# **object-oriented programming (OOP)**

**Об'єктно-орієнтоване програмування (ООП)** - це модель комп'ютерного програмування, яка організовує розробку програмного забезпечення навколо даних або об'єктів, а не функцій і логіки. Об'єкт можна визначити як поле даних, яке має унікальні атрибути та поведінку.

**ООП** фокусується на об'єктах, якими розробники хочуть маніпулювати, а не на логіці, необхідній для маніпулювання ними. Цей підхід до програмування добре підходить для великих, складних програм, які активно оновлюються або підтримуються.

Сюди входять програми для виробництва та дизайну, а також мобільні додатки, наприклад, **ООП** можна використовувати для створення програмного забезпечення для моделювання систем.

Перший крок в **ООП** - зібрати всі об'єкти, якими програміст хоче маніпулювати, і визначити, як вони пов'язані один з одним - вправа, відома як моделювання даних.

Коли об'єкт відомий, йому присвоюється клас об'єктів, який визначає тип даних, які він містить, і будь-які логічні послідовності, які можуть ним маніпулювати. Кожна окрема логічна послідовність називається методом. Об'єкти можуть взаємодіяти за допомогою чітко визначених інтерфейсів, які називаються повідомленнями.

Зазвичай, оголошення класу містить лише ключове слово class, за яким слідує ідентифікатор (ім'я) класу. Але є деякі необов'язкові атрибути, які можна використовувати з оголошенням класу відповідно до вимог програми. Загалом, оголошення класів можуть містити такі компоненти, в порядку їх перерахування:

* **Модифікатори**: Клас може бути загальнодоступним, внутрішнім тощо. За замовчуванням модифікатор класу є внутрішнім.
* **Ключове слово class**: Ключове слово class використовується для оголошення типу class.
* **Ідентифікатор класу**: Передбачена змінна типу class. Ідентифікатор (або ім'я класу) повинен починатися з початкової літери, яка за домовленістю має бути великою.
* **Базовий клас або суперклас**: Ім'я батька класу (суперкласу), якщо такий є, перед яким ставиться символ : (двокрапка). Це не є обов'язковим.
* **Інтерфейси**: Розділений комами список інтерфейсів, реалізованих класом, якщо такі є, перед яким ставиться : (двокрапка). Клас може реалізовувати більше одного інтерфейсу. Це не є обов'язковим.
* **Тіло**: тіло класу оточено { } (фігурні дужки).

**Яка структура об'єктно-орієнтованого програмування?**

Структура, або будівельні блоки, об'єктно-орієнтованого програмування включають в себе наступне:

* **Класи (Classes)** - це визначені користувачем типи даних, які виступають в якості шаблону для окремих об'єктів, атрибутів і методів.
* **Об'єкти** **(Objects)** - це екземпляри класу, створені з конкретно визначеними даними. Об'єкти можуть відповідати об'єктам реального світу або абстрактним сутностям. Коли клас визначається спочатку, опис є єдиним визначеним об'єктом.
* **Методи (Methods)** - це функції, визначені всередині класу, які описують поведінку об'єкта. Кожен метод, що міститься в описі класу, починається з посилання на об'єкт-екземпляр. Крім того, підпрограми, що містяться в об'єкті, називаються **методами екземпляра (instance object)**. Програмісти використовують методи для багаторазового використання або збереження функціональності, інкапсульованої в одному об'єкті за один раз.
* **Атрибути (Attributes)** - визначаються у шаблоні класу і представляють стан об'єкта. Об'єкти будуть мати дані, що зберігаються в полі атрибутів. Атрибути класу належать самому класу.

**Які основні принципи ООП?**

Об'єктно-орієнтоване програмування базується на наступних принципах:

* **Інкапсуляція (Encapsulation).** Цей принцип полягає в тому, що вся важлива інформація міститься всередині об'єкта, а назовні виноситься лише обрана інформація. Реалізація та стан кожного об'єкта зберігаються в закритому режимі всередині певного класу. Інші об'єкти не мають доступу до цього класу або повноважень вносити зміни. Вони можуть лише викликати список загальнодоступних функцій або методів. Така характеристика приховування даних забезпечує більшу безпеку програми та дозволяє уникнути ненавмисного пошкодження даних.
* **Абстракція (Abstraction)**. Об'єкти розкривають лише внутрішні механізми, які є важливими для використання інших об'єктів, приховуючи будь-який непотрібний код реалізації. Похідний клас може мати розширену функціональність. Ця концепція може допомогти розробникам легше вносити додаткові зміни або доповнення з часом.
* **Успадкування (Inheritance)**. Класи можуть повторно використовувати код з інших класів. Зв'язки та підкласи між об'єктами можуть бути призначені, що дозволяє розробникам повторно використовувати загальну логіку, зберігаючи при цьому унікальну ієрархію. Ця властивість ООП змушує ретельніше аналізувати дані, скорочує час розробки та забезпечує вищий рівень точності.
* **Поліморфізм(Polymorphism)**. Об'єкти призначені для обміну поведінкою, і вони можуть набувати більш ніж однієї форми. Програма визначить, яке значення або використання необхідне для кожного виконання цього об'єкта з батьківського класу, зменшуючи потребу в дублюванні коду. Потім створюється дочірній клас, який розширює функціональність батьківського класу. Поліморфізм дозволяє різним типам об'єктів проходити через той самий інтерфейс.

https://www.geeksforgeeks.org/garbage-collection-in-c-sharp-dot-net-framework/