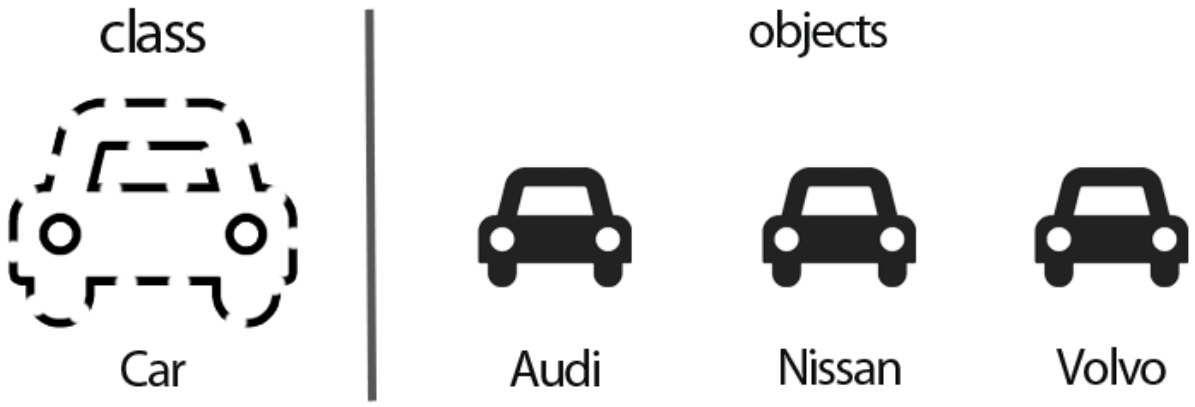
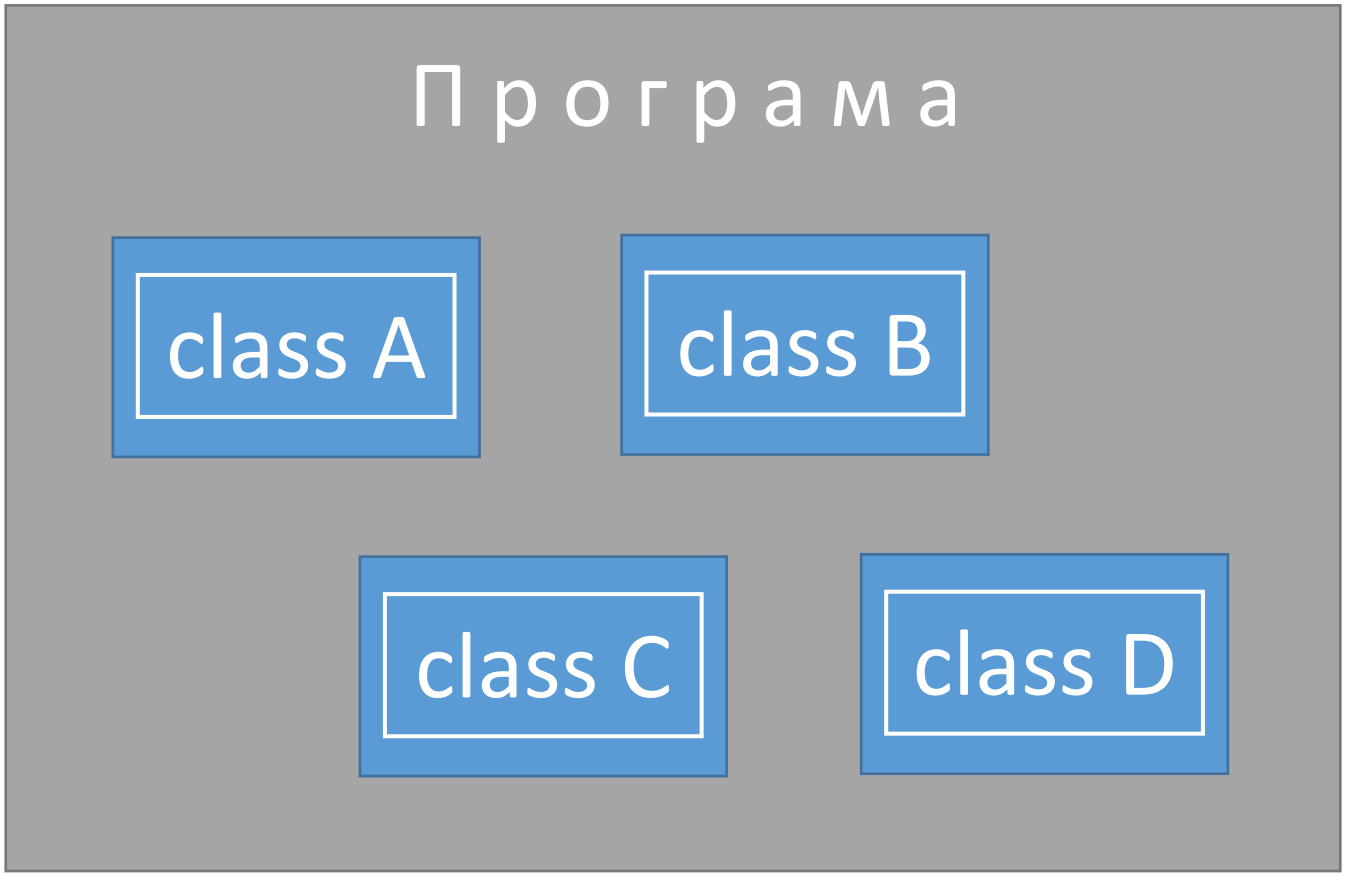


C# Клас

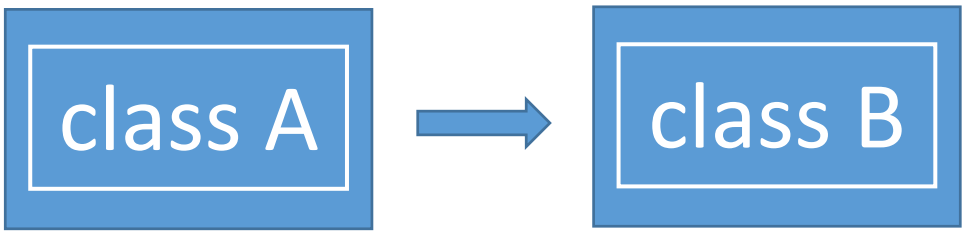
Андрій Янковський

2020

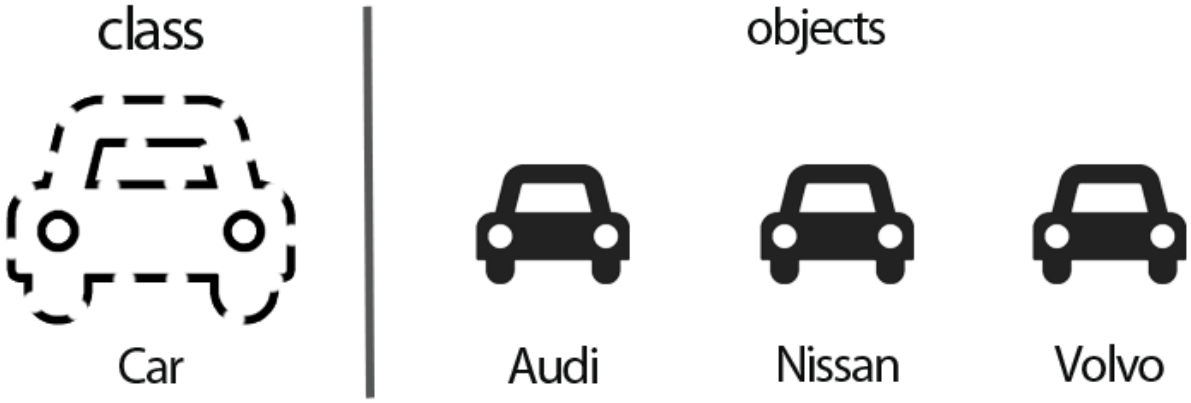
Класи розбивають велику програму на окремі модулі



Данні можна передавати з одного модуля в інший:



Клас Car – це лише опис того, що є всередині певного модуля



Аби використати клас – треба створити його змінну (об’єкт) класу:

```
Car audi = new Car();
Car nissan = new Car();
Car volvo = new Car();
```

Всередині класу можуть бути дані

```
int x;
int y;
```

і методи

```
int Add(int a, int b){
    int rezultat = a + b;
    return rezultat;
}

int Minus(int a, int b){
    int rezultat = a - b;
    return rezultat;
}
```

Данні – це змінні:

```
int a;
double b;
string c;
```

Методи – це назви блоків коду {...}. Всередині методу може і не бути команд, а може бути багато команд.

Ось пустий метод:

```
void Empty(){
    // немає команд
}
```

void – нічого не повертає в точку виклику метода.

Ось метод Message, який виводить на екран слово Hello:

```
void Message(){
      Console.WriteLine("Hello");
 }
```

Ось метод Add, що додає два числа і повертає результат (число типу int):

```
int Add(int a, int b){
    int rezultat = a + b;
    return rezultat;
}
```

А тепер давайте візьмемо якийсь системний клас, скажімо генератор випадкових чисел Random.
Щоб задіяти цей клас, треба спочатку об’явити змінну класу:

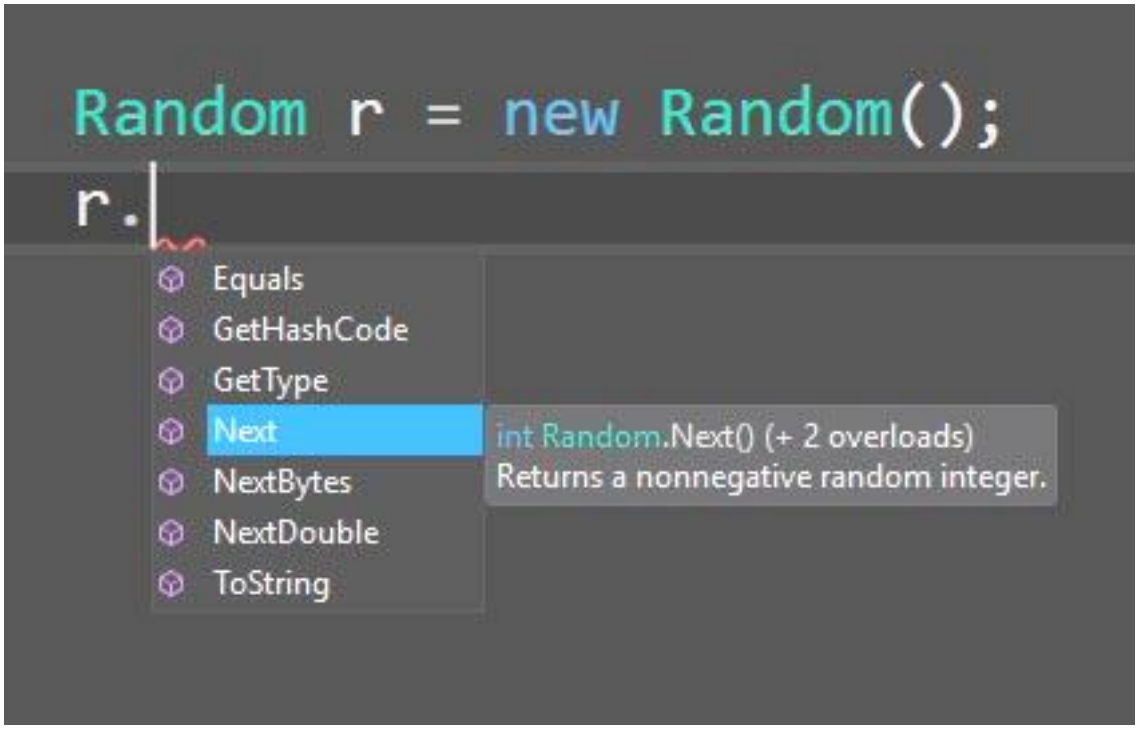
```
Random r = new Random();
```

Для того, щоб відрізнити змінну класу від звичайних змінних використовують слово об’єкт. Отже, ми щойно об’явили об’єкт r класу Random

Для того, аби об’єкт був робочім, ми виділити пам’ять під всі змінні, які в ньому знаходяться – це зробила команда new, а після new ми знову вказали назву класу але вже з ().

Як же ж тепер визивати методи класу?

Дуже просто, ми ставимо крапку після об’єкту і обираємо потрібний метод:



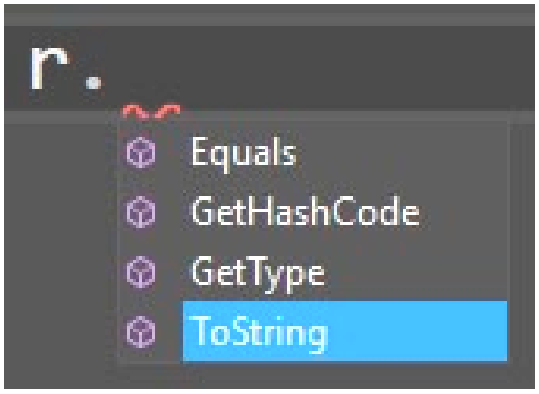
Системний клас Random підказує нам, що цей метод поверне додатне випадкове число

```
int a = r.Next(0, 10);
// 0 - 9
```

Давайте ще раз, метод Next знаходиться всередині класу Random; він приймає два числа і повертає випадкове число у відповідному діапазоні 0 – 9, тобто від першого переданого числа до другого переданого числа мінус один.

Також клас Random має метод NextDouble() – що генерує випадкове число від 0 до 1 з дробовою частиною:

```
double b = r.NextDouble();
// 0,545732851394328
```



А ці 4 стандартні методи Visual Studio додає усім об'єктам

Час настав!

Давайте створимо свій клас!

Клас Calculator з двома методами!

```
class Calculator{
   public int Add(int a, int b){
      int rezultat = a + b;
      return rezultat;
   }
   public int Minus(int a, int b){
      int rezultat = a - b;
      return rezultat;
   }
}
```

Напис public int перед назвами методів Add і Minus означає, що цей метод буде доступним для нас поза межами класу і що він повертає ціле число (int).

А як визивати методи Add і Minus?

Спочатку ми маємо створити об’єкт класу:

```
Calculator calc = new Calculator();
```

От тепер ми можемо визвати метод Add і передати туди два числа:

```
calc.Add(2, 3);
```

Але на екран нічого не виводиться?

Річ у тім, що метод Add повернув ціле число (int), але ми його ніде не використали і воно просто загубилося

А якщо так:

```
int z = calc.Add(3, 2);
Console.WriteLine("z = " + z);
```

Ось тепер на екран виведеться:

```
z = 5
```

Аналогічно:

```
int z = calc.Minus(3, 2);
Console.WriteLine("z = " + z);
```

```
z = 1
```

Ура!!! Ми написали свій перший клас!

А в класі можуть бути лише данні?

(без методів)

Звісно!

```
class Date
{
    public int day;
    public int month;
    public int year;
}
```

А як в цей клас щось записати?

Як завжди, ми маємо спочатку об’явити об’єкт класу:

```
Date d1 = new Date();
```

Надати значення змінним об'єкту:

```
d1.day = 1;
d1.month = 1;
d1.year = 2075;
```

А як вивести ці данні класу на екран?

```
Console.WriteLine(d1.day + "." +
                  d1.month + "." + d1.year);
                  // 1.1.2075
```

Цікаво, а є можливість зробити вивід даних через метод класу?

Щоб кожен раз оце не писати таку довгу команду.

Ми можемо створити метод Show(), який буде виводити усі дані об'єкту класу Date на екран.

Тобто, спочатку ми передамо данні в клас Date, а потім запустимо метод Show().

Класи легко розширюються, ми можемо просто додавати нові змінні і методи в них.

```
class Date{
    public int day;
    public int month;
    public int year;
    public void Show(){
      Console.WriteLine(day + "." +
                        month + "." + year);
  }
}
```

Клас сам по собі – це лише опис того, що в ньому є. Це як сліди на піску, комп'ютер ще не виділив пам'ять для змінних класу. Отже, ми маємо спочатку створити об'єкт класу

```
Date d1 = new Date();
```

Надати змінним об'єкту d1 початкові значення

```
d1.day = 1;
d1.month = 1;
d1.year = 2075;
```

і тепер визвати метод Show();

```
d1.Show();
// 1.1.2075
```

А як може виглядати клас побільше?

```
class Person
{
    public string name;
    public int age;
    public double height;
    public double weight;

    public void Walk() { }
    public void Talk() { }
    public void Eat() { }
    public void Sleep() { }
}
```

доступні поза класом дані

і

методи

А ще, ми можемо створити масив не тільки з стандартних типів даних int, double, string, а й типу Person:

```
Person[] mas = new Person[3];

mas[0] = new Person();
mas[1] = new Person();
mas[2] = new Person();
mas[0].name = "Dmytro";
mas[1].name = "Taras";
mas[2].name = "Olena";
// ще можна додати mas[0].age тощо

for ( int i = 0; i < mas.Length; i++ ){
    Console.Write(mas[i].name + ", ");
}
// Dmytro, Taras, Olena
```

Все ок, але залишається ще два питання:

1. Як класи обмінюються повідомленнями?

Визивають публічні методи один одного!

2. Якщо я не пишу перед змінною слово public, то як я це можу використати?

Це означає, що ця змінна має модифікатор доступу private (це можна не писати), тобто інші класи не мають доступу до неї! Тепер інші класи мають визвати якусь публічну функцію в нашому класі, яка до речі, може перевірити чи у нас достатньо грошей, і тільки після цього змінить значення приватної (захищеної) змінної.

```
class Customer {
    double money;    // EUR (private)

    public void Pay(double a) {
        if (money >= a) {
            Console.WriteLine("Successful
                    payment: " + a + " EUR");
            money = money - a;
        }
        else {
            Console.WriteLine("Error");
        }
    }
    public void AddMoney(double a) {
        money = money + a;
    }
}
```

```
class Metro {
    public double price = 0.5;  // EUR
}


static void Main(string[] args) {
    Customer customer1 = new Customer();
    customer1.AddMoney(1.5);   // EUR

    Metro metro = new Metro();

    for (int i = 0; i < 5; i++)
    {
        customer1.Pay(metro.price);
    }
}
```

Після запуску програми, цикл for запуститься 5 разів, ми 5 разів спробуємо проїхатися в метро, і об'єкт customer1 класу Customer виведе нам таку інформацію:

Successful payment 0,5 EUR

Successful payment 0,5 EUR

Successful payment 0,5 EUR

No money

No money

Після перших трьох поїздок в нас закінчилися гроші 1.5 EUR і в метро нас перестали пускати!

Тепер ви вільний митець!

Ви можете знайти наступні завдання на сайтах для фриланцерів