

```
public static qaytarish_tip operator oneparop(parametrlar)
{ }
```

Ushbu usul umumiy statik modifikatorlarga ega bo'lishi kerak, chunki haddan tashqari yuklangan operator ushbu sinfning barcha ob'ektlari uchun ishlatiladi. Keyingi qaytish turining nomi. Qaytish turi biz ob'ektlarni olishni istagan turni ifodalaydi. Masalan, ikkita Counter moslamasini qo'shish natijasida biz yangi Counter moslamasini olishni kutmoqdamiz. Va ikkalasini taqqoslash natijasida biz bool tipidagi ob'ektni olishni istaymiz, bu shartli ifoda to'g'ri yoki yolg'on ekanligini ko'rsatadi. Ammo vazifaga qarab, qaytarish turlari har qanday bo'lishi mumkin.

Keyin usul nomi o'rniga operator kalit so'zi va operatorning o'zi mavjud. Va keyin parametrlar qavs ichida keltirilgan. Ikkilik operatorlar ikkita parametрни, unaryali operatorlar bitta parametрни oladi. Va har qanday holatda, parametrlardan biri operatorni aniqlaydigan sinf yoki tuzilmani - turini ko'rsatishi kerak.

Masalan, Counter klassi uchun bir qator operatorlarni ortiqcha yuklaylik:

```
class Counter
{
    public int Value { get; set; }

    public static Counter operator +(Counter c1, Counter c2)
    {
        return new Counter { Value = c1.Value + c2.Value };
    }
    public static bool operator >(Counter c1, Counter c2)
    {
        return c1.Value > c2.Value;
    }
    public static bool operator <(Counter c1, Counter c2)
    {
        return c1.Value < c2.Value;
    }
}
```

Barcha ortiqcha yuklangan operatorlar ikkilik bo'lganligi sababli, ya'ni ular ikkita ob'ekt ustida bajariladi, keyin har bir ortiqcha yuk uchun ikkita parametr bo'ladi.

Qo'shish amalida biz Counter sinfining ikkita moslamasini qo'shmoqchimiz, operator shu sinfning ikkita ob'ektini oladi. Qo'shish natijasida biz yangi Counter ob'ektini olishni istaganimiz uchun, bu sinf qaytish turi sifatida ham ishlatiladi. Ushbu operatorning barcha harakatlari Value xususiyati ikkala parametrning Value xususiyatining qiymatlarini birlashtirgan yangi ob'ektni yaratishga qisqartiriladi:

```
public static Counter operator +(Counter c1, Counter c2)
{
    return new Counter { Value = c1.Value + c2.Value };
}
```

Ikkala taqqoslash operatorlari ham qayta aniqlandi. Agar biz ushbu taqqoslash operatsiyalaridan birini bekor qilsak, unda biz ushbu operatsiyalarning ikkinchisini ham bekor qilishimiz kerak. Taqqoslash operatorlari o'zlari Value xususiyatlarining qiymatlarini taqqoslaydilar va taqqoslash natijalariga qarab "true" yoki "false" qiymatlarini qaytaradilar.

Endi biz dasturda ortiqcha yuklangan operatorlardan foydalanamiz:

```
using System;
namespace Qayta
{
    class Counter
    {
        public int Value { get; set; }

        public static Counter operator +(Counter c1, Counter c2)
        {
            return new Counter { Value = c1.Value + c2.Value };
        }
        public static bool operator >(Counter c1, Counter c2)
        {
            return c1.Value > c2.Value;
        }
        public static bool operator <(Counter c1, Counter c2)
        {
            return c1.Value < c2.Value;
        }
    }
}
```

```

class Program
{
    static void Main(string[] args)
    {
        Counter c1 = new Counter { Value = 23 };
        Counter c2 = new Counter { Value = 45 };
        bool result = c1 > c2;
        Console.WriteLine(result); // false

        Counter c3 = c1 + c2;
        Console.WriteLine(c3.Value); // 23 + 45 = 68

        Console.ReadKey();
    }
}

```

Shuni e'tiborga loyiqlik, aslida operatorning ta'rifi bu usul ekan, biz ushbu usulni ortiqcha yuklay olamiz, ya'ni buning uchun boshqa versiyasini yaratamiz. Masalan, Counter sinfiga boshqa operator qo'shaylik:

```

public static int operator +(Counter c1, int val)
{
    return c1.Value + val;
}

```

Ushbu usul Value xususiyati va sonini qo'shib, ularning yig'indisini qaytaradi. Va biz ushbu operatoridan ham foydalanishimiz mumkin:

```

Counter c1 = new Counter { Value = 23 };
int d = c1 + 27; // 50
Console.WriteLine(d);

```

Shuni yodda tutish kerakki, ortiqcha yuk paytida operatorga parametrlar orqali uzatiladigan ob'ektlar o'zgartirilmasligi kerak. Masalan, biz Counter klassi uchun o'sish operatorini aniqlay olamiz:

```

public static Counter operator ++(Counter c1)
{
    c1.Value += 10;
    return c1;
}

```

Operator unary bo'lgani uchun u faqat bitta parametрни oladi - bu operator aniqlangan sinf ob'ekti. Ammo bu o'sishning noto'g'ri ta'rifi, chunki operator o'z parametrlari qiymatlarini o'zgartirmasligi kerak.

Va o'sish operatorining yanada to'g'ri yuklanishi quyidagicha ko'rinadi:

```
public static Counter operator ++(Counter c1)
{
    return new Counter { Value = c1.Value + 10 };
}
```

Ya'ni, Value xususiyatidagi oshirilgan qiymatni o'z ichiga olgan yangi ob'ekt qaytariladi.

Shu bilan birga, biz prefiks va postfix qo'shimchalari (shuningdek, pasayishlar) uchun alohida operatorlarni belgilashimiz shart emas, chunki bitta dastur ikkala holatda ham ishlaydi.

Masalan, prefiksni oshirish amalidan foydalanamiz:

```
Counter counter = new Counter() { Value = 10 };
Console.WriteLine($"{counter.Value}"); // 10
Console.WriteLine($"{(++counter).Value}"); // 20
Console.WriteLine($"{counter.Value}"); // 20
```

Endi biz postfix o'sishidan foydalanamiz:

```
Counter counter = new Counter() { Value = 10 };
Console.WriteLine($"{counter.Value}"); // 10
Console.WriteLine($"{(counter++).Value}"); // 10
Console.WriteLine($"{counter.Value}"); // 20
```

Shuni ham ta'kidlash joizki, biz haqiqiy va noto'g'ri operatorlarni bekor qilishimiz mumkin. Masalan, ularni Counter sinfida aniqlaymiz:

```
class Counter
{
    public int Value { get; set; }

    public static bool operator true(Counter c1)
    {
        return c1.Value != 0;
    }
    public static bool operator false(Counter c1)
    {
        return c1.Value == 0;
    }
}
```

```
}
```

```
// остальное содержимое класса
```

```
}
```

Ushbu turdagi operatorlardan biz shartli ravishda foydalanmoqchi bo'lganimizda, ushbu operatorlar haddan tashqari yuklanadi. Masalan:

```
Counter counter = new Counter() { Value = 0 };
```

```
if (counter)
```

```
    Console.WriteLine(true);
```

```
else
```

```
    Console.WriteLine(false);
```

Operatorlarni haddan tashqari yuklaganda, barcha operatorlar haddan tashqari yuklanmasligi mumkinligini yodda tuting. Xususan, biz quyidagi operatorlarni ortiqcha yuklay olamiz:

unary operatorlari +, -,!, ~, ++, -

ikkilik operatorlar +, -, *, /,%

taqqoslash operatsiyalari == ,!=, <,>, <=,> =

mantiqiy operatorlar &&, ||

29-Amaliy mashg'ulot topshiriqlari.

Geometrik shakllar uchun unar hamda binar amallar yuklanishi bajaring.

Masalan: Kvadrat, kub, shar, doira va h.k.

30-Amaliy mashg'ulot. Sinfning konstanta maydonlari. Qism sinflar bilan ishlash.

Nazariy qism.

Sinf maydonlari oddiy sinf darajasidagi o'zgaruvchidir. Biz allaqachon o'zgaruvchilarni ko'rib chiqdik - ularning e'lon qilinishi va ishga tushirilishi. Biroq, biz hali ham ba'zi fikrlarga, masalan, doimiy va o'qiladigan maydonlarga tegmaganmiz.

Konstanta

Konstantalar quyidagi xususiyatlar bilan tavsiflanadi:

Konstantani aniqlashda uni boshlash kerak

Belgilanganidan so'ng doimiyning qiymatini o'zgartirish mumkin emas

Konstantalar dasturda o'zgartirilmasligi kerak bo'lgan qiymatlarni tavsiflash uchun mo'ljallangan. Const kalit so'zi sobitlarni aniqlash uchun ishlatiladi:

```
const double PI = 3.14;
```

```
const double E = 2.71;
```

Konstantalardan foydalanganda, ularni faqat bir marta e'lon qilishimiz mumkinligini va ular kompilyatsiya vaqti bilan belgilanishi kerakligini unutmang.

```
class MathLib
```

```
{  
    public const double PI = 3.141;  
    public const double E = 2.81;  
    public const double K; // Xato, doimiy qiymat uzlashtirmagan }
```

```
class Program
```

```
{  
    static void Main(string[] args)  
    {  
        MathLib.E = 3.8; // Xato, konstanta qiymat uzgarmaydi  
    }  
}
```

Shuningdek, doimiyni chaqirish uchun sintaksisga e'tibor bering. Bu to'g'ridan-to'g'ri statik maydon bo'lgani uchun, unga murojaat qilish uchun sinf nomidan foydalanishingiz kerak.

```
class MathLib
```

```
{  
    public const double PI = 3.141;  
}
```

```
class Program
```

```

{
    static void Main(string[] args)
    {
        Console.WriteLine(MathLib.PI);
    }
}

```

Shuni yodda tutish kerakki, biz statik modifikator bilan doimiylikni eʼlon qila olmaymiz. Ammo bu aslida hech qanday maʼnoga ega emas.

Doimiylikni sinf darajasida ham, usul ichida ham aniqlash mumkin:

class MathLib

```

{
    public double GetCircleArea(double radius)
    {
        const double PI = 3.141;
        return PI * radius * radius;
    }
}

```

Oʻqish maydonlari

Oʻqish maydonlari eʼlon qilinganida yoki sinf darajasida ishga tushirilishi yoki konstruktorda boshlanishi va oʻzgartirilishi mumkin. Siz ularning qiymatini boshqa joyda boshlashingiz yoki oʻzgartirishingiz mumkin emas, faqat ularning qiymatini oʻqishingiz mumkin.

Oʻqilgan maydon readonly kalit soʻzi bilan eʼlon qilinadi:

class MathLib

```

{
    public readonly double K = 23; // mana shunday uzlashtirish mumkin

    public MathLib(double _k)
    {
        K = _k; // oʻqish maydoni kompilyatsiyadan keyin konstruktorda ishga
        tushirilishi yoki oʻzgartirilishi mumkin    }
    public void ChangeField()
    {
        // bunday mumkinmas
        //K = 34;
    }
}

```

Konstantalarni taqqoslash

Konstantalar kompilyatsiya vaqtida, o'qiladigan maydonlar esa ishlash vaqtida aniqlanishi mumkin.

Shunga ko'ra, doimiyni faqat aniqlanganda boshlash mumkin.

O'qish maydonini belgilashda yoki sinf konstruktorida boshlash mumkin.

Doimiy holatlar statik modifikatorni ishlata olmaydi, chunki ular allaqachon statikdir. O'qish maydonlari statik yoki statik bo'lmagan bo'lishi mumkin.

Tuzilmalarni o'qing

C# da o'qish uchun maydonlardan tashqari siz o'qish uchun tuzilmalarni belgilashingiz mumkin. Buni amalga oshirish uchun ularning oldiga faqat o'qish o'zgaruvchisi keladi:

```
readonly struct User { }
```

Bunday tuzilmalarning o'ziga xos xususiyati shundaki, ularning barcha maydonlari o'qiladigan maydonlar bo'lishi kerak:

```
readonly struct User
```

```
{  
    public readonly string name;  
    public User(string name)  
    {  
        this.name = name;  
    }  
}
```

Xuddi shu narsa faqat o'qilishi kerak bo'lgan xususiyatlarga tegishli:

```
readonly struct User
```

```
{  
    public readonly string Name { get; } // readonly ni ko'rsatish shart emas  
    public int Age { get; } // xususiyat faqat o'qish uchun  
    public User(string name, int age)  
    {  
        this.Name = name;  
        this.Age = age;  
    }  
}
```


30-Amaliy mashg'ulot uchun topshiriqlar.

Topshiriq : Topshiriqlarning barcha variantlari uchun belgilangan ikkita maydon (1-maydon, 2-maydon) va uchta metod bilan yarating:

- obyektни ishga tushirish uchun konstruktor;
- ob'ekt haqidagi ma'lumotlarga ega bo'lgan qatorni shakllantirish metodi;
- sinf uchun qism sinf yarating har bir birida alohida metod shakllansin;
- yaratilgan qism sinflardan obekt oling va metodlarni ishlating.
- topshiriq ichida konstanta qiymatlarni kiriting.

№ var .	Maydon 1	Maydon 2	Maydonga ishlov berish metodi
1	Nominal kupyura lar (1, 2, 5, 10 va h.k)	Kupyura miqdori	Kupyuralar summasini hisoblang
2	Nominal tangalar (1, 2, 5, 10 va h.k)	Tangalar miqdori	Tangalar summasini hisoblang
3	Mahsulotning narxi	Mahsulotlar soni	Buyumning umumiy narxini hisoblang
4	100 g mahsulotning kaloriya miqdori	Mahsulot og'irligi grammda	Mahsulotning umumiy kaloriya miqdorini hisoblang
5	Haqiqiy son - diapazonning chap chegarasi	Haqiqiy son - diapazonning o'ng chegarasi	Berilgan diapazon kvadratini hisoblang.
6	Minut miqdori	Sekund miqdori	Jami soniyalar miqdorini hisoblang
7	Soatlar miqdori	Minut miqdori	Jami mintlar miqdorini hisoblang
8	Haqiqiy son-birinchi kateti to'g'ri burchakli uchburchakning	Haqiqiy son-ikkinchi kateti to'g'ri burchakli uchburchakning	Uchburchakning yuzini hisoblash
9	Haqiqiy son-harakat tezligi (м/сек)	Butun son- harakat vaqti (min)	Bosib o'tilgan yo'lni hisoblash (m hisoblash)
10	Haqiqiy son-birinchi kateti to'g'ri burchakli uchburchakning	Haqiqiy son-ikkinchi kateti to'g'ri burchakli uchburchakning	Uchburchakning gipotenuzasini uzunligini hisoblang
11	Butun son – pastki asosi trapetsiyaning, balandligi-butun son	Butun son – ustki asosi trapetsiyaning	Trapetsiyaning yuzini hamda perimetrini hisoblang

12	Haqiqiy son-birinchi kateti to'g'ri burchakli uchburchakning	Haqiqiy son-ikkinchi kateti to'g'ri burchakli uchburchakning	Ikkinchi katet qarshisidagi tangens α hamda ichki va tashqi radiuslar orasidagi masofani hisoblang
13	Qo'ng'iroq qilingan vaqt s.m.sek larda. Qo'ng'iroq davomiyligi daqiqalarda	Qo'ng'iroq tugagan vaqt s.m.sek larda.Bir daqiqaning narxi	Qo'ng'iroqning umumiy narxini hisoblang hamd gaplashish vaqt oraligini chiqaring.
14	Tekislikdagi nuqta koordinatasi (gorizontal)	Tekislikdagi nuqta koordinatasi (vertikal)	Nuqta koordinatalari hamda Ox va Oy o'qlari bilan chegaralangan to'rtburchak perimetrini va diagonalini aniqlang.
15	Tekislikdagi nuqta koordinatasi (x_1, y_1)	Tekislikdagi nuqta koordinatasi (x_2, y_2)	Ikki nuqta orasidagi masofani hisoblang. Ox uqidan og'ish burchagini toping.
16	Tekislikdagi nuqta koordinatalari (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)	Tekislikdagi nuqta koordinatasi (x_2, y_2)	Koordinatalar asosida turtburchak yuzini hisoblang.
17	Tekislikdagi nuqta koordinatasi (x_1, y_1)	Tekislikdagi nuqta koordinatasi (x_2, y_2)	Ikki nuqta orasidagi masofani hisoblang. OY uqidan og'ish burchagini hamda koordinata boshigacha bo'lgan masofani toping.
18	Konus asosining aylanasi radiusi	Konusning balandligi	Konusning hajmini, yon siritini, asos yuzini, yon qirrasini hisoblang
19	Ish soatlari oy, kun, soat	Bir soatlik ish uchun to'lash	Ishning umumiy summasi, oy, kun, soat misolda hisoblang.